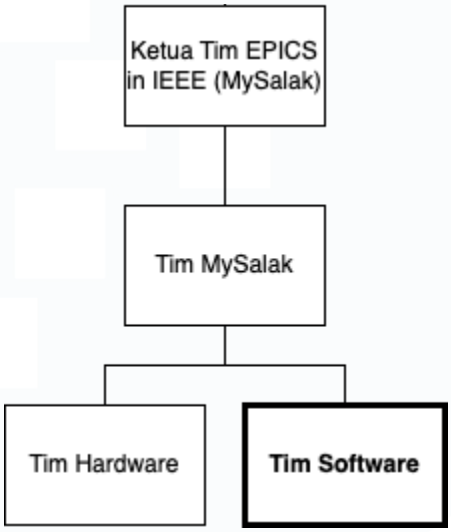
BAB III

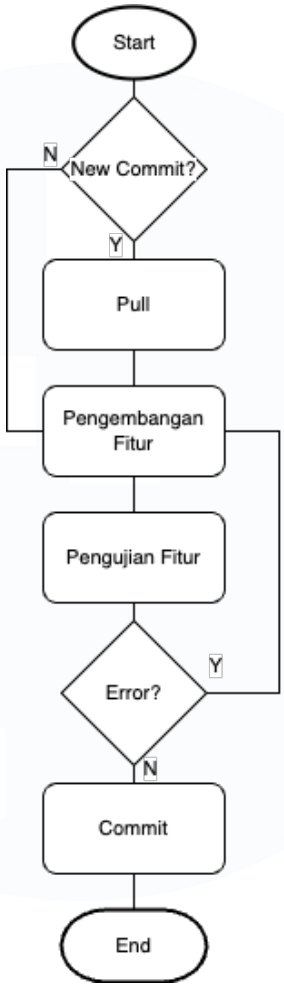
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi



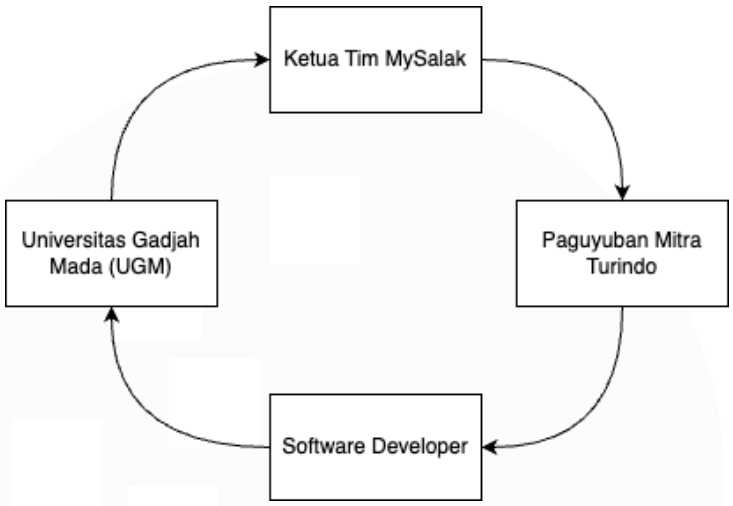
Gambar 3.1 Struktur organisasi MySalak

Seperti yang terlihat pada gambar 3.1, tim MySalak terbagi menjadi 2 divisi, yaitu tim *hardware* dan tim *software*. Sebagai software developer, penulis berada pada tim software, dan bertanggung jawab secara langsung ke Ibu Nabila Husna Shabrina, yang merupakan ketua tim MySalak. Kolaborasi juga dilakukan secara langsung dengan tim hardware untuk mengkoordinasi pengiriman data dari *node hardware* yang telah dibuat ke server atau *backend,* yang kemudian datanya akan diproses lebih lanjut untuk menghasilkan informasi yang relevan bagi para petani.



Gambar 3.2 Alur kerja divisi *software* MySalak

Dalam pengembangan aplikasi MySalak, GitHub digunakan sebagai sarana berbagi dan sinkronisasi *file*, yang memanfaatkan *repository* atau sejenis penyimpanan file online yang berbasis Git [9]. Detail proses kerja tiap *software developer* terlihat pada gambar 3.2, yang dimulai dengan mengecek apakah ada *commit* atau perubahan baru di *repository*, dan melakukan *pull* atau mensinkronisasi perubahan kepada file *local,* jika ada. Kemudian, *developer* akan mengembangkan fitur yang merupakan tanggung jawabnya, dan diikuti oleh pengujian fitur secara mandiri. Jika ditemukan *error*, maka akan dilakukan *bug checking*, untuk menemukan letak kesalahan, dan *bug fixing*, untuk memperbaikinya. Jika fitur sudah lulus pengujian, maka perubahan akan di-*commit* ke *repository*.



Gambar 3.3 Alur koordinasi divisi *software*

Fitur-fitur yang dikembangkan merupakan hasil diskusi dari seluruh pihak terkait, dan pembagian pengerjaan fitur untuk tiap *software developer* dilakukan secara langsung oleh ketua tim. Jika fitur sudah selesai dikerjakan, *developer* langsung menyampaikannya pada ketua tim untuk melakukan validasi fitur melalui pesan ataupun saat *meeting*. *Update* juga dilakukan secara berkala ke pihak UGM dan Paguyuban Mitra Turindo melalui pertemuan tatap muka maupun *online meeting* untuk membahas *progress* pengerjaan aplikasi. Gambaran umum alur koordinasi yang dilakukan dapat dilihat pada gambar 3.3.

3.2 Tugas dan Uraian Kerja Magang

Software Developer bertugas untuk mengembangkan aplikasi MySalak dengan teknologi PWA, yang memiliki berbagai fitur penting untuk memenuhi kebutuhan petani salak. Dalam pengembangan aplikasi, React.js digunakan bersama ChakraUI untuk pengembangan front-end, diikuti dengan Express.js untuk pengembangan back-end yang berkaitan dengan data dan autentikasi, dan Flask untuk back-end yang berkaitan dengan model-model kecerdasan buatan. Untuk detail *timeline* pekerjaan yang dilakukan, dapat dilihat pada tabel 3.1.

Tabel 3.1 *Timeline* Kerja Magang

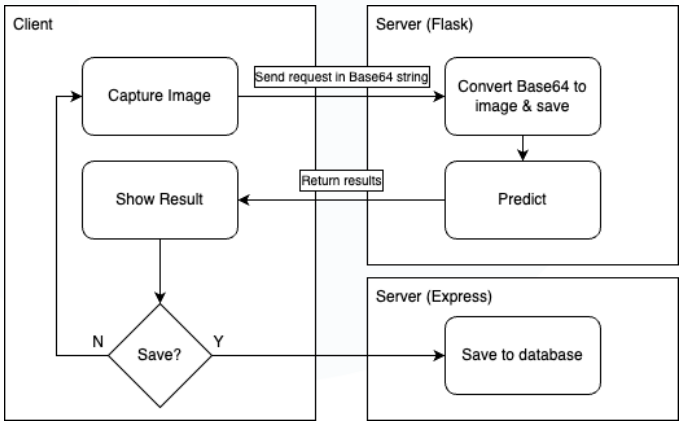
Bulan	Minggu	Kegiatan
November	1	Rapat bersama tim untuk mendiskusikan fitur-fitur yang penting untuk diimplementasikan.
	2-3	Pengembangan fitur Hitung Hama.
	4	Pengembangan fitur Persebaran Hama.
Desember	1	Pengembangan fitur Profil.
	2	Pengembangan menu Admin UMN.
	3	Pengembangan menu Admin Dinas.
	4	Pengembangan menu Admin Dinas dan fitur C*lient-side Authorization.*
Januari	1-3	Finalisasi aplikasi, *bug fixing*, dan peluncuran aplikasi di Sleman.
	4	*bug fixing* setelah *launching*.
Februari	1-4	Periode pembuatan laporan magang.

Ada sejumlah fitur yang dikembangkan selama periode 4 bulan tersebut, meliputi hitung hama, persebaran hama, profil, *client-side authorization*, admin UMN, dan admin Dinas.

3.2.1 Hitung Hama

Gambar 3.4 Tampilan awal menu Hitung Hama

Fitur hitung hama menggunakan model YOLOV5 untuk melakukan deteksi lalat pada gambar, menampilkan hasilnya pada aplikasi. Pada aplikasi MySalak, fitur ini dapat diakses dengan menekan tombol "*Scan*" yang berada di bagian tengah *navigation bar* (*navbar*) saat pengguna masuk sebagai petani. Tampilan awal dari fitur ini adalah tampilan kamera dengan tombol untuk mengambil foto dan memutar kamera, seperti yang terlihat pada gambar 3.4. Menu ini dikembangkan dengan library react-webcam yang memungkinkan akses ke kamera pengguna [10].



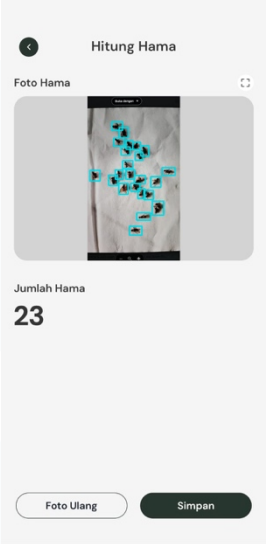
Gambar 3.5 Diagram alur menu Hitung Hama

Cara kerja fitur ini dapat dilihat pada gambar 3.5. Setelah foto ditangkap menggunakan *shutter button*, foto terdsebut akan dikirim sebagai *request* dalam format Base64 langsung ke API endpoint dari back-end Flask untuk segera diproses menggunakan model YOLOV5. Pada sisi back-end, *string* Base64 yang diterima akan di-*decode* menjadi sebuah gambar dan disimpan dengan nama acak yang di-*generate* dengan UUID. Kemudian *script* detect.py dari *repository* YOLOV5 akan dipanggil dengan gambar tadi sebagai argumen, disertai sejumlah hyperparameter lain. Potongan kode back-end dapat dilihat pada gambar 3.6.

```python
def detect_lalat():
  file = request.json['image']

  if "data:image" in file:
      base64_string = file.split(",")[1]
      file = base64toimage(base64_string)

  if not os.path.exists('images'):
      os.makedirs('images')

  uid = uuid.uuid4()

  image_path = os.path.join('images', f'{str(uid)}.jpg')

  with open(image_path, 'wb') as file_converted:
      file_converted.write(file)

  detected = subprocess.run(["python", "./yolov5-mysalak/detectModif.py",
                             "--weights", "./Model/Lalat/best.pt",
                             "--source", f'./images/{uid}.jpg',
                             "--save-txt",
                             "--save-conf",
                             "--hide-labels",
                             "--conf-thres", '0.5',
                             "--line-thickness", '3'
                             ],
                             capture_output=True,
                             text=True)

  print(detected)

  if detected.returncode == 0:
    output = json.loads(detected.stdout)
    return jsonify(output)
  else:
    return jsonify({"error": "Detection failed", "details": detected.stderr}), 500
```

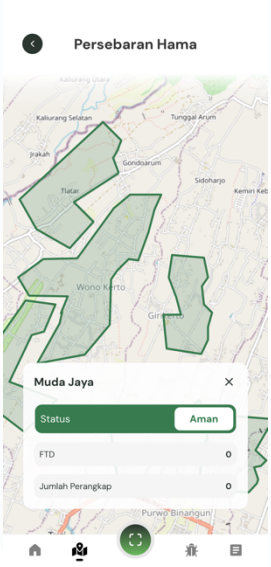
Gambar 3.6 Potongan kode back-end Hitung Hama

Hasil deteksi yang didapatkan akan dikembalikan sebagai *response* ke *client*, dan berupa gambar dengan lalat yang telah ditandai, dengan jumlah lalat yang terdeteksi, seperti pada gambar 3.7. Pengguna kemudian dapat menekan foto atau tombol di pojok kanan atas untuk memperbesar tampilan gambar. Setelah diperiksa, pengguna dapat menekan tombol "Simpan" untuk menyimpan data perhitungan ke server, atau tombol "Foto Ulang" untuk mengambil foto kembali. Saat tombol "Simpan" ditekan, jumlah yang terdeteksi beserta dengan kelompok tani pengguna akan dikirimkan sebagai *payload* dengan metode POST ke API *endpoint* yang sesuai, dan disimpan ke dalam *database*.



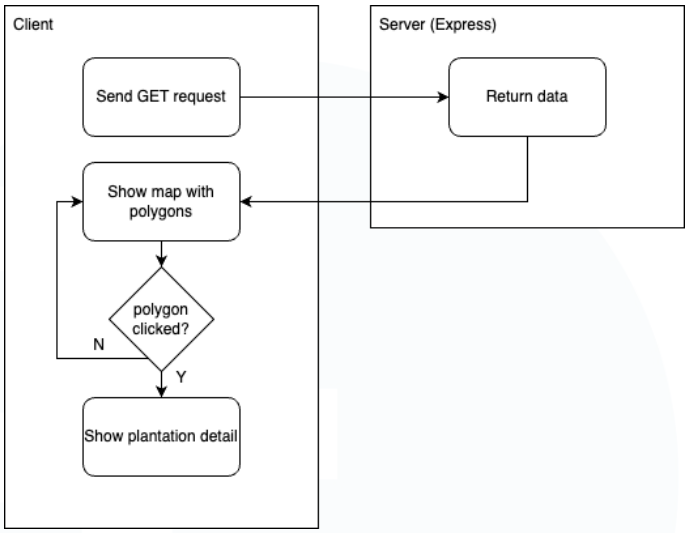
Gambar 3.7 Tampilan hasil perhitungan hama

3.2.2 Persebaran Hama



Gambar 3.8 Tampilan menu Persebaran Hama

Fitur persebaran hama dapat diakses oleh semua pengguna dengan menekan tombol berlogo "Peta" yang terletak pada urutan kedua dari kiri di navbar. Menu ini menampilkan informasi berupa peta dengan blok-blok kelompok tani yang tergabung ke dalam Paguyuban Mitra Turindo. Jika salah satu blok ditekan, maka akan muncul tampilan kecil yang menampilkan status, FTD (*fruit flies per trap per day*), dan jumlah perangkap dari kelompok tani tersebut, seperti pada gambar 3.8.



Gambar 3.9 Diagram alur menu Persebaran Hama

Pembuatan peta dilakukan dengan menggunakan library Leaflet [11], beserta dengan Leaflet Polygon untuk membuat tiap blok yang ada, dengan cara mendefinisikan seluruh titik batasan koordinat dari suatu kelompok tani, dengan alur seperti pada gambar 3.9. Seluruh koordinat tersebut disimpan pada *server*, dan akan di-*fetch* bersama dengan data kelomopok tani, termasuk FTD dan jumlah perangkap, saat menu diakses, untuk menampilkan blok setiap kelompok tani dan informasi yang sesuai. Jika salah satu blok ditekan, maka akan muncul informasi yang berupa status kebun, FTD, dan jumlah perangkap. Potongan kode yang digunakan untuk menampilkan peta dapat dilihat pada gambar 3.10.

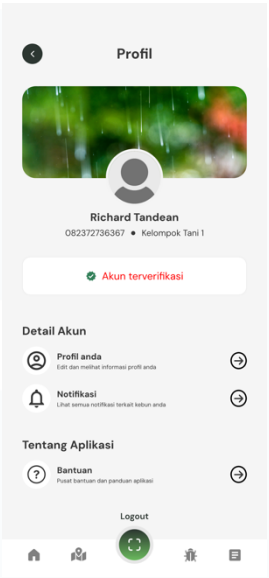
```
return (
  <MapContainer
    center={[latitude, longitude]}
    zoom={14}
    ref={mapRef}
    zoomControl={false}
    style={{
      height: "96vh",
      width: "inherit",
      zIndex: 1,
      position: "absolute",
      top: 0,
      left: 0,
    }}
  >
    <TileLayer
      attribution='&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
      url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
    />

    {polygons.map((item) => (
      <Polygon
        positions={item.coordinates}
        pathOptions={{
          color:
            sebaran[item.id - 1]?.KelompokTani.ftd > 1
              ? "#850707"
              : sebaran[item.id - 1]?.KelompokTani.ftd > 0.8
              ? "#ACA714"
              : "#377B4E"
        }}
        eventHandlers={{
          click: () => {
            props.handleDetail(item.id - 1);
          },
        }}
      />
    ))}
  </MapContainer>
);
```

Gambar 3.10 Potongan kode untuk menampilkan peta

Warna tiap blok kelompok tani dipengaruhi oleh status dari kelompok tersebut, yang dihitung berdasarkan FTD, dengan rincian sebagai berikut:

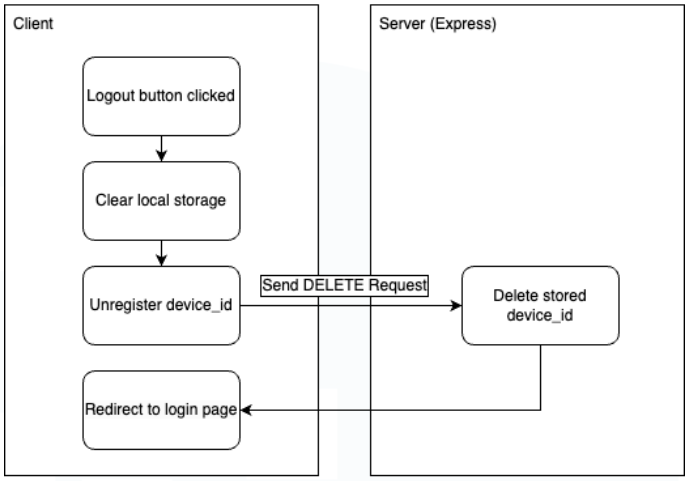
- FTD di bawah 0.8 : aman, berwarna hijau
- FTD di antara 0.8-0.9 : waspada, berwarna kuning
- FTD di atas 0.9 : bahaya, berwarna merah

3.2.3 Profil



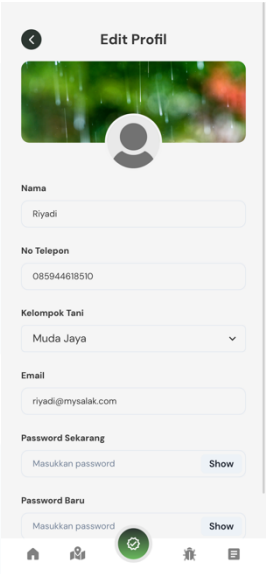
Gambar 3.11 Tampilan menu Profil

Menu profil dapat diakses dengan cara menekan tombol berupa foto profil petani di *dashboard*. Laman profil menampilkan beberapa informasi berupa nama, nomor telepon, dan kelompok tani pengguna, serta status akun petani seperti yang terlihat pada gambar 3.11. Seluruh informasi tersebut telah disimpan pada *local storage* saat *login* berhasil dilakukan, dan diakses pada saat menu profil dibuka, sehingga *fetch* tidak perlu dilakukan. Saat akun petani baru diregistrasi, akun petani bersifat belum diverifikasi, yang membuatnya tidak bisa mengunggah hasil perhitungan lalat buah. Pada laman ini juga terdapat tombol *logout*, yang akan menghapus semua data di *local storage*, dan membuat pengguna tidak terotentikasi lagi. Alur *logout* dapat dilihat pada gambar 3.12.



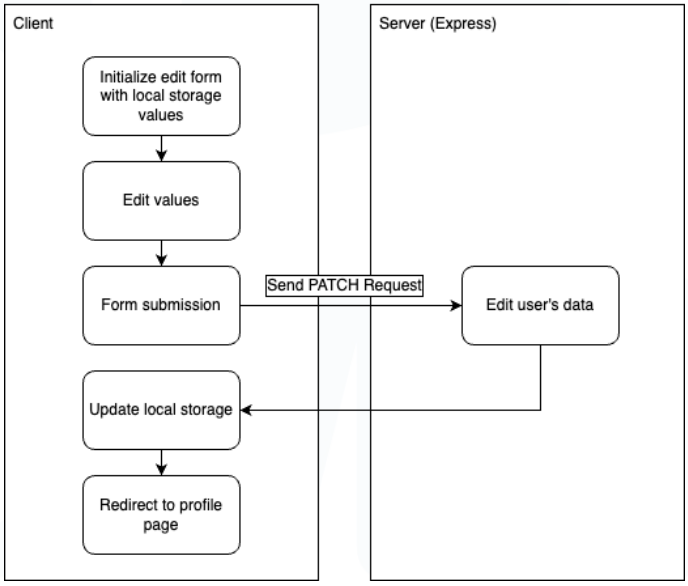
Gambar 3.12 Diagram alur *logout*

Ada 3 fitur lain yang dapat diakses melalui menu ini, yaitu "Profil Anda", "Notifikasi", dan "Bantuan". Menu "Profil Anda" dapat digunakan untuk mengubah informasi akun, termasuk *foto banner*, foto profil, nama, nomor telepon, kelompok tani (jika masuk sebagai petani atau ketua kelompok tani), dan password (jika masuk sebagai admin) seperti pada gambar 3.13. Seluruh data yang relevan akan diambil dari *local storage* untuk mengisi *form fields*, dan kemudian seluruh data yang diinput akan digunakan sebagai *payload* ke *server* dengan metode PUT saat tombol simpan ditekan, bersama dengan *value* pada *local storage* yang disesuaikan.

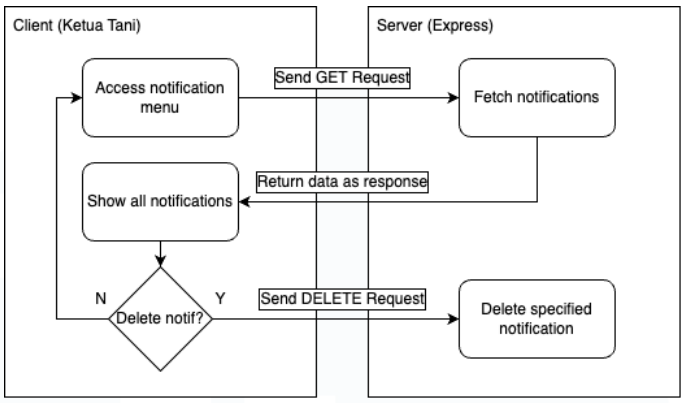


Gambar 3.13 Tampilan menu Edit Profil

Sedangkan, alur untuk melakukan update dapat dilihat pada gambar 3.14.

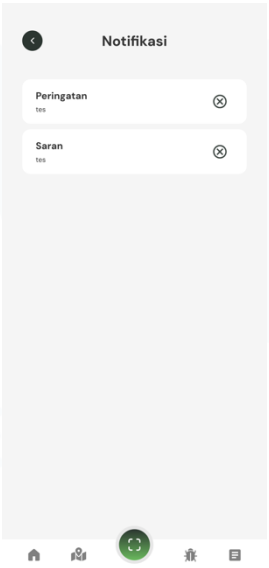


Gambar 3.14 Diagram alur Edit Profil



Gambar 3.15 Diagram alur Notifikasi

Menu selanjutnya, yaitu "Notifikasi" hanya dapat diakses oleh para ketua kelompok tani, baik di akun admin maupun petani miliknya. Gambar 3.15 menunjukkan alur menu notifikasi, yang dimulai dari saat menu diakses, kemudian seluruh notifikasi yang ditujukan untuk ketua tani yang sedang masuk di-*fetch* dari *server*, dan hingga akhirnya ditampilkan dalam bentuk *card*, seperti pada gambar 3.16. Tombol berlogo "ⓧ" yang ada dapat ditekan untuk menghapus notifikasi yang dipilih, dengan cara mengirimkan ID dari notifikasi tersebut sebagai payload ke API endpoint yang sesuai.



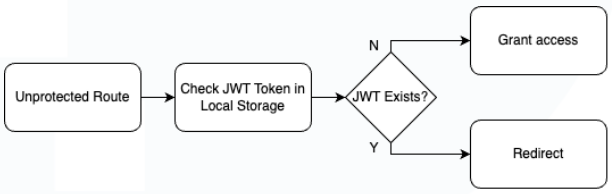
Gambar 3.16 Tampilan menu Notifikasi

Kemudian, menu "Bantuan" akan membuka laman linktr.ee MySalak yang berisi panduan, dan kontak tim MySalak yang dapat dituju untuk melaporkan kendala yang dialami.

3.2.4 Client-side Authorization

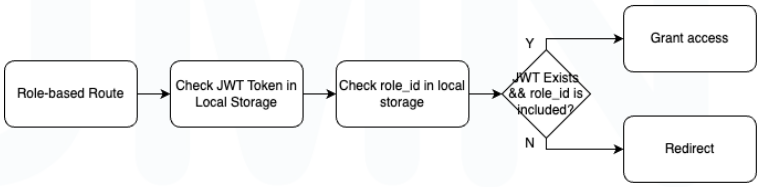
Client-side authorization merupakan sebuah fitur untuk membatasi akses pengguna pada halaman tertentu. Pada aplikasi MySalak, ada 5 role yang terdiri dari UMN, UGM, ketua kelompok tani, dinas pertanian, dan petani. Kelima role tersebut memiliki menu khususnya masing-masing, dan hanya dapat diakses oleh role yang sesuai.

Ada 3 jenis komponen yang digunakan sebagai filter pada setiap komponen Route pada aplikasi MySalak, yang terdiri dari:



Gambar 3.17 Diagram alur *Unauthorized Route*

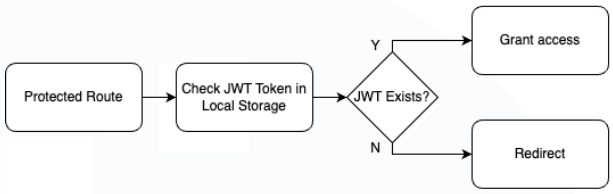
- *Unauthorized Route*, yang mengecek apakah pengguna telah *login*, dan akan melakukan redirect ke *login page* jika belum, dengan alur seperti pada gambar 3.17. Filter ini digunakan untuk membatasi pengguna yang telah masuk untuk mengakses laman tertentu, seperti laman autentikasi.



Gambar 3.18 Diagram alur *Authorized Route*

- *Authorized Route*, yang akan melakukan *redirect* menuju *dashboard* jika pengguna telah *login* tetapi mencoba untuk mengakses *login page,* dengan alur seperti pada gambar 3.18. Filter ini diterapkan pada menu-menu yang dapat diakses oleh semua jenis pengguna, selama pengguna tersebut sudah terautentikasi.

Menu yang termasuk berupa fitur-fitur umum seperti *dashboard*, profil, persebaran hama, manajemen hama, dan artikel



Gambar 3.19 Diagram alur *Role-based Route*

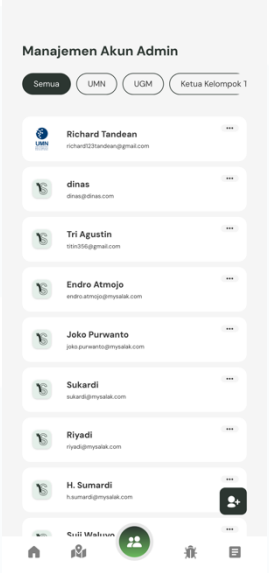
- *Role-based Route*, yang diterapkan dengan cara mengecek apakah pengguna telah login, beserta role_id pengguna yang tersimpan pada *local storage*, dan akan mengembalikan pengguna ke *dashboard* jika role_id tersebut tidak termasuk dalam *role* yang diperbolehkan, seperti pada gambar 3.19. Filter ini digunakan untuk mencegah pengguna dengan *role* tertentu mengakses fitur khusus dari *role* lain.

Contoh penggunaan ketiga komponen tersebut dapat dilihat pada gambar 3.20

```
<Routes>
  <Route element={<UnauthorizedRoute />}>
    <Route path="/" element={<LoginPage />} />
    <Route path="/create-petani" element={<PetaniCreate />} />
    <Route path="/login-petani" element={<PetaniLogin />} />
    <Route path="/login-admin" element={<AdminLogin />} />
  </Route>
  <Route element={<ProtectedRoutes />}>
    <Route path="/dashboard" element={<Dashboard />} />
    <Route path="/persebaran-hama" element={<PersebaranHama />} />
    <Route path="/kamera" element={<Camera />} />
    <Route path="/manajemen-hama" element={<ManajemenHamaPage />} />
    <Route path="/profil" element={<Profil />} />
    <Route path="/edit-profil" element={<EditProfil />} />
    <Route path="/ramalan-cuaca" element={<RamalanCuaca />} />
    <Route path="/artikel" element={<Artikel />} />
    <Route path="/artikel/:id" element={<ArticleDetail />} />
  </Route>

  {/* admin umn */}
  <Route element={<RoleBasedRoutes allowed={[1]} />}>
    <Route
      path="/admin/manajemen-admin"
      element={<ManajemenAdmin />}
    />
    <Route
      path="/admin/manajemen-admin/tambah"
      element={<TambahAdmin />}
    />
    <Route
      path="/admin/manajemen-admin/:id/edit"
      element={<EditAdmin />}
    />
    <Route
      path="/admin/manajemen-admin/daftar-ketua-tani"
      element={<DaftarKetuaTani />}
    />
  </Route>
```

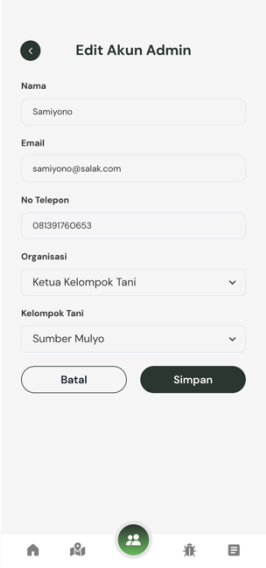
Gambar 3.20 Penggunaan komponen otorisasi

3.2.5 Admin UMN



Gambar 3.21 Tampilan menu Manajemen Akun Admin

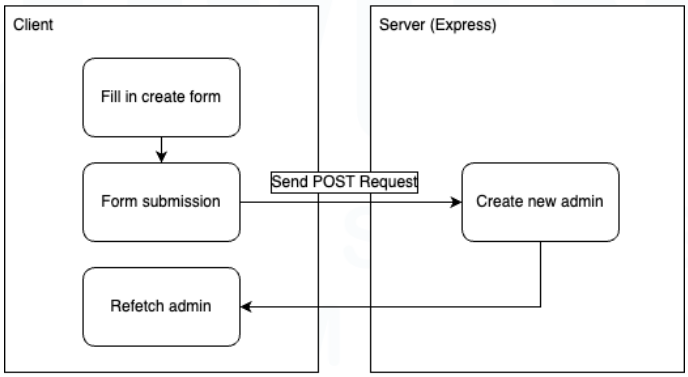
Fitur khusus dari admin UMN adalah fitur manajemen akun admin, yang dapat dilihat pada gambar 3.21. Admin UMN bertindak sebagai administrator utama dari aplikasi MySalak, yang memiliki hak atas pembuatan, pengubahan informasi, maupun penghapusan akun admin. Pada menu ini, terdapat juga filter yang dapat menyortir daftar admin berdasarkan role dari akun mereka masing-masing. Saat menu diakses, data akan di-fetch dari *server*, dan filter pada sisi *client*, dengan mencocokkan "role_id" dari tiap data.



Gambar 3.22 Tampilan menu Edit Akun Admin

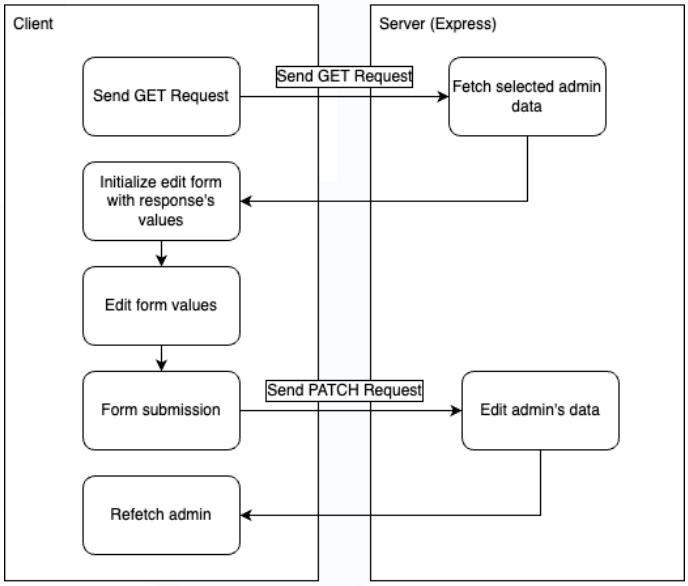
Berbeda dari registrasi petani yang dapat dilakukan via *login page*, akun admin hanya dapat didaftarkan oleh admin UMN dengan menekan tombol di pojok bawah kanan. Sedangkan fitur *edit* dan *delete* admin dapat diakses dengan menekan tombol "⋯" pada pojok kanan atas *card* tiap akun. Tampilan edit informasi admin dapat dilihat pada gambar 3.22. Sedangkan, penghapusan akun admin dilakukan dengan cara *soft delete* untuk mencegah terjadinya *error* pada relasi data.

Registrasi dan edit akun admin memiliki tampilan yang nyaris sama, dengan perbedaan di penambahan kolom *password* pada registrasi.



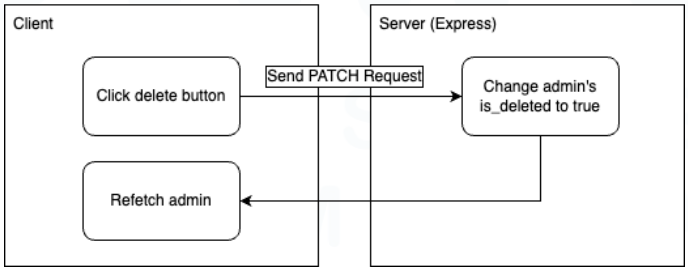
Gambar 3.23 Diagram alur registrasi admin

Pada registrasi admin, pengguna cukup mengisi informasi pada form yang tersedia, lalu menekan tombol "Simpan", yang akan mengirim seluruh data tersebut sebagai payload dengan metode POST ke API endpoint, yang kemudian akan menyimpan data tersebut. Seluruh alur fitur ini dapat dilihat pada gambar 3.23.



Gambar 3.24 Diagram alur edit admin

Fitur edit admin memiliki alur yang cukup mirip dengan registrasi admin, dengan perbedaan yang terletak pada bagian awal, di mana form diinisialisasi dengan nilai dari admin yang dipilih. Nilai tersebut didapatkan dengan cara melakukan *fetch* ke *server* menggunakan sebuah GET *request* yang menyertakan ID dari admin tersebut. Proses yang terjadi tergambarkan pada gambar 3.24



Gambar 3.25 Diagram alur delete admin

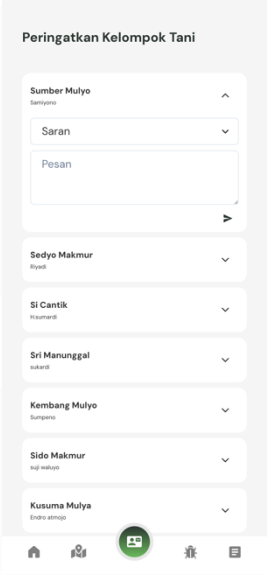
Untuk penghapusan akun, proses yang terjadi diilustrasikan pada gambar 3.25. Saat tombol hapus diteknan, ID dari admin yang dipilih akan disertakan sebagai query parameter untuk API endpoint yang sesuai, dengan metode PATCH, hasilnya, status "is_deleted" dari admin tersebut akan diubah menjadi *true*. Potongan kode back-end untuk proses ini terlihat pada gambar 3.26.

```
adminAuthRouter.patch(
  "/:id/delete",
  verifyJwt,
  checkAdminRole([1]),
  async (req, res) => {
    try {
      const admin = await Admin.update(
        {
          is_deleted: true,
        },
        {
          where: {
            id: req.params.id,
          },
        }
      );

      if (response === 0)
        return res.status(404).json({ msg: "Employee not found" });
      else return res.status(200).json({ msg: "Employee Deleted", admin });
    } catch (e) {
      console.log(e);
    }
  }
);
```

Gambar 3.26 Potongan kode back-end untuk *soft delete* admin

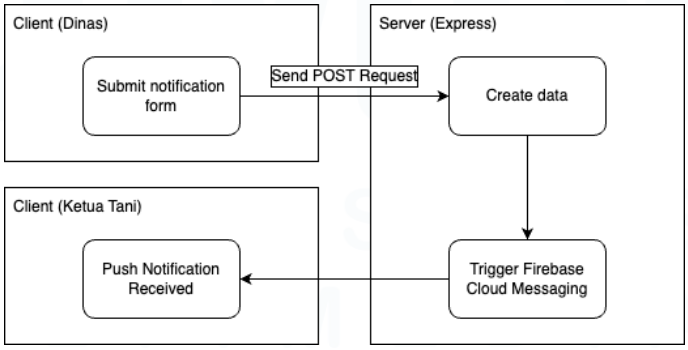
Setelah ketiga proses tersebut dilakukan, data pada menu manajemen admin akan di-*fetch* kembali untuk menampilkan data terkini.

3.2.6 Admin Dinas



Gambar 3.27 Tampilan menu Peringatkan Kelompok Tani

Pengguna dengan role dinas pertanian memiliki fitur khusus untuk memperingatkan kelompok tani, seperti yang terlihat pada gambar 3.27. Ada 3 kategori peringatan yang dapat disampaikan, yaitu saran, himbauan, dan peringatan. Untuk mengirimkan notifikasi, pengguna hanya perlu memilih kelompok tani tertentu, lalu memilih kategori, kemudian mengisi pesan, dan menekan tombol kirim. Seluruh saran, himbauan, ataupun peringatan yang diberikan akan disampaikan sebagai *push notification* kepada perangkat ketua dari kelompok tani yang dimaksud.



Gambar 3.28 Alur diagram menu Peringatkan Kelompok Tani

Alur proses pengiriman notifikasi dapat dilihat pada gambar 3.28. Pesan yang ditulis berdasarkan kategori dan kelompok tani akan dikirimkan sebagai *payload* ke API *endpoint* yang sesuai, dengan metode POST. Kemudian, Firebase Cloud Messaging (FCM) akan digunakan untuk mengirim *push notification* ke akun ketua kelompok tani yang sesuai. Ketua tani tersebut dapat melihat saran dari dinas pertanian pada menu "Notifikasi" yang dapat diakses dari menu "Profil". Potongan kode back-end yang bertanggung jawab untuk menyimpan pesan dan mengirimkan notifikasi dapat dilhat pada gambar 3.29.

```
dinasNotifyRouter.post(
  "/notify-ketua-tani",
  verifyJwt,
  checkAdminRole([4]),
  async (req, res) => {
    try {
      const validate = await dinasNotification.safeParseAsync(req.body);

      if (!validate.success) {
        return res.status(422).json({
          message: parseZodError(validate.error),
        });
      }

      const { dinas_id, kelompok_tani, category, message } = validate.data;

      await DinasNotification.create({
        dinas_id,
        kelompok_tani,
        category,
        message,
      });

      const fromPetani = await Petani.findOne({
        where: {
          is_admin: true,
          kelompok_tani: kelompok_tani,
        },
      });

      const fromAdmin = await Admin.findOne({
        where: {
          role_id: 3,
          kelompok_tani: kelompok_tani,
        },
      });

      const tokenFromPetani = await PushNotificationToken.findOne({
        where: {
          user_id: fromPetani?.id,
        },
      });

      const tokenFromAdmin = await PushNotificationToken.findOne({
        where: {
          user_id: fromAdmin?.email,
        },
      });

      const token = [tokenFromPetani?.token, tokenFromAdmin?.token].filter(
        (t) => t !== null
      );

      const uniqueTokens = [...new Set(token)];

      const payload = {
        tokens: uniqueTokens,
        notification: {
          title: category,
          body: message,
        },
      };

      const response = await admin.messaging().sendEachForMulticast(payload);
      res.status(200).json({
        message: "Notification sent successfully",
        response: response,
      });
    } catch (err) {
      res.status(500).json({ message: err.message });
    }
  }
);
```

Gambar 3.29 Potongan kode back-end untuk mengirim notifikasi

3.3 Kendala yang Ditemukan

Selama melakukan magang sebagai *software developer* di MySalak, ada beberapa kendala yang ditemukan, meliputi:

- **Kesulitan dalam mengintegrasikan model YOLOV5**

  Pada awalnya, endpoint YOLOV5 dikembangkan dengan cara mengimport file detect.py dari *repository* ultralytics/yolov5, yang berisi

script untuk melakukan *object detection* dengan model tersebut. Saat model diubah dengan model terbaru yang memiliki performa lebih baik, cara tersebut berakhir dengan pemrosesan *request* yang tersangkut tepat sebelum melakukan prediksi, dan berakhir menjadi *request timed out*.

- **Model YOLOV5 menghasilkan *response* yang sama untuk beberapa *request* berbeda pada waktu yang sama**

  Setelah ditelusuri, hal ini terjadi karena seluruh gambar yang diterima disimpan dengan nama yang sama, yang awalnya bertujuan untuk menghemat storage. Nama yang sama ini membuat hanya ada 1 foto yang dapat diproses dalam suatu waktu, sehingga seluruh request yang diterima pada waktu yang sama mendapatkan response yang sama pula.

- **Kendala pada fitur edit foto profil dan foto banner**

  Dalam mengubah foto profil dan foto banner, terkadang foto tidak terganti meskipun gambar telah berhasil diunggah dan tombol simpan telah ditekan.

- **Marker pada persebaran hama yang berukuran statis**

  Awalnya, seluruh kelompok tani diwakili dengan sebuah marker berbentuk bulat yang memiliki *pulse animation* dengan warna sesuai dengan status tiap kebun. Akan tetapi, karena *marker* tersebut merupakan sebuah *custom marker* yang dibuat dengan HTML dan CSS, maka saat skala peta diubah, ukuran seluruh marker yang ada tidak mengalami perubahan.

3.4 Feedback Pengguna Aplikasi

Pada hari acara peluncuran aplikasi, dilakukan survei menggunakan *form* untuk mendapatkan *feedback* dari para pengguna dengan fokus pada para petani dan petugas pemerintahan. Ada 33 responden yang didata, yang terdiri dari 1 perwakilan fakultas pertanian UGM, 12 perwakilan lembaga pemerintahan, dan 20 perwakilan petani dari berbagai kelompok tani, didapatkan hasil yang baik pada tingkat pemahaman penggunaan aplikasi. Berikut adalah list pertanyaan yang diberikan:

- Seberapa mudah Anda memahami cara mendaftar dan login ke aplikasi MySalak?
- Seberapa baik Anda memahami cara menggunakan fitur ramalan cuaca dan hama di aplikasi MySalak?
- Seberapa baik Anda memahami cara menggunakan fitur hitung lalat buah di aplikasi MySalak?
- Seberapa baik Anda memahami cara melaporkan masalah melalui aplikasi MySalak?

Dari keempat pertanyaan tersebut, diberikan skala jawaban dari 1-5, dengan 1 melambangkan "tidak memahami sama sekali", dan 5 melambangkan "sangat memahami". Dari keempat pertanyaan tersebut, didapatkan hasil masing-masing berupa: 3,9; 3,8; 4; dan 3,6; yang menandakan bahwa para responden cukup memahami cara penggunaan aplikasi. Selain itu, para responden juga diminta untuk memberi masukan terhadap aplikasi, yang mayoritas jawabannya adalah permintaan untuk meningkatkan tingkat akurasi dan kecepatan perhitungan lalat. Data yang didapatkan dari survei dapat dilihat pada lampiran H.

3.5 Solusi atas Kendala yang Ditemukan

Untuk mengatasi kendala-kendala tersebut, ditemukan pula solusi yang dapat mengatasi tiap masalah tersebut, yaitu:

- **Menggunakan metode subprocess**

  Metode import yang awalnya digunakan diubah menjadi eksekusi melalui *subprocess*. Dengan menggunakan metode ini, *script* akan dijalankan pada sebuah *subprocess* baru yang dapat dieksekusi seperti saat menggunakan *command line*. Hasil yang didapatkan kemudian akan dikembalikan ke *process* utama, dan kemudian di-*return* sebagai *response* dari server.

- **Mengubah nama setiap foto menggunakan UUID sebelum diproses lebih lanjut**

  Setelah mengidentifikasi penyebab utama dari masalah yang disebabkan oleh kesamaan nama pada gambar, digunakan *library* UUID untuk menamai setiap gambar yang diterima secara acak. Masing-masing

gambar akan diproses setelahnya, dan kemudian hasilnya di-*return* sesuai dengan *request* masing-masing.

- **Menambahkan *image compressor* pada setiap foto profil dan foto *banner* yang diunggah**

  Untuk memastikan seluruh gambar yang digunakan sebagai foto profil dan foto *banner* dapat tersimpan pada *database*, ditambahkan *image compressor* yang digunakan sebelum *payload* dikirim ke *server*. Pengaturan yang digunakan meliputi ukuran maksimal 0.5MB, dengan resolusi maksimal 1080 pixel.

- **Mengubah *custom marker* menjadi visualisasi setiap kelompok tani dalam bentuk blok**

  Untuk memastikan konsistennya skala dari tiap kelompok tani yang ada pada peta, digunakan leaflet-polygon yang merupakan fitur dari *library* peta yang digunakan, yaitu Leaflet. *Custom marker* yang awalnya digunakan diganti dengan sebuah poligon yang mewakili tiap kelompok tani, yang dibuat dengan cara menentukan koordinat tiap titik batasan kebun. Karena merupakan fitur bawaan, maka skala dari tiap poligon juga mengikuti skala peta secara otomatis.