

Nama: Fikri Yoma Rosyidan

NRP: 6026232019

Tugas-UAS-[Individu]- Pengembangan dan Penerapan Sistem

No	Deskripsi	Link
1	Video tutorial di Youtube	https://youtu.be/_sLm2abGYuE
2	Source code di Github	Frontend: https://github.com/fikriyoma01/lost-found-app Backend: https://github.com/fikriyoma01/lost-found-api

Soal nomor 0:

Jelaskan overview terkait proyek secara keseluruhan dan bagian proyek yang menjadi bagian tanggung jawab anda. Jelaskan bagian yang anda kerjakan. Penjelasan antara lain terkait latar belakang adanya proyek ini, permasalahan yang perlu diselesaikan pada proyek ini, tujuan, dsb. Hasil pekerjaan atau tugas sebelumnya, dapat dilampirkan pada bagian akhir laporan ini.

Jawaban nomor 0:

1. Jelaskan overview terkait proyek secara keseluruhan

Proyek Layanan Pendataan Barang dan Kendaraan Hilang di Lingkungan Kampus Institut Teknologi Sepuluh Nopember (ITS) Berbasis Website adalah sebuah inisiatif yang bertujuan untuk meningkatkan efisiensi dalam penanganan kasus-kasus kehilangan barang dan kendaraan di lingkungan kampus. Dengan memanfaatkan teknologi website, proyek ini menyediakan platform yang mudah diakses bagi seluruh anggota kampus untuk melaporkan kehilangan, mencari informasi mengenai barang atau kendaraan yang hilang, serta memudahkan proses penemuan dan pengembalian.

a. Latar Belakang

Institut Teknologi Sepuluh Nopember (ITS) merupakan salah satu perguruan tinggi terkemuka di Indonesia dengan lingkungan kampus yang luas dan beragam aktivitas mahasiswa. Dalam lingkungan yang dinamis seperti ini, kejadian kehilangan barang dan kendaraan sering terjadi, baik itu karena kelalaian, kecurian, atau kesalahpahaman.

Sebelumnya, proses penanganan kasus kehilangan ini dilakukan secara manual melalui laporan ke pihak keamanan kampus atau melalui media sosial, yang sering kali tidak efektif dan memakan waktu. Oleh karena itu, dibutuhkan sebuah solusi yang lebih efisien dan terstruktur untuk menangani masalah ini. Proyek Layanan Pendataan Barang dan Kendaraan Hilang berbasis website hadir untuk mengisi kebutuhan tersebut dengan menyediakan platform digital yang terintegrasi dan mudah diakses oleh seluruh civitas akademika ITS.

b. Permasalahan

- **Proses Manual dan Tidak Terstruktur:** Penanganan kasus kehilangan barang dan kendaraan sering dilakukan secara manual, yang mengakibatkan proses yang lambat dan kurang efektif.
- **Kurangnya Akses Informasi yang Terpusat:** Tidak adanya platform terpusat untuk melaporkan kehilangan dan mencari informasi barang atau kendaraan yang hilang membuat civitas akademika kesulitan dalam menemukan barang atau kendaraan mereka yang hilang.
- **Komunikasi yang Kurang Efektif:** Informasi mengenai barang atau kendaraan yang hilang sering kali tersebar di berbagai media, sehingga menyulitkan pihak yang menemukan atau kehilangan untuk berkomunikasi dengan cepat dan tepat.
- **Minimnya Dokumentasi dan Statistik:** Kurangnya dokumentasi yang terstruktur membuat sulit untuk menganalisis pola kehilangan dan mengidentifikasi area yang rawan.

c. Tujuan

- **Meningkatkan Efisiensi Penanganan Kasus Kehilangan:** Dengan adanya platform berbasis website, proses pelaporan dan penanganan kasus kehilangan dapat dilakukan dengan lebih cepat dan terstruktur.
- **Menyediakan Akses Informasi yang Terpusat:** Website ini akan menjadi pusat informasi yang dapat diakses oleh seluruh anggota kampus untuk melaporkan dan mencari barang atau kendaraan yang hilang.
- **Mempermudah Komunikasi:** Memfasilitasi komunikasi antara pihak yang menemukan barang atau kendaraan dengan pihak yang kehilangan, sehingga proses pengembalian dapat dilakukan dengan lebih cepat dan efisien.

- **Membangun Sistem Dokumentasi yang Baik:** Menciptakan basis data yang terstruktur mengenai kasus-kasus kehilangan di kampus, yang dapat digunakan untuk analisis lebih lanjut dan pengambilan keputusan dalam meningkatkan keamanan kampus.
- **Meningkatkan Kesadaran dan Partisipasi:** Meningkatkan kesadaran civitas akademika mengenai pentingnya menjaga barang dan kendaraan pribadi, serta mendorong partisipasi aktif dalam melaporkan dan mencari barang atau kendaraan yang hilang.

2. Jelaskan bagian proyek yang menjadi bagian tanggung jawab anda

3. Notifikasi dan Pembaruan (Fikri Yoma Rosyidan - 6026232019)

Use case name		Notifikasi dan Pembaruan	
Actors		Mahasiswa, Staf & Dosen, Pengguna Umum	
Pre-condition		Pengguna telah mengirimkan laporan kehilangan atau melaporkan temuan barang	
Post-condition		Pengguna menerima notifikasi tentang pembaruan terkait laporan mereka atau kemungkinan kecocokan	
Primary flows			
Actor Actions		System Responses	
		1.Sistem mengidentifikasi kemungkinan kecocokan antara laporan kehilangan dan laporan temuan barang	
		2.Sistem mengirim notifikasi kepada pengguna yang mengirim laporan kehilangan	
3.Pengguna meninjau notifikasi dan memutuskan apakah akan mengklaim barang temuan tersebut			
Alternate path	A1	Notifikasi atau pembaruan yang muncul diabaikan	
Actor Actions		System Responses	
1.Pengguna memilih untuk mengabaikan notifikasi			
		2.Tidak ada tindakan lanjutan yang diambil	
Exception path	E1	Informasi kontak pengguna tidak benar atau sudah kadaluarsa	
Actor Actions		System Responses	
		1.Sistem tidak dapat mengirimkan notifikasi	

➤ Notifikasi dan Pembaruan

Tanggung jawab utama dalam bagian ini mencakup pengembangan dan pengelolaan sistem notifikasi dan pembaruan yang efisien dan efektif. Berikut adalah rincian tanggung jawab:

- **Identifikasi Kecocokan:**
 - Mengembangkan algoritma yang dapat mengidentifikasi kemungkinan kecocokan antara laporan kehilangan dan temuan barang secara otomatis.

- **Pengiriman Notifikasi:**

- Mengimplementasikan sistem untuk mengirim notifikasi kepada pengguna yang mengirimkan laporan kehilangan saat ada kecocokan yang teridentifikasi.
- Memastikan notifikasi dikirim dengan tepat dan dapat diterima oleh pengguna.

- **Pengelolaan Respons Pengguna:**

- Mengelola interaksi pengguna dengan notifikasi, termasuk fitur untuk pengguna meninjau dan memutuskan apakah akan mengklaim barang temuan.

- **Penanganan Kasus Khusus:**

- Menangani skenario di mana notifikasi atau pembaruan diabaikan oleh pengguna (Alternate Path A1).
- Menangani kesalahan atau informasi kontak yang tidak valid sehingga notifikasi tidak dapat dikirim (Exception Path E1).

7. Repair (Fikri Yoma Rosyidan - 6026232019)

Use case name	Repair	
Actors	Tim Pengembang / Developer	
Pre-condition	Kerusakan atau kesalahan sistem telah diidentifikasi.	
Post-condition	Kerusakan atau kesalahan diperbaiki, dan sistem dipulihkan ke operasi normal.	
Primary flows	Use case menggambarkan langkah-langkah yang dilakukan tim pengembang saat ingin melakukan pencarian barang atau kendaraan yang hilang dengan mengakses pencarian barang hilang	
Actor Actions		System Responses
1.Developer mendiagnosis masalah dan mengidentifikasi penyebab kerusakan atau kesalahan.		
2.Developer menerapkan perbaikan atau memperbaiki komponen yang terkena.		
3.Developer menguji sistem untuk memastikan masalah telah teratasi..		
4.Developer mendokumentasikan proses perbaikan dan memperbarui log sistem.		
Alternate path	A1	
Actor Actions		System Responses
Exception path	E1	Perbaikan memerlukan keahlian khusus atau bantuan eksternal:
Actor Actions		System Responses
Developer eskalasi masalah kepada pihak yang tepat atau vendor.		

➤ **Repair**

Bagian ini bertanggung jawab atas pemeliharaan dan perbaikan sistem ketika terjadi kerusakan atau kesalahan. Berikut adalah rincian tanggung jawab:

- **Diagnosa Masalah:**
 - Mengidentifikasi dan mendiagnosa penyebab kerusakan atau kesalahan pada sistem.
- **Perbaikan Sistem:**
 - Menerapkan perbaikan atau memperbaiki komponen yang terkena dampak.
 - Menguji sistem untuk memastikan masalah telah teratasi dan sistem berfungsi dengan baik.
- **Dokumentasi:**
 - Mendokumentasikan proses perbaikan dan memperbarui log sistem agar setiap tindakan perbaikan tercatat dengan baik.
- **Penanganan Kasus Khusus:**
 - Mengelola skenario di mana perbaikan memerlukan keahlian khusus atau bantuan eksternal (Exception Path E1).
 - Eskalasi masalah kepada pihak yang tepat atau vendor jika diperlukan.

3. Jelaskan bagian yang anda kerjakan

- Authentication Service

Bagian yang dikerjakan:

1. User Registration:

Implementasi halaman registrasi untuk pengguna baru dengan validasi input yang kuat. Pengguna dapat membuat akun dengan memasukkan informasi seperti nama, email, nomor telepon, dan kata sandi. Sistem menggunakan enkripsi untuk menyimpan kata sandi dengan aman di database.

2. Login:

Pengembangan fitur login yang memungkinkan pengguna yang terdaftar untuk masuk ke dalam sistem menggunakan email dan kata sandi. Sistem memverifikasi kredensial pengguna dan mengelola sesi autentikasi.

- Reporting Service

Bagian yang dikerjakan:

1. Report Form:

Pembuatan formulir pelaporan kehilangan yang mudah digunakan untuk barang dan kendaraan. Formulir ini mencakup input untuk detail seperti deskripsi barang/kendaraan, lokasi terakhir terlihat, tanggal kehilangan, dan kontak pelapor.

2. Database Integration:

Integrasi dengan database untuk menyimpan data laporan yang di-submit oleh pengguna. Setiap laporan diberi ID unik untuk memudahkan pelacakan dan pengelolaan.

3. Notification System:

Pengembangan sistem notifikasi yang mengirimkan pesan ke pelapor ketika laporan mereka diterima atau ketika ada pembaruan terkait laporan tersebut.

- Search Service

Bagian yang dikerjakan:

1. Search Interface:

Pembuatan antarmuka pencarian yang intuitif bagi pengguna untuk mencari laporan kehilangan berdasarkan kata kunci, kategori (barang atau kendaraan), dan tanggal laporan.

2. Filter and Sort:

Implementasi fitur filter dan sort untuk memudahkan pengguna dalam menyaring hasil pencarian berdasarkan kriteria tertentu seperti lokasi terakhir terlihat, jenis barang/kendaraan, dan status laporan (hilang/ditemukan).

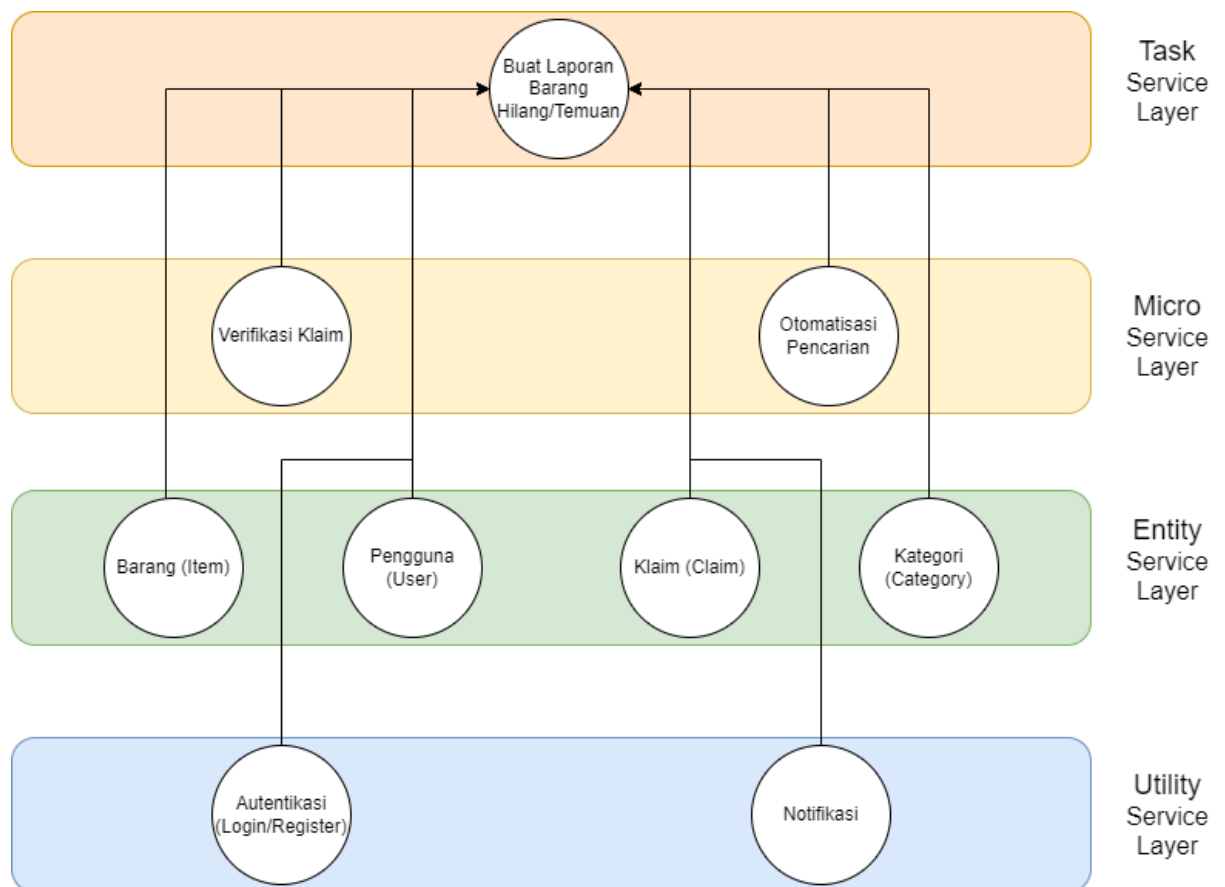
Soal nomor 1:

(a) Tunjukkan dan jelaskan arsitektur final dari SOA yang anda rancang. (b) Tunjukkan dalam bentuk tabel servis yang akan dibuat dan bagaimana mengakses servis yang disediakan tersebut. (c) Tunjukkan bagaimana cara kerja atau bagaimana anda meng-orkestrasi servis-servis tersebut menjadi salah satu layanan use case. (d) Tunjukkan dan jelaskan disertai dengan source code bagaimana implementasi salah satu use case dalam memanfaatkan servis yang disediakan.

Jawaban nomor 1:

(a) Tunjukkan dan jelaskan arsitektur final dari SOA yang anda rancang.

Berikut merupakan arsitektur final dari SOA yang sudah saya rancang sebelumnya.



(b) Tunjukkan dalam bentuk tabel servis yang akan dibuat dan bagaimana mengakses servis yang disediakan tersebut.

Servis yang akan dibuat yaitu Buat Laporan Barang Hilang/Temuan, dari servis ini Task Service Layer akan mengatur eksekusi berbagai tugas utama, seperti membuat laporan baru, Task Service Layer akan memanggil Micro Service Layer untuk memproses tugas lebih lanjut.

Micro Service Layer akan menangani tugas spesifik, seperti verifikasi klaim atau otomatisasi pencarian, Servis mikro ini akan mengakses Entity Service Layer untuk mendapatkan data yang diperlukan.

Entity Service Layer akan menyimpan dan mengelola data inti seperti barang, pengguna, klaim, dan kategori, Setiap entitas memiliki servis tersendiri yang bisa diakses oleh lapisan di atasnya.

Utility Service Layer akan menyediakan fungsi tambahan seperti autentikasi pengguna dan pengiriman notifikasi, Layanan ini mendukung seluruh lapisan untuk memastikan operasi berjalan dengan lancar.

Sementara itu, berikut merupakan penjelasan lebih lanjut mengenai bagaimana pengguna mengakses servis yang disediakan:

- Akses layanan:

Layanan dapat diakses melalui HTTP request dengan alamat `http://localhost:3000`

- Entity Resource dan Endpoint:

Table 1. Entity Resource

Entity	Resource
Barang (Item)	api/items
Pengguna (User)	api/users
Klaim (Claim)	api/claims
Kategori (Category)	api/categorys

Table 2. Endpoint Backend

Method		Path
Barang (Item)		
GET	api/items	
	api/items/:id	
	api/items/search	

	api/items/match
	api/items/:id/similarity
POST	api/items
	api/items/:id/claim
PUT	api/items/:id
DELETE	api/items/:id
Pengguna (User)	
GET	api/users
	api/users/:id
POST	api/users
PUT	api/users/:id
DELETE	api/users/:id
Klaim (Claim)	
PUT	api/claims/:id/verify
Kategori (Category)	
GET	api/categories
POST	api/categories
DELETE	api/categories/:id

(c) Tunjukkan bagaimana cara kerja atau bagaimana anda meng-orkestrasi servis-servis tersebut menjadi salah satu layanan use case.

use case: "Membuat Laporan Barang Hilang". Berikut adalah langkah-langkah dan interaksi antar layanan untuk mewujudkan use case ini:

Use Case: Membuat Laporan Barang Hilang

Pengguna Membuat Laporan:

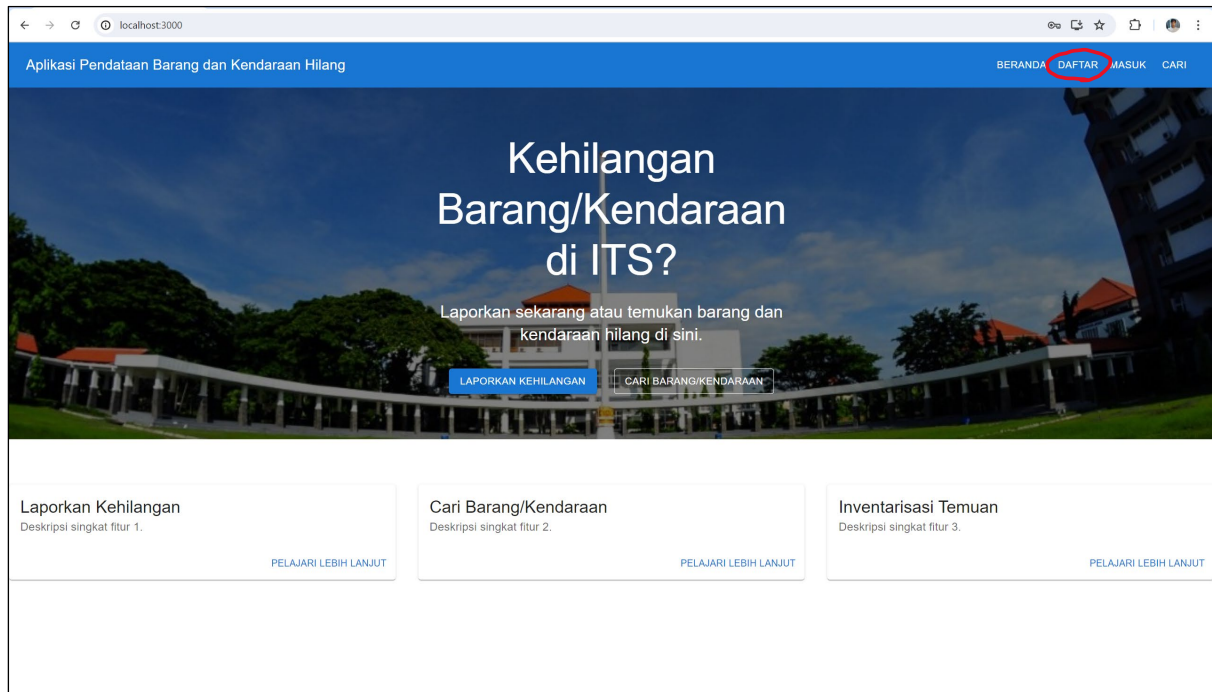
- **Aktor:** User
- **Deskripsi:** User ingin melaporkan barang yang hilang dengan mengisi formulir di aplikasi web.

Alur Proses:

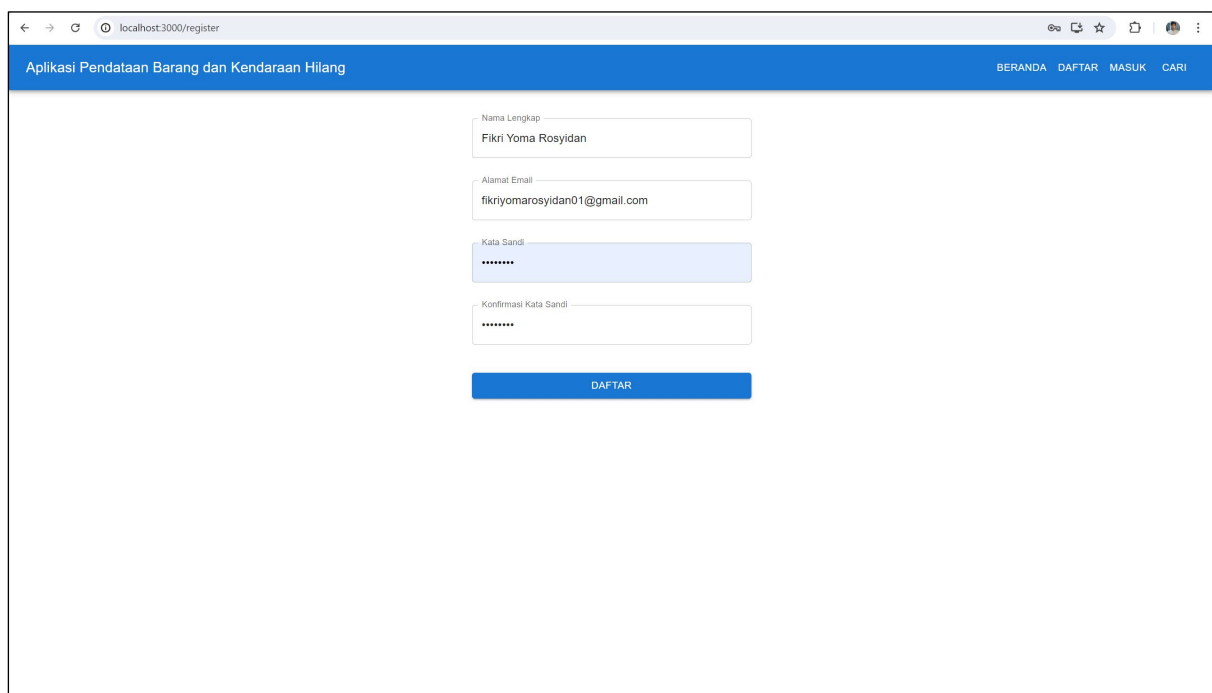
1. User Autentikasi (Login/Register)

- User harus masuk atau mendaftar terlebih dahulu.
- **Servis yang terlibat:**
 - **Utility Service Layer - Autentikasi**
 - User memasukkan kredensial.
 - Sistem memvalidasi kredensial dan mengotentikasi user.

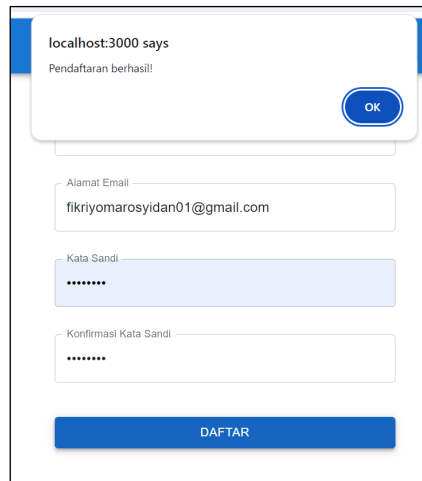
Ketika user ingin melaporkan barang hilang, maka user membuka url website <http://localhost:3000/>. Jika user tersebut belum memiliki akun maka klik “Daftar” pada pojok kanan header halaman.



Ketika user telah melakukan klik “DAFTAR” maka user akan di navigasikan ke halaman registrasi pada url <http://localhost:3000/register>. Selanjutnya user perlu mengisi Nama Lengkap, Alamat Email, dan Kata Sandi. Jika sudah maka user klik “DAFTAR”

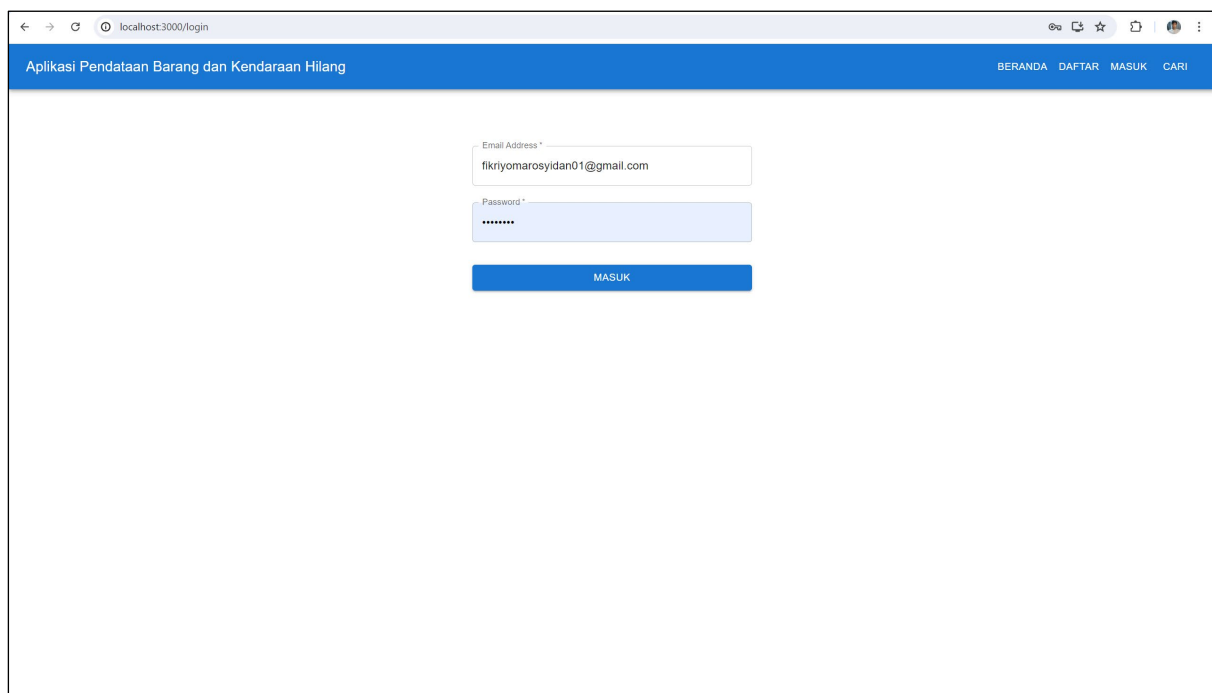


Ketika user sudah mengisi semua data dan klik “DAFTAR” maka user memperoleh notifikasi peringatan “Pendaftaran berhasil” dan selanjutnya diarahkan ke halaman Login



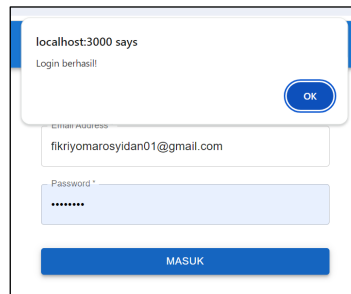
A screenshot of a web application showing a registration success notification and a registration form. The notification is a white box with a blue border, containing the text "localhost:3000 says" and "Pendaftaran berhasil!" with an "OK" button. Below the notification is a registration form with fields for "Alamat Email" (filled with "fikriyomarosyidan01@gmail.com"), "Kata Sandi" (masked with dots), and "Konfirmasi Kata Sandi" (masked with dots). A blue "DAFTAR" button is at the bottom.

Di halaman Login user mengisi Alamat Email dan Password sesuai yang telah diisikan pada saat registrasi. Selanjutnya klik “MASUK”



A screenshot of the login page of a web application. The browser address bar shows "localhost:3000/login". The page has a blue header with the text "Aplikasi Pendataan Barang dan Kendaraan Hilang" and navigation links "BERANDA", "DAFTAR", "MASUK", and "CARI". The main content area contains a login form with fields for "Email Address*" (filled with "fikriyomarosyidan01@gmail.com") and "Password*" (masked with dots). A blue "MASUK" button is at the bottom.

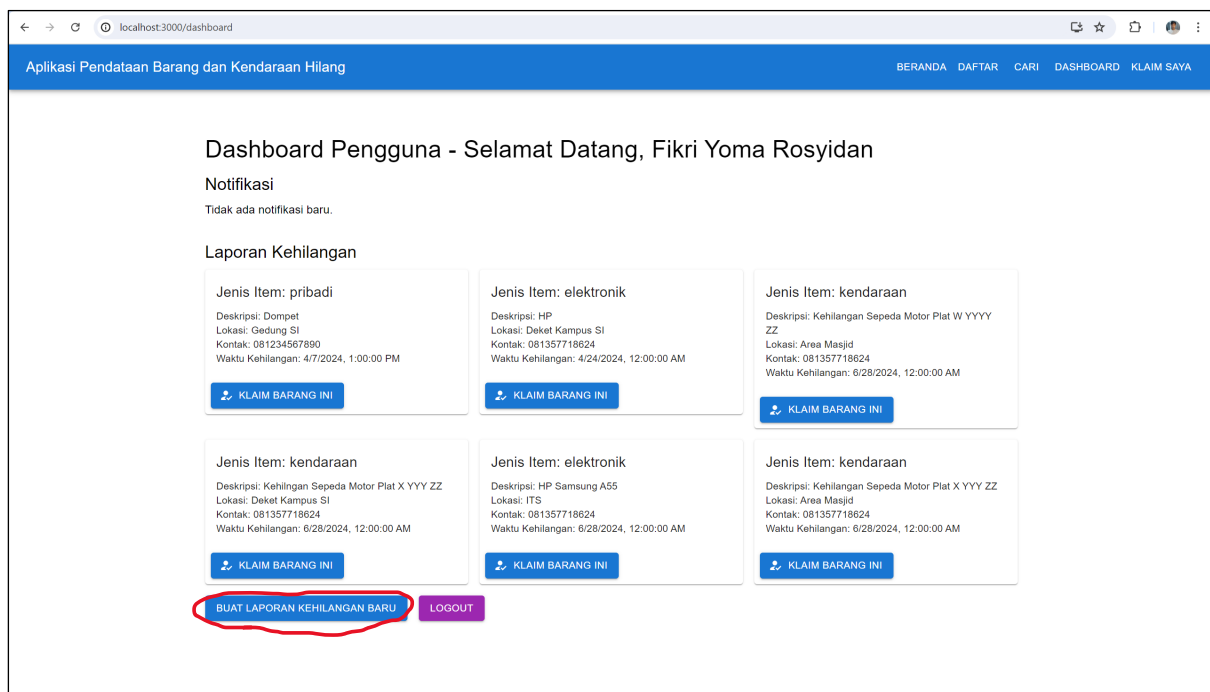
Ketika Alamat Email dan Password benar maka muncul notifikasi peringatan “Login berhasil”



2. User Mengisi Formulir Laporan

- User mengisi formulir dengan rincian barang hilang.
- **Servis yang terlibat:**
 - **Task Service Layer - Buat Laporan Barang Hilang/Temuan**

Ketika user sudah melakukan Login, maka selanjutnya user akan diarahkan ke Dashboard Pengguna yang berisi Notifikasi dan Laporan Kehilangan dari semua pengguna yang telah membuat laporan kehilangan. Untuk membuat laporan kehilangan maka user melakukan klik tombol “BUAT LAPORAN KEHILANGAN BARU”



← → ↻ 📄 ⭐ 📄 👤 ⋮

localhost:3000/dashboard

Aplikasi Pendataan Barang dan Kendaraan Hilang

BERANDA DAFTAR CARİ DASHBOARD KLAIM SAYA

Dashboard Pengguna - Selamat Datang, Fikri Yoma Rosyidan

Notifikasi
Tidak ada notifikasi baru.

Laporan Kehilangan

Jenis Item: pribadi

Deskripsi: Dompet
Lokasi: Gedung SI
Kontak: 081234567890
Waktu Kehilangan: 4/7/2024, 1:00:00 PM

[KLAIM BARANG INI](#)

Jenis Item: elektronik

Deskripsi: HP
Lokasi: Deket Kampus SI
Kontak: 081357718624
Waktu Kehilangan: 4/24/2024, 12:00:00 AM

[KLAIM BARANG INI](#)

Jenis Item: kendaraan

Deskripsi: Kehilangan Sepeda Motor Plat W YYYY ZZ
Lokasi: Area Masjid
Kontak: 081357718624
Waktu Kehilangan: 6/28/2024, 12:00:00 AM

[KLAIM BARANG INI](#)

Jenis Item: kendaraan

Deskripsi: Kehilangan Sepeda Motor Plat X YYY ZZ
Lokasi: Deket Kampus SI
Kontak: 081357718624
Waktu Kehilangan: 6/28/2024, 12:00:00 AM

[KLAIM BARANG INI](#)

Jenis Item: elektronik

Deskripsi: HP Samsung A55
Lokasi: ITS
Kontak: 081357718624
Waktu Kehilangan: 6/28/2024, 12:00:00 AM

[KLAIM BARANG INI](#)

Jenis Item: kendaraan

Deskripsi: Kehilangan Sepeda Motor Plat X YYY ZZ
Lokasi: Area Masjid
Kontak: 081357718624
Waktu Kehilangan: 6/28/2024, 12:00:00 AM

[KLAIM BARANG INI](#)

[BUAT LAPORAN KEHILANGAN BARU](#) [LOGOUT](#)

User selanjutnya akan diarahkan ke halaman “Laporkan Kehilangan” di url <http://localhost:3000/report> di sini user perlu mengisi formulir yang terdiri dari komponen: Jenis Barang/Kendaraan, Deskripsi, Lokasi Kehilangan, Waktu Kehilangan, dan Informasi Kontak

The screenshot shows a web browser at the URL `localhost:3000/report`. The page title is "Aplikasi Pendataan Barang dan Kendaraan Hilang". The navigation bar includes links: BERANDA, DAFTAR, CARI, DASHBOARD, and KLAIM SAYA. The main heading is "Laporkan Kehilangan". The form contains the following fields: a dropdown menu for "Jenis Barang/Kendaraan", a text input for "Deskripsi", a text input for "Lokasi Kehilangan", a date and time picker for "Waktu Kehilangan", and a text input for "Informasi Kontak". A blue button labeled "LAPORKAN KEHILANGAN" is at the bottom.

Ketika user sudah mengisi semua komponen formulir, maka user selanjutnya klik “LAPORKAN KEHILANGAN”

This block contains three screenshots illustrating the form completion process. The first screenshot on the left shows the "Jenis Barang/Kendaraan" dropdown menu open, with options: Elektronik, Kendaraan, Barang Pribadi, and Lainnya. The middle screenshot shows the "Waktu Kehilangan" date and time picker, displaying a calendar for June 2024 and a time selection interface. The third screenshot on the right shows the completed form with the following values: "Jenis Barang/Kendaraan" set to "Elektronik", "Deskripsi" as "Smartphone Samsung A52s 5G Warna Hitam", "Lokasi Kehilangan" as "Departemen SI Ruang Kelas 201", "Waktu Kehilangan" as "06/28/2024 12:00 AM", and "Informasi Kontak" as "Hubungi Fikri Yoma Rosyidan, Mahasiswa S2 SI". The "LAPORKAN KEHILANGAN" button is visible at the bottom.

3. Validasi dan Penyimpanan Laporan

- Sistem memvalidasi data yang diisi oleh user.
- **Servis yang terlibat:**
 - **Entity Service Layer - Barang**
 - **Entity Service Layer - Pengguna**
 - Laporan barang hilang disimpan di database entitas barang.
 - Data pengguna yang melaporkan juga di-update jika diperlukan.

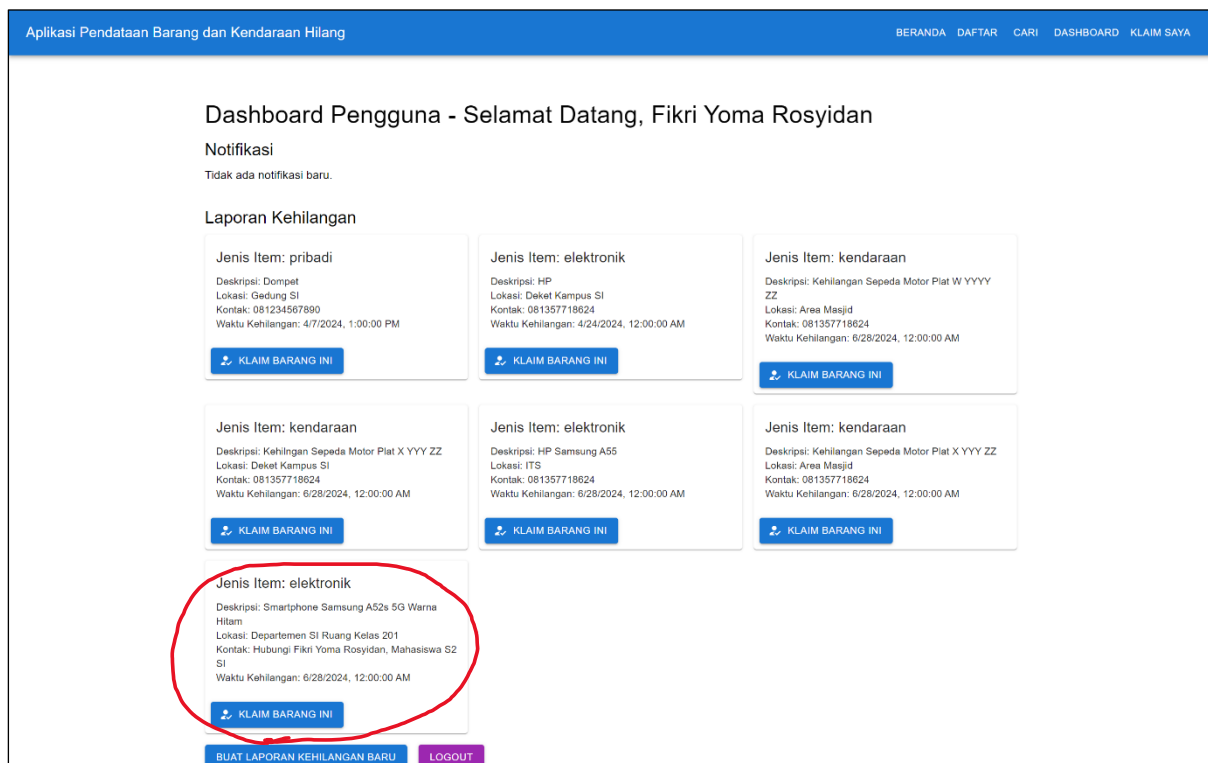
Ketika user sudah klik “LAPORKAN KEHILANGAN” maka akan muncul notifikasi peringatan “Laporan berhasil dikirim” yang tandanya laporan kehilangan berhasil masuk ke dalam database aplikasi

The screenshot shows a web browser window at the URL `localhost:3000/report`. The page has a blue header with the title "Aplikasi Pendataan Barang dan Kendaraan Hilang" on the left and navigation links "BERANDA", "DAFTAR", "CARI", "DASHBOARD", and "KLAIM SAYA" on the right. A notification box at the top center says "localhost:3000 says Laporan berhasil dikirim!" with an "OK" button. The main form is titled "Lap" and contains several input fields: "Jenis Barang/Kendaraan" (a dropdown menu with "Elektronik" selected), "Deskripsi" (a text area containing "Smartphone Samsung A52s 5G Warna Hitam"), "Lokasi Kehilangan" (a text field containing "Departemen SI Ruang Kelas 201"), "Waktu Kehilangan" (a date and time picker showing "06/28/2024 12:00 AM"), and "Informasi Kontak" (a text field containing "Hubungi Fikri Yoma Rosyidan, Mahasiswa S2 SI"). At the bottom of the form is a grey button labeled "MENGIRIM...".

4. Update Dashboard

- Setelah laporan berhasil disimpan, sistem memperbarui data barang hilang yang baru dan ditampilkan dalam dashboard.
- **Servis yang terlibat:**
 - **Utility Service Layer - Barang**
 - Data barang hilang diupdate dengan mengambil lagi data di database entitas barang.

Selanjutnya user akan dinavigasikan ke halaman dashboard pengguna untuk diperlihatkan item yang bertambah di sesi Laporan Kehilangan



(d) Tunjukkan dan jelaskan disertai dengan source code bagaimana implementasi salah satu use case dalam memanfaatkan servis yang disediakan.

Pada use case: "Membuat Laporan Barang Hilang" terdapat banyak code yang digunakan baik di sisi frontend maupun backend. Berikut merupakan detail code yang digunakan:

- User Autentikasi (Login/Register)

Code untuk mengirim registrasi ke server di sisi frontend

```
onSubmit={ (values, { setSubmitting, resetForm, setErrors }) => {  
  const { confirmPassword, ...dataToSend } = values;  
  fetch('/users/register', {  
    method: 'POST',  
    headers: {  
      'Content-Type': 'application/json',  
    },  
    body: JSON.stringify(dataToSend),  
  })  
}
```

```

    })
    .then(response => {
      if (!response.ok) {
        throw new Error(`HTTP error status:
${response.status}`);
      }
      return response.json();
    })
    .then(data => {
      if (data.success) {
        alert('Pendaftaran berhasil!');
        resetForm();
      } else {
        setErrors({ server: data.message });
        alert(`Pendaftaran gagal: ${data.message}`);
      }
    })
    .catch((error) => {
      console.error('Error:', error);
      alert('Gagal mendaftar. Silakan coba lagi.');
    })
    .finally(() => setSubmitting(false));
  }}

```

ReactJS halaman registrasi

```

import React from 'react';
import { Formik, Form, Field } from 'formik';
import { Button, TextField, Container } from '@mui/material';
import * as Yup from 'yup';

const RegistrationSchema = Yup.object().shape({
  name: Yup.string().required('Nama lengkap diperlukan'),
  email: Yup.string().email('Email tidak valid').required('Email
diperlukan'),
  password: Yup.string().min(8, 'Kata sandi terlalu pendek - harus 8
karakter atau lebih').required('Kata sandi diperlukan'),
  confirmPassword: Yup.string().oneOf([Yup.ref('password'), null],
'Kata sandi tidak cocok').required('Konfirmasi kata sandi
diperlukan'),
});

```



```

export default function RegistrationPage() {
  return (
    <Container component="main" maxWidth="xs" sx={{ mt: 3 }}>
      <Formik
        initialValues={{
          name: '',
          email: '',
          password: '',
          confirmPassword: '',
        }}
        validationSchema={RegistrationSchema}
        onSubmit={
=> {
          const { confirmPassword, ...dataToSend } = values;
          fetch('/users/register', {
            method: 'POST',
            headers: {
              'Content-Type': 'application/json',
            },
            body: JSON.stringify(dataToSend),
          })
            .then(response => {
              if (!response.ok) {
                throw new Error(`HTTP error status:
${response.status}`);
              }
              return response.json();
            })
            .then(data => {
              if (data.success) {
                alert('Pendaftaran berhasil!');
                resetForm();
              } else {
                setErrors({ server: data.message });
                alert(`Pendaftaran gagal: ${data.message}`);
              }
            })
            .catch((error) => {
              console.error('Error:', error);
              alert('Gagal mendaftar. Silakan coba lagi.');
            })
        }
      </Formik>
    </Container>
  );
}

```

```

        .finally(() => setSubmitting(false));
    }}
>
    ({ errors, touched, handleChange, handleBlur }) => (
      <Form>
        <Field as={TextField}
          variant="outlined"
          margin="normal"
          fullWidth
          id="name"
          label="Nama Lengkap"
          name="name"
          autoComplete="name"
          autoFocus
          helperText={touched.name && errors.name}
          error={touched.name && Boolean(errors.name)}
          onChange={handleChange}
          onBlur={handleBlur}
          sx={{ mb: 2 }}
        />
        <Field as={TextField}
          variant="outlined"
          margin="normal"
          fullWidth
          id="email"
          label="Alamat Email"
          name="email"
          autoComplete="email"
          helperText={touched.email && errors.email}
          error={touched.email && Boolean(errors.email)}
          onChange={handleChange}
          onBlur={handleBlur}
          sx={{ mb: 2 }}
        />
        <Field as={TextField}
          variant="outlined"
          margin="normal"
          fullWidth
          name="password"
          label="Kata Sandi"
          type="password"
          id="password"

```

```

        autoComplete="current-password"
        helperText={touched.password && errors.password}
        error={touched.password && Boolean(errors.password)}
        onChange={handleChange}
        onBlur={handleBlur}
        sx={{ mb: 2 }}
      />
      <Field as={TextField}
        variant="outlined"
        margin="normal"
        fullWidth
        name="confirmPassword"
        label="Konfirmasi Kata Sandi"
        type="password"
        id="confirmPassword"
        helperText={touched.confirmPassword &&
errors.confirmPassword}
        error={touched.confirmPassword &&
Boolean(errors.confirmPassword)}
        onChange={handleChange}
        onBlur={handleBlur}
        sx={{ mb: 2 }}
      />
      <Button
        type="submit"
        fullWidth
        variant="contained"
        color="primary"
        sx={{ mt: 3, mb: 2 }}
      >
        Daftar
      </Button>
    </Form>
  )}
</Formik>
</Container>
);
}

```

Code tersebut menampilkan halaman registrasi pengguna dalam aplikasi React menggunakan Formik dan Yup untuk manajemen formulir dan MUI (Material-UI) untuk komponen antarmuka pengguna. Code ini membangun halaman registrasi yang interaktif dengan validasi dan penanganan error.

Backend ExpressJS atau sisi server untuk Registrasi

```
const express = require('express');
const router = express.Router();
const bcrypt = require('bcryptjs');
const passport = require('passport');
const jwt = require('jsonwebtoken');
const User = require('../models/UserModel');

// Registrasi Pengguna
router.post('/register', async (req, res) => {
  try {
    let { name, email, password } = req.body;

    // Periksa apakah user sudah terdaftar
    let user = await User.findOne({ email: email });
    if (user) {
      return res.status(400).json({ message: "User sudah terdaftar"
});
    }

    // Enkripsi password
    const salt = await bcrypt.genSalt(10);
    password = await bcrypt.hash(password, salt);

    // Buat user baru
    user = new User({
      name,
      email,
      password
    });

    // Simpan user
    await user.save();
```

```

    res.status(201).json({ success: true, message: "User berhasil
didaftarkan" });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: "Server error" });
  }
});

```

Code di atas merupakan bagian dari aplikasi server yang dibuat menggunakan `express`, sebuah kerangka kerja untuk Node.js. Kode ini menangani proses pendaftaran pengguna baru melalui rute `/register`. Pertama, data yang dikirim oleh pengguna melalui permintaan (request) diambil, termasuk nama, email, dan kata sandi. Sistem kemudian memeriksa apakah ada pengguna yang sudah terdaftar dengan email yang sama di database menggunakan model `User`. Jika pengguna dengan email tersebut sudah ada, sistem mengirimkan balasan bahwa pengguna sudah terdaftar, dengan status kode 400 yang menunjukkan permintaan tidak valid. Jika tidak ada pengguna dengan email yang sama, kata sandi yang diberikan akan dienkripsi menggunakan `bcrypt` untuk memastikan keamanannya. Setelah itu, data pengguna baru dibuat dan disimpan di database. Jika proses ini berhasil, sistem mengirimkan balasan bahwa pendaftaran berhasil dengan status kode 201 yang menunjukkan bahwa sumber daya baru telah dibuat. Jika terjadi kesalahan selama proses pendaftaran, seperti kesalahan server atau masalah lain, sistem akan menangkap kesalahan tersebut, mencatatnya, dan mengirim balasan bahwa terjadi kesalahan di server dengan status kode 500. Dengan demikian, kode ini mengatur alur pendaftaran pengguna baru secara lengkap, mulai dari validasi email hingga penyimpanan data yang aman dan penanganan kesalahan.

ReactJS halaman login

```

import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { Container, TextField, Button } from '@mui/material';

```

```

export default function LoginPage() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const navigate = useNavigate();

  const handleSubmit = (event) => {
    event.preventDefault();
    const userCredentials = { email, password };
    console.log('Sending these credentials:', userCredentials);

    fetch('/users/login', {
      method: 'POST',
      headers: {
        'Accept': 'application/json',
        'Content-Type': 'application/json'
      },
      body: JSON.stringify(userCredentials),
    })
    .then(response => {
      console.log('Response status:', response.status);
      if (!response.ok) {
        response.text().then(text => {
          console.log('Server response text:', text);
          throw new Error('Network response was not ok');
        });
      }
    })
    return response.json();
  })
  .then(data => {
    console.log(data);
    if (data.token && data.user) { // Pastikan ada pengecekan
token yang valid
      localStorage.setItem('token', data.token);
      localStorage.setItem('userInfo', JSON.stringify(data.user));
// Pastikan data.user ada dan berisi data yang benar
      alert('Login berhasil!');
      navigate('/dashboard');
    } else {
      // Handle jika respons tidak mengandung token
      alert('Login status: ' + (data.message || 'Tidak ada
token'));
    }
  })
}

```

```

    })
    .catch((error) => {
        // Tangkap error baik dari network atau server-side
        console.error('Login error:', error);
        alert('Login status: ' + error.message);
    });
};

return (
    <Container component="main" maxWidth="xs" sx={{ marginTop: 8 }}>
        <form onSubmit={handleSubmit} style={{ display: 'flex',
flexDirection: 'column', alignItems: 'center' }}>
            <TextField
                variant="outlined"
                margin="normal"
                required
                fullWidth
                id="email"
                label="Email Address"
                name="email"
                autoComplete="email"
                autoFocus
                value={email}
                onChange={(e) => setEmail(e.target.value)}
            />
            <TextField
                variant="outlined"
                margin="normal"
                required
                fullWidth
                name="password"
                label="Password"
                type="password"
                id="password"
                autoComplete="current-password"
                value={password}
                onChange={(e) => setPassword(e.target.value)}
            />
            <Button
                type="submit"
                fullWidth
                variant="contained"

```

```

        color="primary"
        sx={{ mt: 3, mb: 2 }}
      >
        Masuk
      </Button>
    </form>
  </Container>
);
}

```

Code ini membuat halaman login untuk sebuah aplikasi menggunakan React dan Material-UI. Di dalam halaman login ini, ada dua bidang input untuk pengguna memasukkan alamat email dan kata sandi mereka. Ketika pengguna menekan tombol "Masuk", data yang dimasukkan akan dikirim ke server melalui permintaan POST. Sebelum mengirim, data yang diisi pengguna akan dicatat di konsol untuk tujuan debugging. Server kemudian merespons dengan status dan mungkin pesan teks. Jika login berhasil, server akan mengirimkan token dan informasi pengguna, yang akan disimpan di localStorage browser. Pengguna kemudian akan diberi tahu bahwa login berhasil dan diarahkan ke halaman dasbor. Jika login gagal atau ada kesalahan jaringan, pesan kesalahan akan ditampilkan kepada pengguna. Dengan demikian, halaman ini memungkinkan pengguna untuk memasukkan kredensial mereka, mengirimkannya ke server, dan menangani hasil respons untuk mengarahkan pengguna sesuai dengan status login mereka.

Backend ExpressJS atau sisi server untuk Login

```

// Login Handle
router.post('/login', (req, res, next) => {
  console.log('Received login request with body:', req.body);
  passport.authenticate('local', (err, user, info) => {
    if (err) {
      return res.status(500).json({ message: 'Internal server error'
    });
    }
    if (!user) {

```



```

        return res.status(401).json({ message: 'Email atau password
salah' });
    }
    // Menggenerate token
    const token = jwt.sign({ id: user._id }, 'rahasia', {
        expiresIn: 86400 // expires in 24 hours
    });
    req.login(user, (err) => {
        if (err) {
            return res.status(500).json({message: err.message});
        }
        return res.json({
            message: 'Login berhasil',
            user: {
                _id: user._id,
                email: user.email,
                name: user.name
            },
            token: token
        });
    });
})(req, res, next);
});

```

Code ini mengatur proses login untuk aplikasi. Saat pengguna mengirim permintaan login, server mencatat data yang diterima untuk tujuan debugging. Kemudian, server menggunakan mekanisme autentikasi untuk memeriksa apakah email dan kata sandi yang dimasukkan pengguna sesuai dengan data yang ada di database. Jika ada masalah server, pengguna akan diberi tahu bahwa terjadi kesalahan internal. Jika email atau kata sandi salah, pengguna akan mendapatkan pesan bahwa kredensial yang dimasukkan tidak valid. Jika login berhasil, server akan membuat token yang berlaku selama 24 jam. Token ini berfungsi untuk mengidentifikasi pengguna dalam sesi berikutnya tanpa perlu login ulang. Setelah itu, server menyimpan informasi pengguna dan mengirimkan balasan ke pengguna yang berisi pesan sukses, informasi pengguna, dan token. Dengan cara ini, pengguna bisa masuk ke akun mereka dan melanjutkan menggunakan aplikasi dengan sesi yang aman.

- User Mengisi Formulir Laporan

ReactJS halaman formulir laporan

```
import React from 'react';
import { Formik, Form, Field } from 'formik';
import { Button, TextField, Container, MenuItem, Typography } from
 '@mui/material';
import * as Yup from 'yup';
import { DateTimePicker } from '@mui/x-date-pickers/DateTimePicker';
import { AdapterDateFns } from '@mui/x-date-
pickers/AdapterDateFnsV3';
import { LocalizationProvider } from '@mui/x-date-
pickers/LocalizationProvider';
import { useNavigate } from 'react-router-dom';

const itemTypes = [
  { value: 'elektronik', label: 'Elektronik' },
  { value: 'kendaraan', label: 'Kendaraan' },
  { value: 'pribadi', label: 'Barang Pribadi' },
  { value: 'lainnya', label: 'Lainnya' },
];

const ReportLossSchema = Yup.object().shape({
  itemType: Yup.string().required('Jenis barang/kendaraan
diperlukan'),
  description: Yup.string().required('Deskripsi diperlukan'),
  location: Yup.string().required('Lokasi kehilangan diperlukan'),
  timeLost: Yup.date().required('Waktu kehilangan
diperlukan').nullable(),
  contactInfo: Yup.string().required('Informasi kontak diperlukan'),
});

export default function ReportLossPage() {
  const navigate = useNavigate();

  return (
    <Container component="main" maxWidth="sm" sx={{ mt: 3 }}>
      <Typography variant="h4" component="h1" gutterBottom>
        Laporkan Kehilangan
      </Typography>
      <Formik
        initialValues={{
```

```

        itemType: '',
        description: '',
        location: '',
        timeLost: null,
        contactInfo: '',
    }}
    validationSchema={ReportLossSchema}
    onSubmit={values, { setSubmitting, resetForm, setErrors }}
=> {
    const userInfo =
JSON.parse(localStorage.getItem('userInfo'));
    if (!userInfo || !userInfo._id) {
        alert('Anda harus login untuk melaporkan kehilangan.');
```

navigate('/login');

return;

}

```

    const reportData = {
        ...values,
        reportedBy: userInfo._id,
    };

    fetch('/lostitems/add', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer
${localStorage.getItem('token')}`
        },
        body: JSON.stringify(reportData),
    })
    .then(response => {
        if (!response.ok) {
            return response.json().then(err => {
                throw new Error(err.message || 'Terjadi kesalahan
saat mengirim laporan.');
```

});

}

return response.json();

})

.then(data => {

console.log('Success:', data);

```

        alert('Laporan berhasil dikirim!');
        resetForm();
        navigate('/dashboard');
    })
    .catch((error) => {
        console.error('Error:', error);
        setErrors({ submit: error.message });
    })
    .finally(() => {
        setSubmitting(false);
    });
  }}
  >
  ({ errors, touched, handleChange, handleBlur,
setFieldValue, values, isSubmitting }) => (
    <Form>
      <Field as={TextField}>
        select
        variant="outlined"
        margin="normal"
        fullWidth
        id="itemType"
        label="Jenis Barang/Kendaraan"
        name="itemType"
        helperText={touched.itemType && errors.itemType}
        error={touched.itemType && Boolean(errors.itemType)}
        onChange={handleChange}
        onBlur={handleBlur}
        sx={{ mb: 2 }}
      >
        {itemTypes.map((option) => (
          <MenuItem key={option.value} value={option.value}>
            {option.label}
          </MenuItem>
        ))}
      </Field>
      <Field as={TextField}>
        variant="outlined"
        margin="normal"
        fullWidth
        id="description"
        label="Deskripsi"

```

```

        name="description"
        multiline
        rows={4}
        helperText={touched.description && errors.description}
        error={touched.description &&
Boolean(errors.description)}
        onChange={handleChange}
        onBlur={handleBlur}
        sx={{ mb: 2 }}
    />
    <Field as={TextField}
        variant="outlined"
        margin="normal"
        fullWidth
        id="location"
        label="Lokasi Kehilangan"
        name="location"
        helperText={touched.location && errors.location}
        error={touched.location && Boolean(errors.location)}
        onChange={handleChange}
        onBlur={handleBlur}
        sx={{ mb: 2 }}
    />
    <LocalizationProvider dateAdapter={AdapterDateFns}>
    <DateTimePicker
        label="Waktu Kehilangan"
        inputFormat="dd/MM/yyyy HH:mm"
        value={values.timeLost}
        onChange={(value) => setFieldValue('timeLost',
value)}
        renderInput={({params) => <TextField {...params}
fullWidth sx={{ mb: 2 }} />}
    />
    </LocalizationProvider>
    <Field as={TextField}
        variant="outlined"
        margin="normal"
        fullWidth
        id="contactInfo"
        label="Informasi Kontak"
        name="contactInfo"
        helperText={touched.contactInfo && errors.contactInfo}

```

```

        error={touched.contactInfo &&
Boolean(errors.contactInfo)}
        onChange={handleChange}
        onBlur={handleBlur}
        sx={{ mb: 2 }}
      />
      {errors.submit && (
        <Typography color="error" sx={{ mt: 2 }}>
          {errors.submit}
        </Typography>
      )}

      <Button
        type="submit"
        fullWidth
        variant="contained"
        color="primary"
        disabled={isSubmitting}
        sx={{ mt: 3 }}
      >
        {isSubmitting ? 'Mengirim...' : 'Laporkan Kehilangan'}
      </Button>
    </Form>
  )}
</Formik>
</Container>
);
}

```

Code ini adalah sebuah halaman di aplikasi yang memungkinkan pengguna untuk melaporkan barang yang hilang. Ketika halaman ini dibuka, pengguna akan melihat formulir dengan beberapa bidang input seperti jenis barang, deskripsi barang, lokasi kehilangan, waktu kehilangan, dan informasi kontak. Pengguna harus mengisi semua bidang ini untuk melaporkan barang yang hilang. Setelah formulir diisi, pengguna dapat mengirimnya dengan menekan tombol "Laporkan Kehilangan". Sebelum mengirim, sistem akan memeriksa apakah pengguna sudah login. Jika belum login, pengguna akan diarahkan ke halaman login. Jika sudah login, data yang dimasukkan akan dikirim ke

server melalui permintaan POST untuk disimpan di database. Jika pengiriman berhasil, pengguna akan diberitahu bahwa laporan telah berhasil dikirim dan kemudian diarahkan ke halaman dasbor. Jika terjadi kesalahan, pengguna akan mendapatkan pesan kesalahan di layar. Halaman ini juga menampilkan elemen antarmuka pengguna yang rapi dan mudah digunakan, serta menyediakan feedback visual untuk kesalahan input dan status pengiriman.

Backend ExpressJS atau sisi server untuk mengirim laporan

```
const router = require('express').Router();
let LostItem = require('../models/LostItemModel');
const FoundItem = require('../models/FoundItemModel');
const { createNotification } =
require('../utils/notificationUtils');

// Tambahkan laporan kehilangan baru
router.post('/add', async (req, res) => {
  const newLostItem = new LostItem({ ...req.body });

  try {
    const savedItem = await newLostItem.save();

    // Cari item yang ditemukan yang cocok
    const foundItem = await FoundItem.findOne({
      itemType: savedItem.itemType,
      description: { $regex: savedItem.description, $options: 'i' }
    });

    if (foundItem) {
      // Kirim notifikasi ke pengguna yang melaporkan item hilang
      await createNotification(
        savedItem.reportedBy,
        `Item yang cocok dengan laporan Anda mungkin telah
ditemukan: ${foundItem.description}`,
        'item_found',
        foundItem._id,
        'FoundItem'
      );
    }
  }
});
```

```

    }

    res.status(201).json(savedItem);
  } catch (err) {
    res.status(400).json('Error: ' + err);
  }
});

```

Code ini menangani penambahan laporan barang hilang di aplikasi. Saat pengguna mengirimkan laporan barang hilang melalui permintaan POST, data yang dikirimkan akan digunakan untuk membuat entri baru di database. Setelah data tersimpan, sistem akan mencari barang yang ditemukan di database yang memiliki jenis dan deskripsi yang mirip dengan laporan barang hilang tersebut. Jika ditemukan barang yang cocok, sistem akan mengirimkan notifikasi kepada pengguna yang melaporkan barang hilang tersebut, memberi tahu mereka bahwa mungkin ada barang yang ditemukan yang cocok dengan deskripsi barang mereka. Notifikasi ini berisi deskripsi barang yang ditemukan dan mengarahkan pengguna ke detail barang tersebut. Jika proses penyimpanan atau pencarian barang yang cocok mengalami masalah, sistem akan mengirimkan pesan kesalahan kepada pengguna. Dengan cara ini, kode ini tidak hanya menyimpan laporan barang hilang, tetapi juga membantu pengguna menemukan kembali barang mereka dengan mencocokkan laporan dengan barang yang ditemukan.

- Update Dashboard

ReactJS halaman dashboard

```

import React, { useEffect, useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { Container, Typography, Button, Card, CardContent,
CardActions, Grid, Box, CircularProgress } from '@mui/material';
import ClaimIcon from '@mui/icons-material/HowToReg';

export default function UserDashboardPage() {
  const [reports, setReports] = useState([]);
  const [notifications, setNotifications] = useState([]);
  const [isLoading, setIsLoading] = useState(true);
  const [error, setError] = useState(null);
  const navigate = useNavigate();

```



```

const userInfo = JSON.parse(localStorage.getItem('userInfo'));

const handleLogout = () => {
  localStorage.removeItem('token');
  localStorage.removeItem('userInfo');
  navigate('/login');
};

useEffect(() => {
  const fetchData = async () => {
    setIsLoading(true);
    try {
      const token = localStorage.getItem('token');
      console.log('Fetching data with token:', token);

      // Fetch notifications
      const notificationsResponse = await
fetch('/api/notifications', {
  headers: { 'Authorization': `Bearer ${token}` }
});

      if (!notificationsResponse.ok) {
        throw new Error(`HTTP error! status:
${notificationsResponse.status}`);
      }

      const notificationsData = await
notificationsResponse.json();
      console.log('Notifications received:', notificationsData);
      setNotifications(Array.isArray(notificationsData) ?
notificationsData : []);

      // Fetch lost items reports
      const reportsResponse = await fetch('/lostitems', {
        headers: { 'Authorization': `Bearer ${token}` }
      });

      if (!reportsResponse.ok) {
        throw new Error(`HTTP error! status:
${reportsResponse.status}`);
      }
    }
  };
  fetchData();
}, []);

```

```

        const reportsData = await reportsResponse.json();
        console.log('Lost items reports received:', reportsData);
        setReports(Array.isArray(reportsData) ? reportsData : []);

    } catch (err) {
        console.error("Error fetching data:", err);
        setError(err.message);
    } finally {
        setIsLoading(false);
    }
};

fetchData();
}, []);

const claimItem = (itemId) => {
    const token = localStorage.getItem('token');
    if (!token) {
        alert('Anda harus login untuk mengklaim barang.');
```

navigate('/login');

return;

}

```

    fetch(`http://localhost:5000/founditems/claim/${itemId}`, {
        method: 'PUT',
        headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${token}`
        },
        credentials: 'include'
    })
    .then(async response => {
        const contentType = response.headers.get("content-type");
        if (contentType && contentType.indexOf("application/json") !==
-1) {
            return response.json().then(data => {
                if (!response.ok) throw new Error(data.message || 'Failed
to claim item');
                return data;
            });
        } else {
            return response.text().then(text => {

```

```

        if (!response.ok) throw new Error(text || 'Failed to claim
item');
        return { message: text };
    });
}
})
.then(data => {
    alert(data.message || 'Barang berhasil diklaim. Cek email Anda
untuk detail lebih lanjut.');
    navigate('/dashboard');
})
.catch(error => {
    console.error('Error claiming item:', error);
    if (error.message === 'Invalid token' || error.message ===
'Token expired') {
        alert('Sesi Anda telah berakhir. Silakan login kembali.');
```

```

        localStorage.removeItem('token');
        navigate('/login');
    } else {
        alert(`Gagal mengklaim barang: ${error.message}`);
    }
});
};

const markNotificationAsRead = async (notificationId) => {
    try {
        const response = await
fetch(`/api/notifications/${notificationId}`, {
    method: 'PATCH',
    headers: {
        'Authorization': `Bearer
${localStorage.getItem('token')}`,
        'Content-Type': 'application/json'
    }
});
        if (!response.ok) throw new Error('Failed to mark notification
as read');
```

```

        setNotifications(prevNotifications =>
            prevNotifications.map(notif =>
                notif._id === notificationId ? { ...notif, isRead: true }
            : notif

```

```

    )
  );
} catch (error) {
  console.error("Error marking notification as read:", error);
  alert('Failed to mark notification as read. Please try
again.');
```

```

  }
};

if (isLoading) return <Typography>Loading
notifications...</Typography>;
if (error) return <Typography color="error">Error:
{error}</Typography>;

// Render komponen
return (
  <Container sx={{ marginTop: 8 }}>
    <Typography variant="h4" gutterBottom>
      Dashboard Pengguna - Selamat Datang, {userInfo.name}
    </Typography>

    <Box sx={{ mb: 4 }}>
      <Typography variant="h5"
gutterBottom>Notifikasi</Typography>
      {notifications.length > 0 ? (
        notifications.map((notification, index) => (
          <Card key={index} sx={{ mb: 2, backgroundColor:
notification.isRead ? 'white' : '#e3f2fd' }}>
            <CardContent>
              <Typography>{notification.message}</Typography>
              <Typography variant="caption">{new
Date(notification.createdAt).toLocaleString()}</Typography>
            </CardContent>
            {!notification.isRead && (
              <CardActions>
                <Button size="small" onClick={() =>
markNotificationAsRead(notification._id)}>
                  Tandai Sudah Dibaca
                </Button>
              </CardActions>
            )}
          </Card>

```

```

    ))
  ) : (
    <Typography>Tidak ada notifikasi baru.</Typography>
  )}
</Box>

<Typography variant="h5" gutterBottom>Laporan
Kehilangan</Typography>
{reports.length > 0 ? (
  <Grid container spacing={2}>
    {reports.map((report, index) => (
      <Grid item xs={12} md={6} lg={4} key={index}>
        <Card>
          <CardContent>
            <Typography variant="h6">Jenis Item:
{report.itemType}</Typography>
            <Box sx={{ mt: 1 }}>
              <Typography variant="body2">Deskripsi:
{report.description}</Typography>
              <Typography variant="body2">Lokasi:
{report.location}</Typography>
              <Typography variant="body2">Kontak:
{report.contactInfo}</Typography>
              <Typography variant="body2">Waktu Kehilangan:
{new Date(report.timeLost).toLocaleString()}</Typography>
            </Box>
          </CardContent>
          <CardActions disableSpacing>
            <Button startIcon={<ClaimIcon />}
variant="contained" color="primary" onClick={() =>
claimItem(report._id)}>
              Klaim Barang Ini
            </Button>
          </CardActions>
        </Card>
      </Grid>
    )})}
  </Grid>
) : (
  <Typography>Tidak ada laporan kehilangan saat
ini.</Typography>
)}}

```

```

        <Button variant="contained" color="primary" sx={{ mt: 2 }}
component={Link} to='/report'>
        Buat Laporan Kehilangan Baru
    </Button>
    <Button variant="contained" color="secondary" sx={{ mt: 2, ml:
2 }} onClick={handleLogout}>
        Logout
    </Button>
</Container>
);
}

```

Code ini adalah halaman dashboard untuk pengguna di sebuah aplikasi. Setelah pengguna login, halaman ini menampilkan berbagai laporan kehilangan yang telah dibuat oleh pengguna serta notifikasi yang diterima. Saat halaman dibuka, data laporan dan notifikasi diambil dari server menggunakan token otentikasi yang disimpan di penyimpanan lokal perangkat pengguna. Jika ada kesalahan selama proses pengambilan data, pesan kesalahan akan ditampilkan.

Notifikasi yang diterima pengguna ditampilkan di bagian atas halaman, dan jika pengguna ingin menandai notifikasi sebagai sudah dibaca, mereka bisa mengklik tombol yang tersedia. Setiap laporan kehilangan yang dibuat oleh pengguna ditampilkan dalam bentuk kartu dengan rincian seperti jenis item, deskripsi, lokasi, informasi kontak, dan waktu kehilangan. Pengguna juga bisa mengklaim barang yang dilaporkan hilang dengan mengklik tombol "Klaim Barang Ini" di kartu laporan. Jika klaim berhasil, pengguna akan diberi tahu melalui pesan.

Selain itu, halaman ini juga menyediakan tombol untuk membuat laporan kehilangan baru atau untuk logout dari aplikasi. Saat pengguna mengklik logout, data otentikasi akan dihapus dan pengguna akan diarahkan kembali ke halaman login. Halaman ini dirancang untuk memberikan akses mudah bagi pengguna untuk melihat dan mengelola laporan kehilangan serta menerima notifikasi penting terkait laporan mereka.

Backend ExpressJS atau sisi server untuk dashboard

```
const router = require('express').Router();
let LostItem = require('../models/LostItemModel');

// Dapatkan semua laporan kehilangan
router.get('/', async (req, res) => {
  try {
    const lostItems = await LostItem.find();
    res.json(lostItems);
  } catch (err) {
    res.status(400).json('Error: ' + err);
  }
});
```

```
const express = require('express');
const router = express.Router();
const Notification = require('../models/NotificationModel');
const auth = require('../middleware/auth.js');

// Get all notifications for a user
router.get('/', auth, async (req, res) => {
  console.log('Fetching notifications for user:', req.user.id);
  try {
    const notifications = await Notification.find({ user:
req.user.id })
      .sort({ createdAt: -1 })
      .populate('relatedItem');
    console.log('Notifications found:', notifications.length);
    res.json(notifications);
  } catch (err) {
    console.error('Error fetching notifications:', err);
    res.status(500).json({ message: err.message });
  }
});

// Mark a notification as read
router.patch('/:id', auth, async (req, res) => {
  try {
    const notification = await Notification.findOneAndUpdate(
      { _id: req.params.id, user: req.user.id },
```

```

        { isRead: true },
        { new: true }
    );
    if (!notification) {
        return res.status(404).json({ message: 'Notification not
found' });
    }
    res.json(notification);
} catch (err) {
    res.status(400).json({ message: err.message });
}
});

```

Code ini terdiri dari dua bagian utama yang terkait dengan pengelolaan laporan kehilangan dan notifikasi untuk pengguna. Pada bagian pertama, ada rute yang mengizinkan pengguna untuk mendapatkan semua laporan kehilangan dari database. Ketika permintaan diterima, sistem mencoba mengambil semua laporan kehilangan yang ada dan mengirimkannya kembali kepada pengguna. Jika terjadi kesalahan selama proses ini, pesan kesalahan akan dikirimkan kembali.

Pada bagian kedua, ada rute yang mengelola notifikasi pengguna. Ketika pengguna meminta untuk melihat semua notifikasi, sistem akan memeriksa identitas pengguna menggunakan middleware otentikasi, kemudian mencari semua notifikasi yang terkait dengan pengguna tersebut dalam database. Notifikasi ini diurutkan berdasarkan waktu pembuatan dari yang terbaru dan termasuk informasi terkait item yang relevan. Jika ditemukan, notifikasi dikirim kembali kepada pengguna; jika terjadi kesalahan, pesan kesalahan dikirimkan.

Selain itu, ada juga rute untuk menandai notifikasi sebagai sudah dibaca. Ketika pengguna mengirim permintaan untuk menandai notifikasi tertentu sebagai sudah dibaca, sistem akan memperbarui status notifikasi tersebut di database. Jika notifikasi yang dimaksud tidak ditemukan, pesan kesalahan akan dikirimkan. Jika berhasil, notifikasi yang diperbarui akan dikirim kembali kepada pengguna.

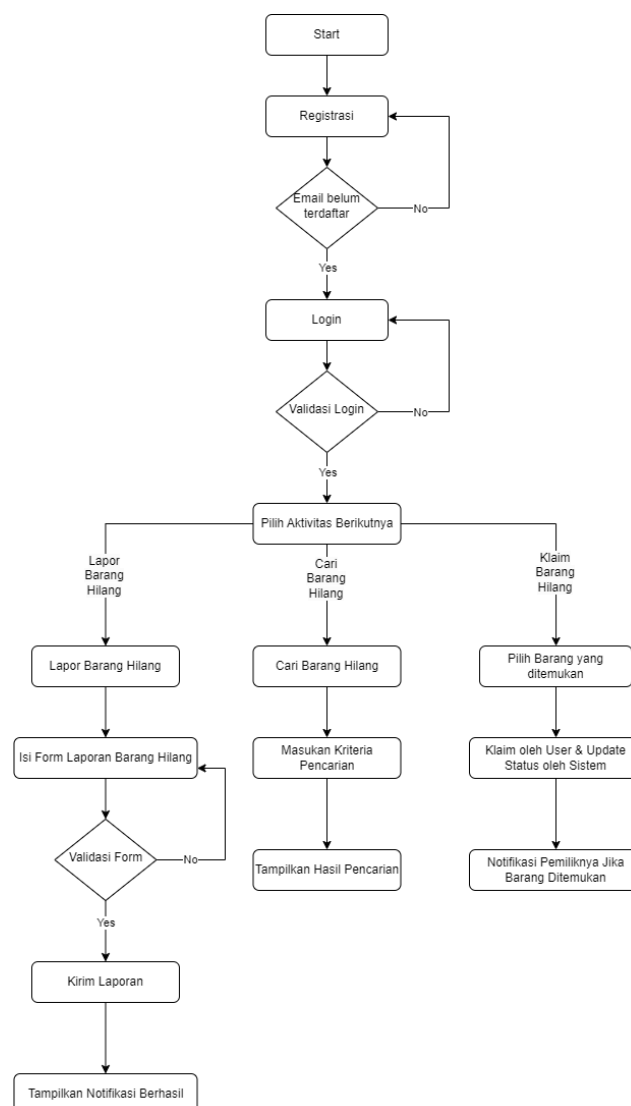
Soal nomor 2:

Jelaskan secara tertulis dan buatlah video tutorial yang mengupas server-side, bagaimana anda mengimplementasikan salah satu servis yang menjadi tanggung jawab anda dengan menggunakan NodeJS. (Note: Wajah wajib kelihatan saat memberikan penjelasan tsb, jika tidak nilai NOL. :))

- Jelaskan terlebih dulu servis yang akan dibuat (kegunaan, flowchart, rancangan).
- Jelaskan bagaimana menerapkannya dengan NodeJS.
- Tunjukkan ujicoba servis tsb dengan Postman.

Jawaban nomor 2:

- Jelaskan terlebih dulu servis yang akan dibuat (kegunaan, flowchart, rancangan).



Flowchart ini menggambarkan proses yang harus diikuti oleh pengguna dalam sistem untuk menggunakan aplikasi layanan pendataan barang dan kendaraan hilang di ITS. Proses dimulai dengan langkah pendaftaran (registrasi), di mana pengguna baru harus mendaftarkan diri mereka ke dalam sistem. Setelah mendaftar, sistem akan memeriksa apakah email pengguna sudah terdaftar sebelumnya. Jika email belum terdaftar, pengguna dapat melanjutkan ke proses login. Jika email sudah terdaftar, pengguna akan diarahkan kembali ke halaman registrasi. Setelah berhasil masuk ke sistem, login pengguna akan divalidasi, dan jika berhasil, pengguna akan diberikan pilihan aktivitas berikutnya, yaitu melapor barang hilang, mencari barang hilang, atau mengklaim barang yang ditemukan.

Jika pengguna memilih untuk melapor barang hilang, mereka harus mengisi formulir laporan barang hilang. Formulir tersebut kemudian akan divalidasi oleh sistem. Jika formulir valid, laporan akan dikirimkan dan sistem akan menampilkan notifikasi bahwa laporan berhasil dikirim. Jika formulir tidak valid, pengguna akan diminta untuk mengisi ulang formulir tersebut.

Jika pengguna memilih untuk mencari barang hilang, mereka harus memasukkan kriteria pencarian tertentu, seperti jenis barang atau lokasi kehilangan. Sistem kemudian akan menampilkan hasil pencarian berdasarkan kriteria yang dimasukkan.

Untuk klaim barang yang ditemukan, pengguna dapat memilih barang yang telah ditemukan dalam sistem. Setelah memilih, pengguna dapat mengklaim barang tersebut, dan sistem akan memperbarui status barang sebagai telah diklaim. Selain itu, sistem juga akan mengirim notifikasi kepada pemilik barang jika barang yang hilang berhasil ditemukan.

Flowchart ini memberikan gambaran tentang langkah-langkah yang harus diikuti oleh pengguna dalam melaporkan, mencari, dan mengklaim barang hilang, serta bagaimana sistem memproses setiap langkah untuk memastikan proses berjalan dengan lancar dan efisien.

b. Jelaskan bagaimana menerapkannya dengan NodeJS.

Aplikasi ini merupakan sistem layanan pendataan barang dan kendaraan hilang yang dibangun menggunakan NodeJS dan Express. Sistem ini menyediakan fungsionalitas bagi pengguna untuk melaporkan barang yang hilang, mendaftarkan barang yang ditemukan, mengklaim barang, dan menerima pemberitahuan. Sistem ini terdiri dari beberapa layanan yang saling berhubungan, masing-masing menangani aspek aplikasi tertentu.

1. User Service

File: routes/users.js

Description: Menangani operasi terkait pengguna seperti registrasi, otentikasi, dan manajemen pengguna.

Key Features:

- User registration
- User authentication (login)
- User logout
- JWT token generation for authenticated requests

Code:

```
const express = require('express');
const router = express.Router();
const bcrypt = require('bcryptjs');
const passport = require('passport');
const jwt = require('jsonwebtoken');
const User = require('../models/UserModel');

// Registrasi Pengguna
router.post('/register', async (req, res) => {
  try {
    let { name, email, password } = req.body;

    // Periksa apakah user sudah terdaftar
    let user = await User.findOne({ email: email });
```

```

    if (user) {
      return res.status(400).json({ message: "User sudah terdaftar"
    });
    }

    // Enkripsi password
    const salt = await bcrypt.genSalt(10);
    password = await bcrypt.hash(password, salt);

    // Buat user baru
    user = new User({
      name,
      email,
      password
    });

    // Simpan user
    await user.save();

    res.status(201).json({ success: true, message: "User berhasil
didaftarkan" });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: "Server error" });
  }
});

// Login Handle
router.post('/login', (req, res, next) => {
  console.log('Received login request with body:', req.body);
  passport.authenticate('local', (err, user, info) => {
    if (err) {
      return res.status(500).json({ message: 'Internal server error'
    });
    }
    if (!user) {
      return res.status(401).json({ message: 'Email atau password
salah' });
    }
    // Menggenerate token
    const token = jwt.sign({ id: user._id }, 'rahasia', {
      expiresIn: 86400 // expires in 24 hours

```

```

});
req.login(user, (err) => {
  if (err) {
    return res.status(500).json({message: err.message});
  }
  return res.json({
    message: 'Login berhasil',
    user: {
      _id: user._id,
      email: user.email,
      name: user.name
    },
    token: token
  });
});
})(req, res, next);
});

module.exports = router;

// Logout Handle
router.get('/logout', (req, res) => {
  req.logout();
  res.redirect('/users/login');
});

```

2. Lost Item Service

File: routes/lostItems.js

Description: Mengelola operasi yang terkait dengan barang hilang yang dilaporkan oleh pengguna.

Key Features:

- Tambahkan laporan barang hilang baru
- Ambil semua laporan barang yang hilang
- Memperbarui laporan barang hilang yang ada

- Hapus laporan barang hilang
- Mencari barang yang hilang berdasarkan deskripsi atau lokasi

Key Endpoints:

- POST /lostitems/add: Tambahkan item baru yang hilang
- GET /lostitems: Ambil semua barang yang hilang
- PUT /lostitems/update/:id: Perbarui item tertentu yang hilang
- DELETE /lostitems/:id: Hapus item tertentu yang hilang
- GET /lostitems/search: Cari barang yang hilang

Code:

```
const router = require('express').Router();
let LostItem = require('../models/LostItemModel');
const FoundItem = require('../models/FoundItemModel');
const { createNotification } =
require('../utils/notificationUtils');

// Tambahkan laporan kehilangan baru
router.post('/add', async (req, res) => {
  const newLostItem = new LostItem({ ...req.body });

  try {
    const savedItem = await newLostItem.save();

    // Cari item yang ditemukan yang cocok
    const foundItem = await FoundItem.findOne({
      itemType: savedItem.itemType,
      description: { $regex: savedItem.description, $options: 'i' }
    });

    if (foundItem) {
      // Kirim notifikasi ke pengguna yang melaporkan item hilang
      await createNotification(
        savedItem.reportedBy,
        `Item yang cocok dengan laporan Anda mungkin telah
ditemukan: ${foundItem.description}`,

```

```

        'item_found',
        foundItem._id,
        'FoundItem'
    );
}

res.status(201).json(savedItem);
} catch (err) {
    res.status(400).json('Error: ' + err);
}
});

// Dapatkan semua laporan kehilangan
router.get('/', async (req, res) => {
    try {
        const lostItems = await LostItem.find();
        res.json(lostItems);
    } catch (err) {
        res.status(400).json('Error: ' + err);
    }
});

// Update laporan kehilangan berdasarkan ID
router.put('/update/:id', async (req, res) => {
    try {
        const updatedItem = await
LostItem.findByIdAndUpdate(req.params.id, req.body, { new: true });
        res.json(updatedItem);
    } catch (err) {
        res.status(400).json('Error: ' + err);
    }
});

// Hapus laporan kehilangan berdasarkan ID
router.delete('/:id', async (req, res) => {
    try {
        await LostItem.findByIdAndDelete(req.params.id);
        res.json('Lost item has been deleted.');
```

```

router.get('/search', async (req, res) => {
  try {
    const { query } = req.query;
    // Mencari dokumen yang cocok dengan query di field description
    atau location
    const foundItems = await LostItem.find({
      $or: [
        { description: { $regex: query, $options: 'i' } },
        { location: { $regex: query, $options: 'i' } }
      ]
    });
    res.json(foundItems);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

router.get('/:id', async (req, res) => {
  try {
    const item = await LostItem.findById(req.params.id);
    res.json(item);
  } catch (error) {
    res.status(404).json({ message: "Item not found." });
  }
});

module.exports = router;

```

3. Found Item Service

File: routes/foundItems.js

Description: Mengelola operasi yang terkait dengan item yang telah ditemukan dan dilaporkan oleh pengguna.

Key Features:

- Tambahkan laporan item baru yang ditemukan
- Ambil semua laporan item yang ditemukan

- Memperbarui laporan barang yang ditemukan
- Hapus laporan item yang ditemukan
- Klaim barang yang ditemukan
- Ambil item yang diklaim untuk pengguna tertentu

Key Endpoints:

- POST /founditems/add: Tambahkan item baru yang ditemukan
- GET /founditems: Ambil semua item yang ditemukan
- PUT /founditems/update/:id: Perbarui item tertentu yang ditemukan
- DELETE /founditems/:id: Hapus item tertentu yang ditemukan
- PUT /founditems/claim/:id: Klaim item yang ditemukan
- GET /founditems/claimeditems/:userId: Dapatkan item yang diklaim oleh pengguna tertentu

Code:

```
const express = require('express');
const router = express.Router();
const FoundItem = require('../models/FoundItemModel');
const LostItem = require('../models/LostItemModel');
const auth = require('../middleware/auth');
const { createNotification } =
require('../utils/notificationUtils');

// Tambahkan item temuan baru
router.post('/add', async (req, res) => {
  const newItem = new FoundItem({ ...req.body });

  try {
    const savedItem = await newItem.save();

    // Cari item yang hilang yang cocok
    const lostItem = await LostItem.findOne({
      itemType: savedItem.itemType,
      description: { $regex: savedItem.description, $options: 'i' }
    });
```

```

    });

    if (lostItem) {
        // Kirim notifikasi ke pengguna yang melaporkan kehilangan
        await createNotifcation(
            lostItem.reportedBy,
            `Item yang Anda laporkan hilang mungkin telah ditemukan:
${savedItem.description}`,
            'item_found',
            savedItem._id,
            'FoundItem'
        );
    }

    res.status(201).json(savedItem);
} catch (err) {
    res.status(400).json('Error: ' + err);
}
});

// Dapatkan semua item temuan
router.get('/', async (req, res) => {
    try {
        const foundItems = await FoundItem.find();
        res.json(foundItems);
    } catch (err) {
        res.status(400).json('Error: ' + err);
    }
});

// Update item temuan berdasarkan ID
router.put('/update/:id', async (req, res) => {
    try {
        const updatedItem = await
FoundItem.findByIdAndUpdate(req.params.id, req.body, { new: true });
        res.json(updatedItem);
    } catch (err) {
        res.status(400).json('Error: ' + err);
    }
});

// Hapus item temuan berdasarkan ID

```

```

router.delete('/:id', async (req, res) => {
  try {
    await FoundItem.findByIdAndDelete(req.params.id);
    res.json('Found item has been deleted.');
```

```

  } catch (err) {
    res.status(400).json('Error: ' + err);
  }
});

// Endpoint untuk mengklaim barang
// Route untuk mengklaim barang
router.put('/claim/:id', auth, async (req, res) => {
  const { id } = req.params;
  const userId = req.user.id;

  console.log(`Attempting to claim item with ID: ${id}`);

  try {
    const lostItem = await LostItem.findById(id);
    if (!lostItem) {
      return res.status(404).json({ message: 'Lost item not found.'
});
    }

    let foundItem = await FoundItem.findOne({ lostItemRef:
lostItem._id });
    if (!foundItem) {
      foundItem = new FoundItem({
        itemType: lostItem.itemType,
        description: lostItem.description,
        locationFound: lostItem.location,
        dateFound: new Date(),
        contactInfo: lostItem.contactInfo,
        lostItemRef: lostItem._id,
        reportedBy: userId
      });

      if (foundItem.status === 'claimed') {
        return res.status(400).json({ message: 'Found item already
claimed.' });
      }
    }
  }
});

```

```

    foundItem.status = 'claimed';
    foundItem.claimedBy = userId;
    await foundItem.save();

    console.log(`Found item with ID: ${foundItem._id} claimed.`);

    // Create notification for the user who reported the lost item
    await createNotification(
        lostItem.reportedBy.toString(),
        `telah menemukan item yang Anda laporkan hilang:
${lostItem.description}`,
        'item_found',
        foundItem._id,
        'FoundItem',
        userId // ID of the user who found/claimed the item
    );

    res.json({ message: 'Found item successfully claimed.' });
} catch (error) {
    console.error(`Error when claiming found item with lost item ID:
${id}`, error);
    res.status(500).json({ message: 'Server error.' });
}
});

// Route untuk mendapatkan daftar item yang telah diklaim oleh
pengguna tertentu
router.get('/claimeditems/:userId', async (req, res) => {
    const { userId } = req.params;

    try {
        const claimedItems = await FoundItem.find({ claimedBy: userId
    });
        res.json(claimedItems);
    } catch (error) {
        console.error(`Error when retrieving claimed items for user ID:
${userId}`, error);
        res.status(500).send('Server error.');
```

```
module.exports = router;
```

4. Notification Service

File: routes/notifications.js dan utils/notificationUtils.js

Description: Mengelola pembuatan, pengambilan, dan penanganan notifikasi untuk pengguna.

Key Features:

- Buat notifikasi baru
- Ambil semua notifikasi untuk pengguna
- Menandai notifikasi sebagai telah dibaca
- Hapus notifikasi

Key Endpoints:

- GET /notifications: Ambil semua notifikasi untuk pengguna yang diautentikasi
- PATCH /notifications/:id: Tandai notifikasi tertentu sebagai telah dibaca
- DELETE /notifications/:id: Hapus notifikasi tertentu

Utility Functions:

- createNotification: Membuat notifikasi baru untuk pengguna

Code:

```
const express = require('express');
const router = express.Router();
const Notification = require('../models/NotificationModel');
const auth = require('../middleware/auth.js');

// Get all notifications for a user
router.get('/', auth, async (req, res) => {
  console.log('Fetching notifications for user:', req.user.id);
  try {
```

```

    const notifications = await Notification.find({ user:
req.user.id })
    .sort({ createdAt: -1 })
    .populate('relatedItem');
    console.log('Notifications found:', notifications.length);
    res.json(notifications);
  } catch (err) {
    console.error('Error fetching notifications:', err);
    res.status(500).json({ message: err.message });
  }
});

// Mark a notification as read
router.patch('/:id', auth, async (req, res) => {
  try {
    const notification = await Notification.findOneAndUpdate(
      { _id: req.params.id, user: req.user.id },
      { isRead: true },
      { new: true }
    );
    if (!notification) {
      return res.status(404).json({ message: 'Notification not
found' });
    }
    res.json(notification);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
});

// Delete a notification
router.delete('/:id', auth, async (req, res) => {
  try {
    const notification = await Notification.findOneAndDelete({ _id:
req.params.id, user: req.user.id });
    if (!notification) {
      return res.status(404).json({ message: 'Notification not
found' });
    }
    res.json({ message: 'Notification deleted' });
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

```

```
    }  
  });  
  
module.exports = router;
```

```
const Notification = require('../models/NotificationModel');  
const User = require('../models/UserModel');  
  
const createNotification = async (userId, message, type,  
  relatedItemId = null, itemModel = null, finderId = null) => {  
  console.log('Creating notification with params:', { userId,  
    message, type, relatedItemId, itemModel, finderId });  
  
  if (!userId || typeof userId !== 'string' || userId.trim() === '')  
  {  
    throw new Error('Valid user ID is required to create a  
notification');  
  }  
  
  try {  
    let finderName = 'Seseorang';  
    if (finderId) {  
      const finder = await User.findById(finderId);  
      if (finder) {  
        finderName = finder.name || finder.email || 'Seseorang';  
      }  
    }  
  }  
  
  const notification = new Notification({  
    user: userId,  
    message: `${finderName} ${message}`,  
    type,  
    relatedItem: relatedItemId,  
    itemModel: itemModel  
  });  
  
  const savedNotification = await notification.save();  
  console.log('Notification created successfully:',  
savedNotification);  
  return savedNotification;  
} catch (error) {
```

```

        console.error('Error creating notification:', error);
        throw error;
    }
};

module.exports = { createNotification };

```

5. Authentication Middleware

File: middleware/auth.js

Description: Middleware untuk mengautentikasi permintaan menggunakan token JWT.

Key Features:

- Verifikasi token JWT di header Otorisasi
- Tambahkan pengguna yang diautentikasi ke objek permintaan
- Menangani masa berlaku token dan token yang tidak valid

Code:

```

const jwt = require('jsonwebtoken');
const User = require('../models/UserModel');

module.exports = async function(req, res, next) {
    // Get token from header
    const authHeader = req.header('Authorization');

    if (!authHeader) {
        return res.status(401).json({ message: 'No token, authorization denied' });
    }

    // Check if it's a Bearer token
    const parts = authHeader.split(' ');
    if (parts.length !== 2 || parts[0] !== 'Bearer') {
        return res.status(401).json({ message: 'Authorization header must be Bearer token' });
    }

```



```

const token = parts[1];

try {
  // Verify token
  const decoded = jwt.verify(token, process.env.JWT_SECRET);

  // Add user from payload
  req.user = await User.findById(decoded.id).select('-password');
  if (!req.user) {
    return res.status(401).json({ message: 'User not found' });
  }
  next();
} catch (err) {
  if (err.name === 'TokenExpiredError') {
    return res.status(401).json({ message: 'Token expired' });
  }
  res.status(401).json({ message: 'Token is not valid' });
}
};

```

c. Tampilkan ujicoba servis tsb dengan Postman.

1. Registrasi Pengguna (POST /users/register)

a. Registrasi Berhasil

- Method: POST
- URL: http://localhost:3000/users/register
- Headers: Content-Type: application/json
- Body (raw JSON):

```

{
  "name": "John Doe",
  "email": "john@example.com",
  "password": "password123"
}

```

- Expected Response:

- Status Code: 201
- Body:

```
{  
  
  "success": true,  
  
  "message": "User berhasil didaftarkan"  
  
}
```

b. Registrasi Gagal (Email Sudah Terdaftar)

- Method: POST
- URL: `http://localhost:3000/users/register`
- Headers: Content-Type: application/json
- Body (raw JSON):

```
{  
  
  "name": "John Doe",  
  
  "email": "john@example.com",  
  
  "password": "password123"  
  
}
```

- Expected Response:
 - Status Code: 400
 - Body:

```
{  
  
  "message": "User sudah terdaftar"  
  
}
```

2. Login Pengguna (POST /users/login)

a. Login Berhasil

- Method: POST
- URL: http://localhost:3000/users/login
- Headers: Content-Type: application/json
- Body (raw JSON):

```
{
  "email": "john@example.com",
  "password": "password123"
}
```

- Expected Response:
 - Status Code: 200
 - Body:

```
{
  "message": "Login berhasil",
  "user": {
    "_id": "some_user_id",
    "email": "john@example.com",
    "name": "John Doe"
  },
  "token": "jwt_token_here"
}
```

b. Login Gagal (Email atau Password Salah)

- Method: POST
- URL: http://localhost:3000/users/login
- Headers: Content-Type: application/json
- Body (raw JSON):

```
{  
  
  "email": "john@example.com",  
  
  "password": "wrongpassword"  
  
}
```

- Expected Response:
 - Status Code: 401
 - Body:

```
{  
  
  "message": "Email atau password salah"  
  
}
```

3. Logout Pengguna (GET /users/logout)

- Method: GET
- URL: http://localhost:3000/users/logout
- Expected Behavior: Pengguna akan di-logout dan diarahkan ke halaman login

4. Tambahkan Item Baru yang Hilang (POST /lostitems/add)

- Method: POST
- URL: http://localhost:3000/lostitems/add
- Headers:
 - Content-Type: application/json
 - Authorization: Bearer {token}
- Body:

```
{  
  
  "itemType": "Phone",  
  
  "description": "iPhone 12 Pro, Space Gray",  
  
}
```

```
"location": "Central Park",  
  
"timeLost": "2023-06-29T10:00:00Z",  
  
"contactInfo": "john@example.com",  
  
"reportedBy": "{userId}"  
  
}
```

- Expected Response:
 - Status: 201
 - Body: Berisi objek item hilang yang dibuat dengan _id

5. Get Semua Item yang Hilang (GET /lostitems)

- Method: POST
- URL: http://localhost:3000/lostitems/lostitems
- Headers:
 - Authorization: Bearer {token}
- Expected Response:
 - Status: 200
 - Body: Berisi array objek item yang hilang

6. Update Item yang Hilang (PUT /lostitems/update/{id})

- Method: PUT
- URL: http://localhost:3000/lostitems/update/{id}
- Headers:
 - Content-Type: application/json
 - Authorization: Bearer {token}

- Body:

```
{  
  
"description": "Updated description"
```

}

- Expected Response:
 - Status: 200
 - Body: Update objek item yang hilang

7. Delete Item yang Hilang (DELETE /lostitems/{id})

- Method: DELETE
- URL: http://localhost:3000/lostitems/{id}
- Headers:
 - Authorization: Bearer {token}
- Expected Response:
 - Status: 200
 - Body: objek item yang hilang telah dihapus

8. Search Item yang Hilang (GET /lostitems/search)

- Method: GET
- URL: http://localhost:3000/lostitems/search?query=iPhone
- Headers:
 - Authorization: Bearer {token}
- Expected Response:
 - Status: 200
 - Body: Array dari objek item hilang yang cocok

9. Tambahkan Item Baru yang Ditemukan (POST /founditems/add)

- Method: POST
- URL: http://localhost:3000/founditems/add
- Headers:
 - Content-Type: application/json
 - Authorization: Bearer {token}

- Body:

```
{  
  
  "itemType": "Phone",  
  
  "description": "iPhone 12 Pro, Space Gray",  
  
  "locationFound": "Central Park",  
  
  "dateFound": "2023-06-29T11:00:00Z",  
  
  "contactInfo": "jane@example.com"  
}
```

- Expected Response:
 - Status: 201
 - Body: Berisi objek item ditemukan yang dibuat dengan _id

10. Dapatkan Semua Barang yang Ditemukan (GET /founditems)

- Method: GET
- URL: http://localhost:3000/founditems
- Headers:
 - Authorization: Bearer {token}
- Expected Response:
 - Status: 200
 - Body: Array dari objek item yang ditemukan

11. Perbarui Item yang Ditemukan (PUT /founditems/update/{id})

- Method: POST
- URL: http://localhost:3000/founditems/update/{id}
- Headers:
 - Content-Type: application/json
 - Authorization: Bearer {token}
- Body:

```
{  
  
  "description": "Updated description"  
  
}
```

- Expected Response:
 - Status: 200
 - Body: Memperbarui objek item yang ditemukan

12. Hapus Item yang Ditemukan (DELETE /founditems/{id})

- Method: DELETE
- URL: http://localhost:3000/founditems/{id}
- Headers:
 - Authorization: Bearer {token}
- Expected Response:
 - Status: 200
 - Body: "Barang yang ditemukan telah dihapus"

13. Klaim Barang yang Ditemukan (PUT /founditems/claim/{id})

- Method: PUT
- URL: http://localhost:3000/founditems/claim/{id}
- Headers:
 - Authorization: Bearer {token}
- Expected Response:
 - Status: 200
 - Body: { "message": " Barang yang ditemukan berhasil diklaim." }

14. Get Barang yang Diklaim untuk User (GET /founditems/claimeditems/{userId})

- Method: GET
- URL: http://localhost:3000/founditems/claimeditems/{userId}
- Headers:
 - Authorization: Bearer {token}
- Expected Response:

- Status: 200
- Body: Array dari objek item ditemukan yang diklaim

15. Get All Notifications untuk User (GET /notifications)

- Method: GET
- URL: http://localhost:3000/notifications
- Headers:
 - Authorization: Bearer {token}
- Expected Response:
 - Status: 200
 - Body: Array dari objek notifikasi

16. Tandai Notifikasi sebagai Telah Dibaca (PATCH /notifications/{id})

- Method: PATCH
- URL: http://localhost:3000/notifications/{id}
- Headers:
 - Authorization: Bearer {token}
- Expected Response:
 - Status: 200
 - Body: Objek notifikasi diperbarui dengan isRead: true

17. Hapus Notifikasi (DELETE /notifications/{id})

- Method: DELETE
- URL: http://localhost:3000/notifications/{id}
- Headers:
 - Authorization: Bearer {token}
- Expected Response:
 - Status: 200
 - Body: { "message": " Notifikasi dihapus " }

Soal nomor 3:

Jelaskan secara tertulis dan buatlah video tutorial yang mengupas client-side, bagaimana anda penggunaan servis tsb dengan menggunakan Postman, aplikasi web atau mobile phone, pilih salah satu sesuai kemampuan programming anda. (Note: Wajah wajib kelihatan saat memberikan penjelasan tsb, jika tidak nilai NOL. :))

- a. Jelaskan scenario yang akan digunakan untuk demo
- b. Tunjukkan dan jelaskan kodingan aplikasi yang akan didemokan
- c. Tunjukkan bahwa scenario telah sesuai yang diharapkan

Jawaban nomor 3:

- a. Jelaskan scenario yang akan digunakan untuk demo

Untuk demo sistem ini, kita akan menggunakan skenario di mana seorang pengguna mengalami kehilangan barang dan menggunakan sistem untuk melaporkannya serta mencari barang tersebut. Berikut adalah langkah-langkah yang akan diikuti dalam skenario demo:

- 1. Registrasi Pengguna Baru:

Pengguna baru akan membuka aplikasi dan memilih opsi registrasi. Mereka akan memasukkan detail yang diperlukan seperti nama, email, dan kata sandi. Sistem akan memeriksa apakah email yang dimasukkan belum terdaftar sebelumnya. Jika belum, pengguna akan berhasil mendaftar dan diarahkan ke halaman login.

- 2. Login Pengguna:

Setelah berhasil mendaftar, pengguna akan melakukan login dengan memasukkan email dan kata sandi yang telah didaftarkan. Sistem akan memvalidasi informasi login tersebut. Jika informasi valid, pengguna akan masuk ke dalam sistem.

- 3. Melaporkan Barang Hilang:

Setelah login, pengguna akan memilih opsi "Lapor Barang Hilang". Pengguna akan diminta untuk mengisi formulir laporan barang hilang yang mencakup detail seperti jenis barang, deskripsi, lokasi terakhir terlihat, dan waktu kehilangan. Setelah mengisi

formulir, pengguna akan mengirim laporan tersebut. Sistem akan memvalidasi formulir dan menampilkan notifikasi bahwa laporan berhasil dikirim.

4. Mencari Barang Hilang:

Setelah melaporkan barang hilang, pengguna akan mencoba mencari barang yang hilang dengan memilih opsi "Cari Barang Hilang". Pengguna akan memasukkan kriteria pencarian seperti jenis barang dan lokasi kehilangan. Sistem akan menampilkan hasil pencarian berdasarkan kriteria yang dimasukkan. Pengguna dapat melihat daftar barang yang ditemukan yang sesuai dengan kriteria pencarian mereka.

5. Klaim Barang yang Ditemukan:

Jika pengguna menemukan barang yang mirip dengan barang yang hilang dalam hasil pencarian, mereka akan memilih barang tersebut dan mengajukan klaim. Sistem akan meminta konfirmasi dan kemudian memperbarui status barang sebagai telah diklaim oleh pengguna. Sistem juga akan mengirimkan notifikasi kepada pemilik asli barang bahwa barang mereka telah ditemukan dan diklaim.

6. Notifikasi:

Sistem akan memastikan bahwa semua langkah sudah diikuti dengan benar dan memberikan notifikasi kepada pengguna saat klaim barang berhasil dilakukan.

Demo ini akan menunjukkan bagaimana pengguna dapat menggunakan sistem untuk melaporkan barang hilang, mencari barang yang hilang, dan mengklaim barang yang ditemukan dengan mudah dan efisien, serta bagaimana sistem mendukung dan memproses setiap langkah dengan lancar.

- b. Tunjukkan dan jelaskan kodingan aplikasi yang akan didemokan

Code untuk aplikasi yang didemokan akan menggunakan perpaduan antara frontend dan backend, untuk penjelasan codenya adalah sebagai berikut:

1. Registrasi Pengguna Baru:

Di frontend telah disediakan form untuk diisi {name, email, password} berikut code untuk menampung field tersebut:

```
onSubmit=({values, { setSubmitting, resetForm, setErrors }}) => {
  const { confirmPassword, ...dataToSend } = values;
  fetch('/users/register', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(dataToSend),
  })
  .then(response => {
    if (!response.ok) {
      throw new Error(`HTTP error status:
${response.status}`);
    }
    return response.json();
  })
  .then(data => {
    if (data.success) {
      alert('Pendaftaran berhasil!');
      resetForm();
    } else {
      setErrors({ server: data.message });
      alert(`Pendaftaran gagal: ${data.message}`);
    }
  })
  .catch((error) => {
    console.error('Error:', error);
    alert('Gagal mendaftar. Silakan coba lagi.');
```

isian field tersebut kemudian di send atau dikirim ke server untuk dilakukan pengecekan dan validasi dengan code di server:

```
// Registrasi Pengguna
router.post('/register', async (req, res) => {
  try {
    let { name, email, password } = req.body;

    // Periksa apakah user sudah terdaftar
    let user = await User.findOne({ email: email });
    if (user) {
      return res.status(400).json({ message: "User sudah terdaftar"
});
    }

    // Enkripsi password
    const salt = await bcrypt.genSalt(10);
    password = await bcrypt.hash(password, salt);

    // Buat user baru
    user = new User({
      name,
      email,
      password
    });

    // Simpan user
    await user.save();

    res.status(201).json({ success: true, message: "User berhasil
didaftarkan" });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: "Server error" });
  }
});
```

Disini server menggunakan method POST dan melakukan pengecekan apakah user sudah terdaftar, jika belum maka buat user baru dan simpan ke dalam database.

2. Login Pengguna:

Setelah user berhasil registrasi maka user akan diarahkan ke halaman login dan telah disediakan form untuk diisi {email, password} dengan code di frontend sebagai berikut:

```
const userCredentials = { email, password };
console.log('Sending these credentials:', userCredentials);

fetch('/users/login', {
  method: 'POST',
  headers: {
    'Accept': 'application/json',
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(userCredentials),
})
.then(response => {
  console.log('Response status:', response.status);
```

isian field tersebut kemudian di send atau dikirim ke server untuk dilakukan pengecekan dan validasi dengan code di server:

```
// Login Handle
router.post('/login', (req, res, next) => {
  console.log('Received login request with body:', req.body);
  passport.authenticate('local', (err, user, info) => {
    if (err) {
      return res.status(500).json({ message: 'Internal server error'
    });
    }
    if (!user) {
      return res.status(401).json({ message: 'Email atau password
      salah' });
    }
    // Menggenerate token
    const token = jwt.sign({ id: user._id }, 'rahasia', {
      expiresIn: 86400 // expires in 24 hours
    });
    req.login(user, (err) => {
      if (err) {
        return res.status(500).json({message: err.message});
```

```

    }
    return res.json({
      message: 'Login berhasil',
      user: {
        _id: user._id,
        email: user.email,
        name: user.name
      },
      token: token
    });
  });
})(req, res, next);
});

```

Disini server menggunakan method POST dan melakukan pengecekan apakah user email dan password benar, jika benar maka user bisa login dengan digenerate token autentifikasi yang expired hingga 24 jam.

3. Melaporkan Barang Hilang:

Sama seperti code frontend sebelumnya, di frontendnya sendiri telah disediakan form untuk diisi oleh user melalui UI-nya langsung. Code tersebut kemudian akan melakukan pengiriman ke server dengan fetch('/lostitems/add'), berikut codenya:

```

onSubmit=({values, { setSubmitting, resetForm, setErrors }}) => {
  const userInfo =
JSON.parse(localStorage.getItem('userInfo'));
  if (!userInfo || !userInfo._id) {
    alert('Anda harus login untuk melaporkan kehilangan.');
```

navigate('/login');

```
    return;
  }

  const reportData = {
    ...values,
    reportedBy: userInfo._id,
  };

  fetch('/lostitems/add', {
    method: 'POST',

```

```

        headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer
${localStorage.getItem('token')}`
        },
        body: JSON.stringify(reportData),
    })
    .then(response => {
        if (!response.ok) {
            return response.json().then(err => {
                throw new Error(err.message || 'Terjadi kesalahan
saat mengirim laporan.');
```

Setelah dikirim maka server akan menerima permintaan dari client untuk diproses dengan juga melakukan beberapa lapisan pengecekan di sisi server:

```

// Tambahkan laporan kehilangan baru
router.post('/add', async (req, res) => {
    const newLostItem = new LostItem({ ...req.body });

    try {
        const savedItem = await newLostItem.save();
```



```

// Cari item yang ditemukan yang cocok
const foundItem = await FoundItem.findOne({
  itemType: savedItem.itemType,
  description: { $regex: savedItem.description, $options: 'i' }
});

if (foundItem) {
  // Kirim notifikasi ke pengguna yang melaporkan item hilang
  await createNotification(
    savedItem.reportedBy,
    `Item yang cocok dengan laporan Anda mungkin telah
ditemukan: ${foundItem.description}`,
    'item_found',
    foundItem._id,
    'FoundItem'
  );
}

res.status(201).json(savedItem);
} catch (err) {
  res.status(400).json('Error: ' + err);
}
});

```

Setelah berhasil melewati pengecekan maka lost item atau laporan kehilangan bisa ditambahkan ke dalam database

4. Mencari Barang Hilang:

Untuk mencari barang yang hilang akan saya dijelaskan disini servernya karena proses logika ada di dalam server berikut code servernya:

```

router.get('/search', async (req, res) => {
  try {
    const { query } = req.query;
    // Mencari dokumen yang cocok dengan query di field description
    atau location
    const foundItems = await LostItem.find({
      $or: [
        { description: { $regex: query, $options: 'i' } },
        { location: { $regex: query, $options: 'i' } }
      ]
    });
  } catch (err) {
    res.status(400).json('Error: ' + err);
  }
});

```

```

    ]
  });
  res.json(foundItems);
} catch (err) {
  res.status(500).json({ message: err.message });
}
});

```

Method yang digunakan adalah type GET dengan mengambil query dari frontend yang diinputkan oleh user dan dilakukan pengecekan dengan fungsi .find() untuk mencari dari field yang dicari adalah {description dan location}

5. Klaim Barang yang Ditemukan:

Untuk code Klaim Barang yang Ditemukan disisi server adalah sebagai berikut:

```

// Route untuk mengklaim barang
router.put('/claim/:id', auth, async (req, res) => {
  const { id } = req.params;
  const userId = req.user.id;

  console.log(`Attempting to claim item with ID: ${id}`);

  try {
    const lostItem = await LostItem.findById(id);
    if (!lostItem) {
      return res.status(404).json({ message: 'Lost item not found.'
});
    }

    let foundItem = await FoundItem.findOne({ lostItemRef:
lostItem._id });
    if (!foundItem) {
      foundItem = new FoundItem({
        itemType: lostItem.itemType,
        description: lostItem.description,
        locationFound: lostItem.location,
        dateFound: new Date(),
        contactInfo: lostItem.contactInfo,
        lostItemRef: lostItem._id,
        reportedBy: userId

```

```

    });
}

if (foundItem.status === 'claimed') {
    return res.status(400).json({ message: 'Found item already
claimed.' });
}

foundItem.status = 'claimed';
foundItem.claimedBy = userId;
await foundItem.save();

console.log(`Found item with ID: ${foundItem._id} claimed.`);

// Create notification for the user who reported the lost item
await createNotification(
    lostItem.reportedBy.toString(),
    `telah menemukan item yang Anda laporkan hilang:
${lostItem.description}`,
    'item_found',
    foundItem._id,
    'FoundItem',
    userId // ID of the user who found/claimed the item
);

res.json({ message: 'Found item successfully claimed.' });
} catch (error) {
    console.error(`Error when claiming found item with lost item ID:
${id}`, error);
    res.status(500).json({ message: 'Server error.' });
}
});

```

Untuk melakukan claim pastikan kita melakukan login yang otomatis sudah memperoleh userId, lalu userId tersebut digunakan untuk melakukan klaim sehingga nama kita atau dalam hal ini userId kita telah tercatat di database dan akan dikirimkan notifikasi kepada orang yang melaporkan kehilangan barang tersebut.

6. Notifikasi:

Untuk notifikasi ini adalah sebuah respon dari server dengan skenario user lain yang sudah melakukan claim terhadap penemuan kehilangan barang kepada user lain dari laporan yang sudah dibuat, dalam hal ini notifikasi adalah jawaban terhadap user yang telah melakukan klaim kepada pelapornya. Untuk codenya sudah ada di code 'Klaim Barang yang Ditemukan'. untuk isi dari fungsi createNotification adalah sebagai berikut:

```
const createNotification = async (userId, message, type,
relatedItemId = null, itemModel = null, finderId = null) => {
  console.log('Creating notification with params:', { userId,
message, type, relatedItemId, itemModel, finderId });

  if (!userId || typeof userId !== 'string' || userId.trim() === '')
  {
    throw new Error('Valid user ID is required to create a
notification');
  }

  try {
    let finderName = 'Seseorang';
    if (finderId) {
      const finder = await User.findById(finderId);
      if (finder) {
        finderName = finder.name || finder.email || 'Seseorang';
      }
    }

    const notification = new Notification({
      user: userId,
      message: `${finderName} ${message}`,
      type,
      relatedItem: relatedItemId,
      itemModel: itemModel
    });

    const savedNotification = await notification.save();
    console.log('Notification created successfully:',
savedNotification);
    return savedNotification;
  } catch (error) {
```

```
    console.error('Error creating notification:', error);  
    throw error;  
  }  
};
```

- c. Tunjukkan bahwa scenario telah sesuai yang diharapkan

Penjelasan ada di video Tutorial Youtube.