BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama menjalani kegiatan magang, peserta ditempatkan pada Divisi Pengembangan Web dan berperan sebagai Web Developer Laravel, yang bertanggung jawab dalam proses pengembangan aplikasi berbasis web.. Divisi ini merupakan bagian dari Departemen Teknologi Informasi yang memiliki fokus utama pada pengembangan aplikasi berbasis web guna mendukung peningkatan efisiensi dan kinerja sistem internal perusahaan.

Dalam kegiatan magang ini, bimbingan langsung diberikan oleh Muhamad Widyantoro selaku Supervisor. Tanggung jawab utama meliputi pengembangan fitur-fitur aplikasi, melakukan debugging untuk memastikan performa aplikasi tetap optimal, serta menyesuaikan fungsionalitas sistem dengan kebutuhan pengguna.

Peserta magang juga diberikan kesempatan untuk mengeksplorasi beragam pendekatan dan teknologi terkini dalam pengembangan Laravel. Meskipun demikian, setiap perubahan besar yang akan diterapkan pada sistem tetap harus dikonsultasikan dan mendapatkan persetujuan dari Supervisor. Kesempatan untuk mengeksplorasi ini menjadi nilai tambah yang berharga dalam memperdalam pemahaman teknis serta praktik nyata dalam pengembangan aplikasi menggunakan framework Laravel.

3.2 Tugas yang Dilakukan

Selama melaksanakan praktik kerja magang sebagai Web Developer Laravel di PT WINNICODE GARUDA TEKNOLOGI, saya bertanggung jawab dalam mengerjakan berbagai tugas yang berkaitan dengan pengembangan aplikasi berbasis web. Tugas-tugas ini dilaksanakan secara bertahap dan terstruktur sesuai dengan rencana kerja mingguan yang telah ditetapkan bersama pembimbing lapangan.

Secara umum, tugas yang dilakukan meliputi:

1. **Analisis dan Perencanaan Sistem**
   Melakukan identifikasi kebutuhan pengguna dan sistem, menentukan spesifikasi teknis, serta menyusun dokumentasi awal untuk pengembangan sistem.

2. **Perancangan Wireframe dan UI/UX**
   Membuat wireframe dan mockup desain antarmuka menggunakan *Figma* untuk menggambarkan alur navigasi serta tampilan aplikasi yang akan dikembangkan.

3. **Konfigurasi Lingkungan Pengembangan**
   Melakukan setup proyek Laravel dan ReactJS, termasuk konfigurasi database, environment, struktur folder proyek, dan pengaturan *version control* menggunakan Git.

4. **Perancangan Flowchart dan Struktur Database**
   Mendesain flowchart proses utama aplikasi dan menyusun skema struktur basis data menggunakan MySQL untuk mendukung kebutuhan sistem.

5. **Pengembangan Backend Laravel**
   Membangun RESTful API menggunakan Laravel untuk berbagai fitur seperti autentikasi, manajemen pengguna, pengelolaan komentar, pengiriman pesan kontak, dan integrasi dengan API eksternal.

6. **Pengembangan Frontend ReactJS**
   Mengembangkan antarmuka pengguna dengan ReactJS yang terhubung ke API Laravel untuk menampilkan data dinamis, formulir, dan navigasi.

7. **Pengujian dan Debugging**
   Melakukan pengujian menggunakan Postman untuk API serta pengujian manual pada antarmuka pengguna untuk menemukan dan memperbaiki *bug*.

8. **Finalisasi dan Optimalisasi Sistem**
   Menyempurnakan tampilan antarmuka, meningkatkan performa backend, dan mempersiapkan aplikasi agar siap digunakan oleh pengguna akhir.

Setiap tugas dikerjakan di bawah bimbingan supervisor dan disesuaikan dengan standar kerja yang berlaku di perusahaan. Tugas-tugas ini memberikan pemahaman yang mendalam mengenai proses pengembangan perangkat lunak modern serta keterampilan teknis yang relevan dengan industri teknologi saat ini.

3.3 Uraian Pelaksanaan Magang

Selama melaksanakan kerja magang di PT WINNICODE GARUDA TEKNOLOGI, beberapa pekerjaan dilakukan sebagai Web Developer Laravel

intern di departemen Web Developer Laravel. Uraian timeline pekerjaan dapat dilihat pada Tabel 3.1.

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

Minggu Ke -	Pekerjaan yang dilakukan
1	Analisis Perancangan
2-3	Pembuatan Wireframe dan UI/UX
4	Konfigurasi sistem
5-6	Pembuatan Flowchart dan Struktur Database
7-10	Pengembangan Backend
11-14	Pengembangan Frontend
15	Pengujian dan Debuging
16	Finalisasi

Tabel 3.1 menjelaskan pembagian aktivitas magang yang dilakukan oleh pemagang selama periode kerja praktik di PT. Winnicode Garuda Teknologi. Pada minggu pertama, pemagang melakukan analisis perancangan guna memahami kebutuhan sistem serta merumuskan tujuan utama pengembangan portal berita. Selanjutnya, pada minggu ke-2 hingga ke-3, fokus diarahkan pada pembuatan wireframe dan desain UI/UX yang menjadi dasar visual tampilan portal berita.

Memasuki minggu ke-4, dilakukan konfigurasi awal sistem seperti setup environment pengembangan, pengelolaan versi dengan Git, serta instalasi dependensi yang diperlukan. Minggu ke-5 hingga ke-6 diisi dengan pembuatan flowchart dan perancangan struktur database, yang bertujuan untuk memastikan alur sistem dan penyimpanan data berjalan efisien.

Pada minggu ke-7 hingga ke-10, pemagang melaksanakan pengembangan backend menggunakan Laravel, yang mencakup pembuatan API, otentikasi pengguna, dan manajemen konten berita. Pengembangan frontend dilakukan pada minggu ke-11 hingga ke-14 menggunakan React.js dan Tailwind CSS, yang dirancang agar portal memiliki tampilan modern, responsif, dan user-friendly.

Minggu ke-15 digunakan untuk pengujian sistem dan debugging, dengan tujuan untuk memastikan bahwa seluruh fungsi berjalan sesuai rencana tanpa bug yang signifikan. Terakhir, minggu ke-16 difokuskan pada proses finalisasi proyek, termasuk dokumentasi teknis, evaluasi, dan persiapan presentasi akhir kepada pihak perusahaan.

3.3.1 Analisis Perancangan

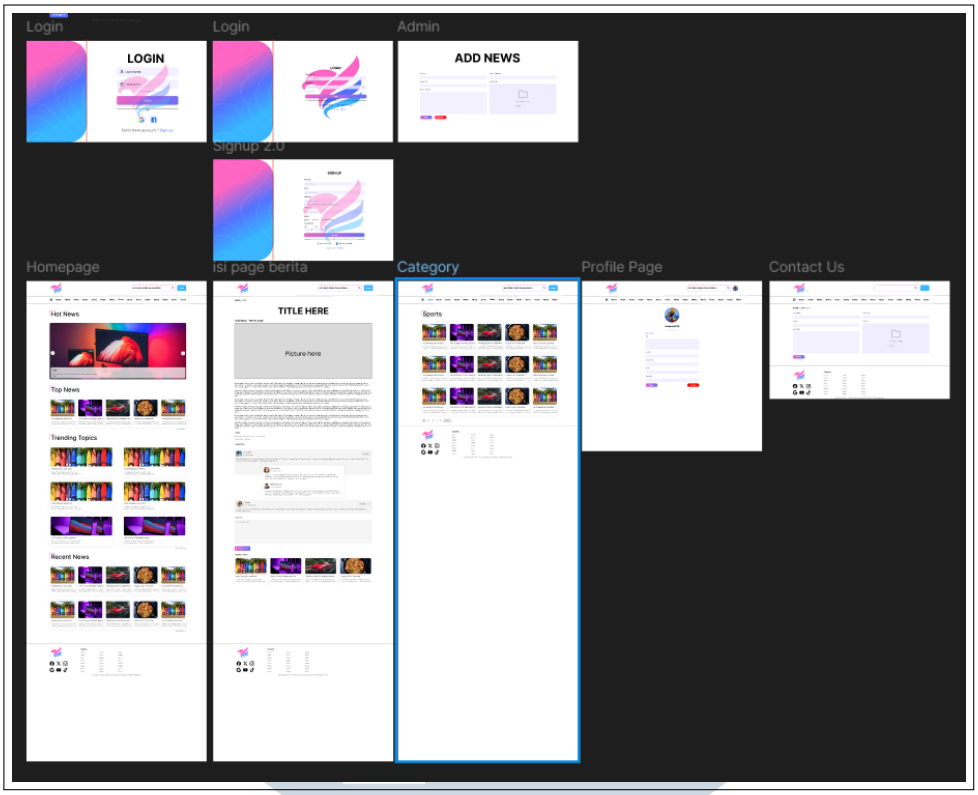
Selama masa magang, saya terlibat dalam tahap analisis dan perencanaan proyek pengembangan perangkat lunak. Saya melakukan riset kebutuhan pengguna untuk memahami permasalahan dan ekspektasi yang ingin diselesaikan melalui sistem yang akan dibangun. Berdasarkan hasil riset tersebut, saya menyusun spesifikasi proyek secara terstruktur, termasuk pemilihan teknologi yang relevan dan sesuai dengan kebutuhan teknis dan fungsional.

Selain itu, saya juga berperan dalam menentukan fitur-fitur utama yang akan dikembangkan serta merancang arsitektur sistem secara menyeluruh. Hal ini mencakup pemetaan alur kerja sistem, struktur basis data, hingga integrasi komponen antar modul agar sistem dapat berjalan secara efisien dan mudah dikembangkan lebih lanjut.

3.3.2 Pembuatan Wireframe dan UI/UX

Dalam tahap perancangan antarmuka, saya bertanggung jawab dalam pembuatan wireframe dan desain UI/UX untuk sistem yang dikembangkan. Menggunakan Figma sebagai alat bantu desain, saya membuat wireframe sebagai kerangka awal tampilan sistem guna memvisualisasikan struktur dan alur navigasi antar halaman. Wireframe ini menjadi dasar dalam penyusunan mockup UI/UX yang mencakup tampilan untuk perangkat desktop maupun mobile, dengan mempertimbangkan prinsip desain responsif dan kenyamanan pengguna.

Setelah mockup selesai, saya juga melakukan proses review desain untuk memastikan konsistensi elemen visual, kemudahan penggunaan, serta kesesuaian dengan kebutuhan pengguna yang telah dianalisis sebelumnya. Proses ini dilakukan secara iteratif agar desain akhir dapat memberikan pengalaman pengguna yang optimal dan mendukung tujuan fungsional dari sistem.



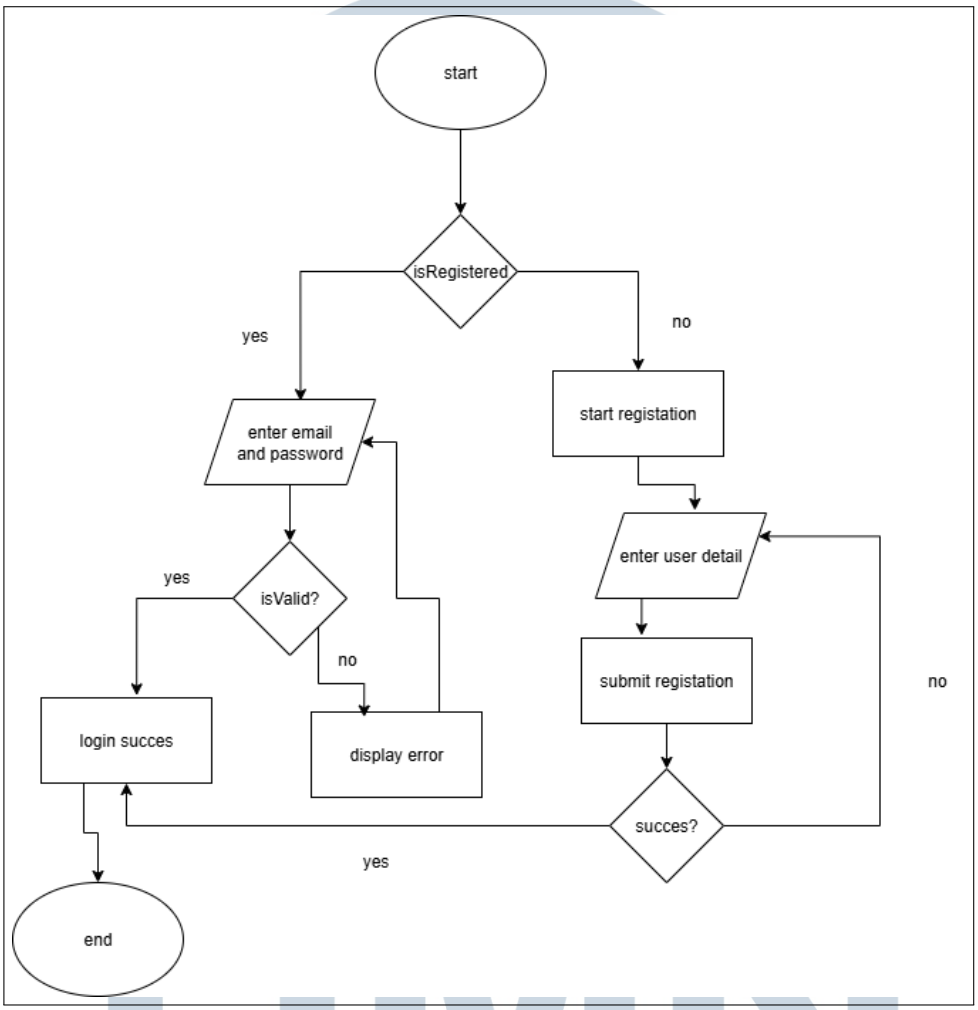
Gambar 3.1. Wireframe Figma

3.3.3 Konfigurasi System

Pada tahap pengaturan lingkungan dan setup teknologi, saya memulai dengan menyiapkan repository proyek di GitHub sebagai pusat pengelolaan versi dan kolaborasi tim pengembang. Setelah itu, saya melakukan instalasi dan konfigurasi environment backend menggunakan Laravel serta MySQL sebagai basis data, memastikan struktur direktori dan koneksi database siap untuk pengembangan lebih lanjut.

Untuk sisi frontend, saya menyiapkan lingkungan kerja menggunakan ReactJS, termasuk pengaturan struktur komponen dan instalasi dependensi yang dibutuhkan. Selain itu, saya juga mengatur konfigurasi API agar komunikasi antara frontend dan backend dapat berjalan lancar dan aman. Proses setup ini menjadi fondasi utama agar sistem dapat dikembangkan dan diuji secara efisien di lingkungan yang terstandarisasi.

3.3.4 Pembuatan Flowchart dan Struktur Database

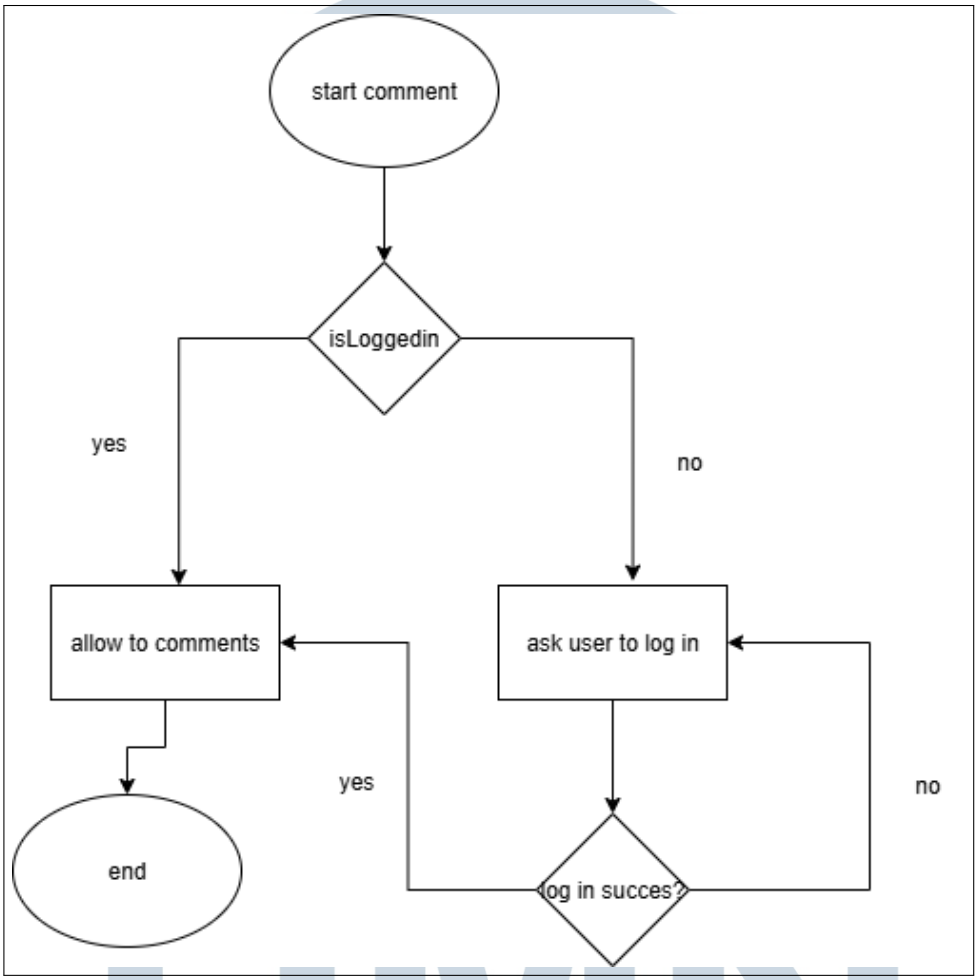


Gambar 3.2. Login Register Flowchart

Flowchart pada gambar tersebut menggambarkan proses login dan registrasi pengguna dalam sebuah sistem. Proses dimulai dengan menanyakan apakah pengguna sudah terdaftar. Jika sudah, pengguna diminta memasukkan username dan password, lalu sistem memverifikasi kredensial tersebut. Jika valid, login berhasil dan proses berakhir. Jika tidak valid, sistem menampilkan pesan kesalahan dan mengarahkan kembali ke form login.

Jika pengguna belum terdaftar, proses registrasi dimulai dengan mengisi data pengguna dan mengirimkannya ke sistem. Jika registrasi berhasil, pengguna diarahkan kembali untuk login, meskipun dalam flowchart ditampilkan sebagai ”DisplayError” yang tampaknya merupakan kesalahan label. Jika registrasi gagal, sistem menampilkan pesan kesalahan dan pengguna diminta mengisi ulang data

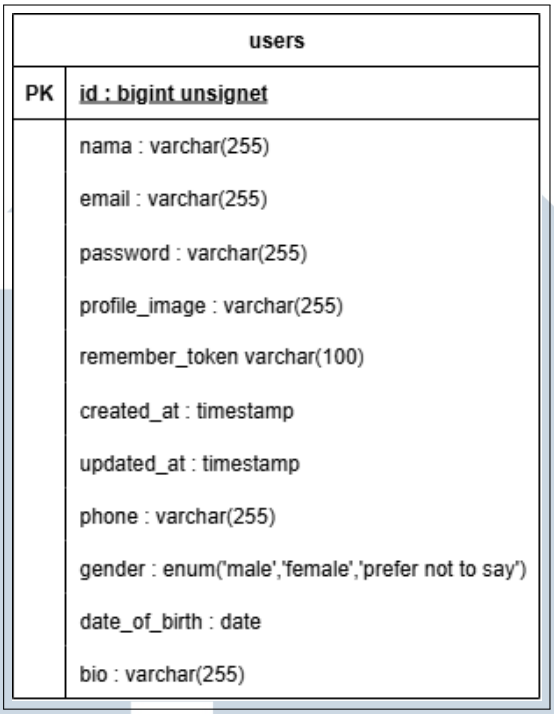
registrasi. Alur ini mencakup penanganan kondisi sukses maupun gagal baik dalam proses login maupun registrasi.



Gambar 3.3. Comment Flowchart

flowchart (diagram alur) tersebut menggambarkan proses alur logika dari fitur komentar dalam sebuah aplikasi atau website. Proses dimulai dari start comment, kemudian sistem akan mengecek apakah pengguna sudah login melalui langkah IsLoggedIn?.

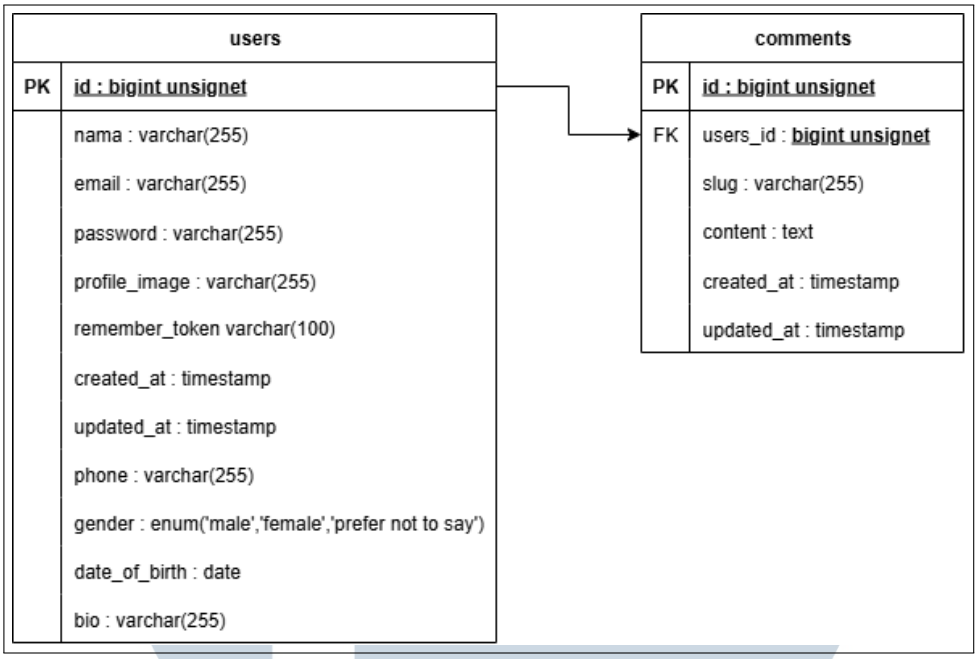
Jika pengguna sudah login, maka langsung diarahkan ke proses allow to comments. Namun jika belum login, sistem akan menampilkan prompt ask user to log in. Selanjutnya akan dicek apakah proses login berhasil. Jika berhasil, pengguna diperbolehkan memberikan komentar. Jika tidak, sistem akan meminta user untuok melakukan log in ulang. Flowchart ini menggambarkan kontrol akses sederhana namun penting untuk menjaga agar hanya pengguna terautentikasi yang dapat memberikan komentar.



Gambar 3.4. Struktur Tabel Database User

Tabel pada gambar tersebut merupakan struktur penyimpanan data pengguna dalam database MySQL. Setiap baris pada tabel mewakili satu pengguna, sementara setiap kolom menyimpan atribut spesifik seperti nama, email, password, dan informasi profil lainnya. Kolom id berfungsi sebagai primary key yang unik dan bertipe bigint dengan AUTO INCREMENT, sehingga nilainya bertambah otomatis untuk setiap pengguna baru. Data disimpan menggunakan tipe data yang sesuai, seperti varchar untuk teks, timestamp untuk waktu, enum untuk pilihan terbatas (seperti gender), dan date untuk tanggal lahir.

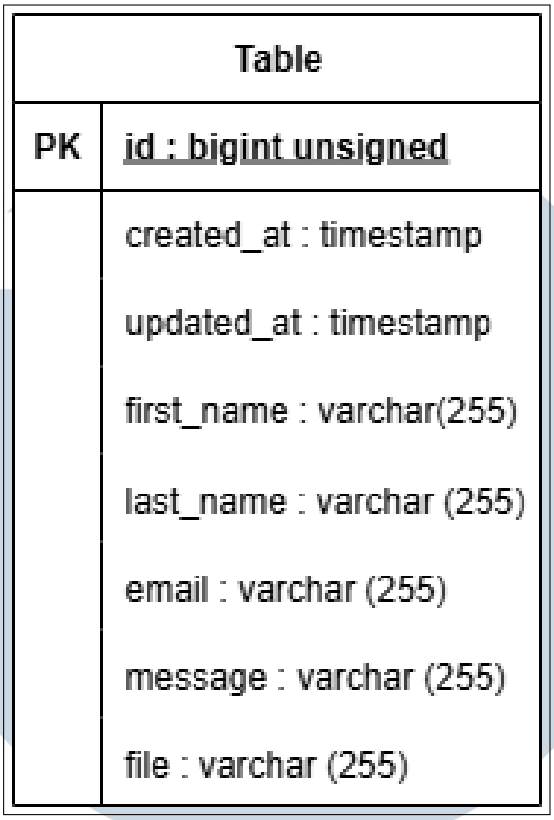
Dengan struktur ini, MySQL menyimpan data secara relasional dan memungkinkan pengelolaan yang efisien melalui query SQL. Validasi data dapat dilakukan melalui aturan kolom seperti NOT NULL, default value, dan tipe data yang ditentukan. Kombinasi atribut seperti email dan password mendukung sistem autentikasi, sedangkan kolom seperti created at dan updated at berguna untuk pelacakan aktivitas pengguna. Struktur ini sangat umum dipakai dalam pengembangan aplikasi berbasis web atau mobile yang memerlukan sistem login dan pengelolaan akun.



Gambar 3.5. Struktur Tabel Database Komentar

Tabel pada gambar tersebut merupakan struktur basis data yang berfungsi untuk menyimpan data konten komentar. Setiap entri memiliki kolom id sebagai primary key dengan atribut auto-increment, serta user id sebagai foreign key yang menghubungkan entri dengan pengguna yang membuatnya. Kolom slug digunakan sebagai URL-friendly identifier, sedangkan content menyimpan isi utama dari artikel atau postingan.

Selain itu, tabel ini juga memiliki kolom created at dan updated at yang mencatat waktu pembuatan dan pembaruan data. Struktur ini dirancang agar mendukung kebutuhan sistem dalam mengelola konten secara efisien, serta memudahkan pelacakan riwayat perubahan data. Tabel ini juga sudah siap untuk diintegrasikan dengan fitur-fitur tambahan seperti autentikasi pengguna, sistem komentar, dan manajemen kategori.



Gambar 3.6. Struktur Tabel Database Message

Tabel pada gambar di atas merupakan struktur basis data yang digunakan untuk menyimpan data dari formulir kontak pengguna. Setiap entri memiliki kolom id sebagai primary key dengan atribut auto-increment, serta kolom created at dan updated at untuk mencatat waktu pembuatan dan pembaruan pesan. Informasi pengguna seperti first name, last name, email, dan isi message disimpan dalam format teks, memungkinkan administrator untuk meninjau dan menanggapi pesan yang masuk.

Selain itu, terdapat kolom file yang bersifat opsional (NULL) dan digunakan untuk menyimpan path file yang diunggah oleh pengguna melalui formulir kontak, seperti dokumen atau lampiran lainnya. Struktur ini dirancang untuk mendukung fitur Contact Us pada aplikasi atau website, dengan fleksibilitas dalam menerima pesan dan file dari pengguna secara terorganisir.

3.3.5 Pengembangan Backend

Dalam pengembangan backend pada proyek ini, saya menggunakan Laravel, sebuah framework PHP yang populer dan memiliki arsitektur MVC (Model-View-Controller). Laravel dipilih karena kemudahan dalam pengelolaan routing,

autentikasi, validasi data, dan manajemen database. Fitur-fitur bawaan Laravel seperti middleware, migration, dan sistem artisan command juga sangat membantu dalam mempercepat proses pengembangan sekaligus menjaga struktur kode tetap rapi dan terorganisir.[10]

Selama proses pengembangan, Laravel digunakan untuk membangun RESTful API yang menjadi penghubung antara frontend dan backend. Saya membuat berbagai endpoint untuk mengelola data pengguna, berita, kategori, komentar, dan pesan pengguna. Selain itu, Laravel juga digunakan untuk mengatur autentikasi pengguna. Secara keseluruhan, Laravel memberikan fondasi backend yang kuat, aman, dan scalable bagi aplikasi web berita ini.

```
class WinnicodeNewsController extends Controller
{
    public function index()
    {
        $apiKey = config('services.winnicode.key');

        $response = Http::withHeaders([
            'Authorization' => "Bearer {$apiKey}",
            'Accept' => 'application/json',
            'Content-Type' => 'application/json',
        ])->get('https://winnicode.com/api/publikasi-berita');

        return response()->json($response->json(), $response->status());
    }

    public function getDetail($slug)
    {
        $apiKey = env('EXTERNAL_API_KEY');
        $response = Http::withHeaders([
            'Authorization' => "Bearer {$apiKey}",
            'Accept' => 'application/json',
            'Content-Type' => 'application/json',
        ])->get("https://winnicode.com/api/publikasi-berita/{$slug}");
```

```
24
25          if ($response->successful()) {
26              return response()->json($response->json());
27          } else {
28              return response()->json(['message' => '
    Failed to fetch article'], $response->status());
29          }
30      }
31  }
```

Kode 3.1: APIController

Potongan kode di atas merupakan controller Laravel yang digunakan untuk mengambil data dari API eksternal milik Winnicode. Dalam method index(), sistem mengambil API key dari konfigurasi dan mengirim permintaan GET ke endpoint https://winnicode.com/api/publikasi-berita menggunakan header otorisasi Bearer, serta menetapkan bahwa data yang diterima dan dikirim bertipe application/json.

Respons dari API kemudian dikembalikan dalam format JSON bersamaan dengan status HTTP-nya, sehingga dapat digunakan langsung oleh frontend atau layanan lain yang membutuhkan data berita tersebut. Pendekatan ini memungkinkan integrasi yang aman dan efisien antara aplikasi internal dengan sistem eksternal melalui pemanggilan REST API.

```
1  class AuthController extends Controller
2  {
3      public function register(Request $request)
4      {
5          // Validasi request
6          $validator = Validator::make($request->all(), [
7              'name' => 'required|string|max:255',
8              'email' => 'required|string|email|max:255|
    unique:users',
9              'password' => 'required|string|min:6',
10             'phone' => 'required|string',
11             'gender' => 'required|string',
12             'date_of_birth' => 'required|date',
13             'profile_image' => 'nullable|image|mimes:
    jpeg,png,jpg,gif|max:2048',
```

```
        ]);

        if ($validator->fails()) {
            return response()->json(['errors' =>
$validator->errors()], 422);
        }

        // Handle profile image upload if provided
        $profileImagePath = null;
        if ($request->hasFile('profile_image')) {
            $image = $request->file('profile_image');
            $profileImagePath = $image->store('profile-
images', 'public');
        }

        // Simpan user baru
        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password
),
            'phone' => $request->phone,
            'gender' => $request->gender,
            'date_of_birth' => $request->date_of_birth,
            'bio' => $request->bio ?? null,
            'profile_image' => $profileImagePath,
        ]);

        return response()->json([
            'message' => 'User registered successfully!
',
            'user' => $user,
            'profile_image_url' => $profileImagePath ?
url('storage/' . $profileImagePath) : null
        ], 201);
    }
```

```
45
46      public function login(Request $request)
47      {
48          // Validate request
49          $validator = Validator::make($request->all(), [
50              'email' => 'required|string|email',
51              'password' => 'required|string',
52          ]);
53
54          if ($validator->fails()) {
55              return response()->json(['errors' =>
    $validator->errors()], 422);
56          }
57
58          // Find user by email
59          $user = User::where('email', $request->email)->
    first();
60
61          // Check if user exists and password matches
62          if (!$user || !Hash::check($request->password,
    $user->password)) {
63              return response()->json([
64                  'message' => 'Invalid credentials'
65              ], 401);
66          }
67
68          // Add profile image URL to response
69          $profileImageUrl = $user->profile_image
70              ? url('storage/' . $user->profile_image)
71              : null;
72
73
74          // Generate token for the user
75          Auth::login($user);
76          $token = $user->createToken('auth_token')->
    plainTextToken;
```

```
77
78          return response()->json([
79              'message' => 'Login successful',
80              'token' => $token,
81              'user' => $user,
82              'profile_image_url' => $profileImageUrl
83          ], 200);
84      }
85
86      public function updateUser(Request $request, $id)
87      {
88          // Validate request
89          $validator = Validator::make($request->all(), [
90              'name' => 'required|string|max:255',
91              'email' => 'required|string|email|max:255|
    unique:users,email,'.$id,
92              'phone' => 'nullable|string',
93              'gender' => 'nullable|string',
94              'date_of_birth' => 'nullable|date',
95              'bio' => 'nullable|string',
96              'profile_image' => 'nullable|image|mimes:
    jpeg,png,jpg,gif|max:2048',
97          ]);
98
99          if ($validator->fails()) {
100             return response()->json(['errors' =>
    $validator->errors()], 422);
101         }
102
103         // Find user
104         $user = User::find($id);
105
106         if (!$user) {
107             return response()->json(['message' => 'User
     not found'], 404);
108         }
```

```
109
110         // Handle profile image upload if provided
111         if ($request->hasFile('profile_image')) {
112             // Delete old image if exists
113             if ($user->profile_image) {
114                 Storage::disk('public')->delete($user->
    profile_image);
115             }
116
117             // Store new image
118             $image = $request->file('profile_image');
119             $profileImagePath = $image->store('profile-
    images', 'public');
120             $user->profile_image = $profileImagePath;
121         }
122
123         // Update user data
124         $user->name = $request->name;
125         $user->email = $request->email;
126         $user->phone = $request->phone;
127         $user->gender = $request->gender;
128         $user->date_of_birth = $request->date_of_birth;
129         $user->bio = $request->bio;
130
131         // Save changes
132         $user->save();
133
134         // Add profile image URL to response
135         $profileImageUrl = $user->profile_image
136             ? url('storage/' . $user->profile_image)
137             : null;
138
139         return response()->json([
140             'message' => 'User updated successfully',
141             'user' => $user,
142             'profile_image_url' => $profileImageUrl
```

```
143         ], 200);
144     }
145
146     public function uploadProfileImage(Request $request
    , $id)
147     {
148         // Validate request
149         $validator = Validator::make($request->all(), [
150             'profile_image' => 'required|image|mimes:
    jpeg,png,jpg,gif|max:2048',
151         ]);
152
153         if ($validator->fails()) {
154             return response()->json(['errors' =>
    $validator->errors()], 422);
155         }
156
157         // Find user
158         $user = User::find($id);
159
160         if (!$user) {
161             return response()->json(['message' => 'User
     not found'], 404);
162         }
163
164         // Delete old image if exists
165         if ($user->profile_image) {
166             Storage::disk('public')->delete($user->
    profile_image);
167         }
168
169         // Store new image
170         $image = $request->file('profile_image');
171         $profileImagePath = $image->store('profile-
    images', 'public');
172         $user->profile_image = $profileImagePath;
```

```
173
174         // Save changes
175         $user->save();
176
177         return response()->json([
178             'message' => 'Profile image uploaded
    successfully',
179             'user' => $user,
180             'profile_image_url' => url('storage/' .
    $profileImagePath)
181         ], 200);
182     }
183
184     public function getUser(Request $request)
185     {
186         $user = Auth::user();
187
188         if (!$user) {
189             return response()->json(['message' => '
    Unauthenticated'], 401);
190         }
191
192         // Add profile image URL to response
193         $profileImageUrl = $user->profile_image
194             ? url('storage/' . $user->profile_image)
195             : null;
196
197         return response()->json([
198             'user' => $user,
199             'profile_image_url' => $profileImageUrl
200         ]);
201     }
202 }
```

Kode 3.2: AuthController

Potongan kode tersebut merupakan bagian dari controller autentikasi

(AuthController) dalam aplikasi Laravel yang mengelola proses registrasi, login, pembaruan data pengguna, serta pengelolaan foto profil. Pada fungsi register(), sistem memvalidasi data yang dikirim oleh pengguna baru, menyimpan informasi tersebut ke dalam basis data, dan menangani unggahan gambar profil bila tersedia. Pada fungsi login(), sistem memverifikasi kredensial pengguna dan memberikan token autentikasi sebagai tanda masuk yang sah, serta mengembalikan data pengguna termasuk URL gambar profil.

Fungsi tambahan seperti updateUser(), uploadProfileImage(), dan getUser() digunakan untuk memperbarui informasi pengguna, mengganti gambar profil, dan mengambil data pengguna yang sedang login. Implementasi ini mencerminkan praktik standar RESTful API yang mendukung keamanan (melalui validasi dan token autentikasi), serta pengalaman pengguna yang lebih lengkap dengan dukungan pengelolaan data profil secara fleksibel dan responsif.

```
class MessagesController extends Controller
{
    // Menampilkan semua pesan
    public function index()
    {
        // Mengambil semua pesan dan mengembalikannya dalam format JSON
        $messages = Messages::all();
        return response()->json($messages);
    }

    // Menyimpan pesan
    public function store(Request $request)
    {
        $request->validate([
            'first_name' => 'required|string|max:255',
            'last_name' => 'required|string|max:255',
            'email' => 'required|email|max:255',
            'message' => 'required|string',
            'file' => 'nullable|file|max:2048', // Maksimal 2MB
        ]);

```

```
22          $filePath = null;
23
24          if ($request->hasFile('file')) {
25              $filePath = $request->file('file')->store('
    contact_files', 'public');
26          }
27
28          Messages::create([
29              'first_name' => $request->first_name,
30              'last_name' => $request->last_name,
31              'email' => $request->email,
32              'message' => $request->message,
33              'file' => $filePath,
34          ]);
35
36          return response()->json(['message' => 'Pesan
    berhasil dikirim!']);
37      }
38
39      // Menampilkan pesan berdasarkan ID
40      public function show($id)
41      {
42          $contact = Messages::findOrFail($id);
43          return response()->json($contact);
44      }
45
46      // Menghapus pesan berdasarkan ID
47      public function destroy($id)
48      {
49          $contact = Messages::findOrFail($id);
50          $contact->delete();
51          return response()->json(['message' => 'Pesan
    berhasil dihapus!']);
52      }
53  }
```

Kode 3.3: MessageController

Kode tersebut merupakan controller Laravel bernama MessagesController yang menangani fitur kontak pesan dari pengguna ke sistem. Controller ini menyediakan fungsionalitas penuh berbasis RESTful API untuk mengelola data pesan, termasuk menampilkan, menyimpan, menampilkan berdasarkan ID, dan menghapus pesan.

Metode store() bertugas menerima input dari form kontak yang dikirimkan pengguna, memvalidasi data seperti nama, email, isi pesan, dan file opsional yang diunggah (maksimal 2MB). Jika file disertakan, file tersebut akan disimpan di direktori storage/public/contact files, dan path-nya dicatat ke dalam basis data. Metode index() digunakan untuk mengambil semua pesan, sedangkan show(id) untuk mengambil satu pesan tertentu berdasarkan ID. Metode destroy(id) memungkinkan penghapusan pesan berdasarkan ID yang diberikan. Fitur ini bermanfaat sebagai bagian dari halaman Contact Us, serta mendukung pengelolaan interaksi pengguna melalui backend secara efektif.

```
class CommentController extends Controller
{
    // Get all comments (for debugging/admin)
    public function index()
    {
        return response()->json(
            Comment::with('user')->latest()->get()
        );
    }

    // Store comment
    public function store(Request $request)
    {
        // Validate only slug and content; user_id from auth
        $request->validate([
            'slug'    => 'required|string|max:255',
            'content' => 'required|string|max:1000',
        ]);

        $user = Auth::user();

```

```
22
23          $comment = Comment::create([
24              'user_id' => $user->id,
25              'slug'    => $request->slug,
26              'content' => $request->content,
27          ]);
28
29         $comment->load('user');
30         return response()->json($comment, 201);
31     }
32
33     // Get comments by article slug
34     public function getComments($slug)
35     {
36          $comments = Comment::with('user')
37              ->where('slug', $slug)
38              ->latest()
39              ->get();
40
41          return response()->json($comments);
42     }
43
44     // Optional: delete comment by id
45     public function destroy($id)
46     {
47          $comment = Comment::findOrFail($id);
48
49          // Only the owner can delete
50          if ($comment->user_id !== Auth::id()) {
51              return response()->json(['error' => '
Forbidden'], 403);
52          }
53
54          $comment->delete();
55          return response()->json(null, 204);
56     }
```

```
}
```

Kode 3.4: CommemtController

Kode tersebut merupakan CommentController di Laravel yang mengatur fungsionalitas komentar pengguna terhadap artikel atau berita. Controller ini menyediakan endpoint RESTful untuk menampilkan semua komentar (index), menyimpan komentar (store), mengambil komentar berdasarkan slug artikel (getComments), dan menghapus komentar (destroy).

Fungsi store() memungkinkan pengguna yang telah login untuk menambahkan komentar ke artikel dengan menyertakan slug artikel dan isi content komentar. Komentar yang dikirimkan otomatis terhubung ke user id pengguna yang sedang login melalui autentikasi. Fungsi getComments() menampilkan daftar komentar yang terkait dengan satu artikel tertentu berdasarkan slug. Untuk keamanan, fungsi destroy() memastikan bahwa hanya pengguna yang membuat komentar tersebut yang dapat menghapusnya. Implementasi ini memberikan sistem komentar yang aman dan interaktif, terintegrasi dengan sistem autentikasi Laravel.

```
class ForgotPasswordController extends Controller
{
    public function sendResetLinkEmail(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'email' => 'required|email|exists:users,email',
        ]);

        if ($validator->fails()) {
            return response()->json([
                'message' => 'Validation failed',
                'errors'  => $validator->errors()
            ], 422);
        }

        $status = Password::sendResetLink(
            $request->only('email')
```

```
        );

        return $status === Password::RESET_LINK_SENT
            ? response()->json(['message' => 'Reset link sent to your email.'])
            : response()->json(['message' => 'Unable to send reset link.'], 500);
    }
}
```

Kode 3.5: ForgotPasswordController

```
class ResetPasswordController extends Controller
{
    public function reset(Request $request)
    {
        $request->validate([
            'token' => 'required',
            'email' => 'required|email',
            'password' => 'required|string|min:8|confirmed',
        ]);

        $status = Password::reset(
            $request->only('email', 'password', 'password_confirmation', 'token'),
            function ($user, $password) {
                $user->forceFill([
                    'password' => Hash::make($password),
                    'remember_token' => Str::random(60),
                ])->save();
            }
        );

        return $status === Password::PASSWORD_RESET
```

```
22                 ? response()->json(['message' => __($status
    )])
23                 : response()->json(['message' => __($status
    )], 400);
24             }
25 }
```

Kode 3.6: ResetPasswordController

Kedua kode tersebut merupakan implementasi fitur lupa kata sandi (forgot password) dan reset kata sandi dalam framework Laravel menggunakan controller ForgotPasswordController dan ResetPasswordController. Fitur ini memungkinkan pengguna yang lupa kata sandi untuk mendapatkan tautan reset melalui email, serta mengatur ulang kata sandinya melalui endpoint yang ditentukan.

Pada ForgotPasswordController, fungsi sendResetLinkEmail() memvalidasi alamat email yang dikirim pengguna, lalu menggunakan fitur built-in Laravel (Password::sendResetLink) untuk mengirim email tautan reset jika email tersebut valid dan terdaftar. Sementara itu, pada ResetPasswordController, fungsi reset() memproses permintaan pengaturan ulang kata sandi berdasarkan token reset yang diterima pengguna. Kata sandi baru akan disimpan secara aman menggunakan hash, dan remember token diperbarui untuk menjaga sesi pengguna tetap aman. Fitur ini merupakan bagian penting dari sistem autentikasi untuk meningkatkan kenyamanan dan keamanan pengguna aplikasi.

3.3.6 Pengembangan Frontend

Pada tahap pengembangan frontend, saya mengimplementasikan desain UI/UX yang telah disusun sebelumnya menggunakan ReactJS dan Tailwind CSS untuk memastikan tampilan yang responsif, modern, dan sesuai dengan kebutuhan pengguna. Komponen-komponen antarmuka disusun secara modular untuk memudahkan pengelolaan dan pengembangan fitur secara berkelanjutan.

Saya membangun struktur navigasi serta beberapa halaman utama seperti halaman beranda, detail berita, dan kategori berita. Selanjutnya, frontend dihubungkan dengan backend melalui pemanggilan RESTful API yang telah dikembangkan, sehingga data berita dan pengguna dapat ditampilkan dan dikelola secara dinamis. Saya juga menambahkan fitur pencarian berita dan kolom komentar untuk meningkatkan interaksi pengguna dengan konten yang tersedia di dalam aplikasi.

3.3.7 Pengujian dan Debuging

Dalam tahap pengujian dan debugging, saya melakukan pengujian backend menggunakan Postman untuk memastikan setiap endpoint API berjalan sesuai dengan fungsinya, termasuk pengujian autentikasi, CRUD data, dan respons dari server. Pengujian ini penting untuk menjamin stabilitas dan keandalan sistem sebelum diintegrasikan secara penuh dengan frontend.

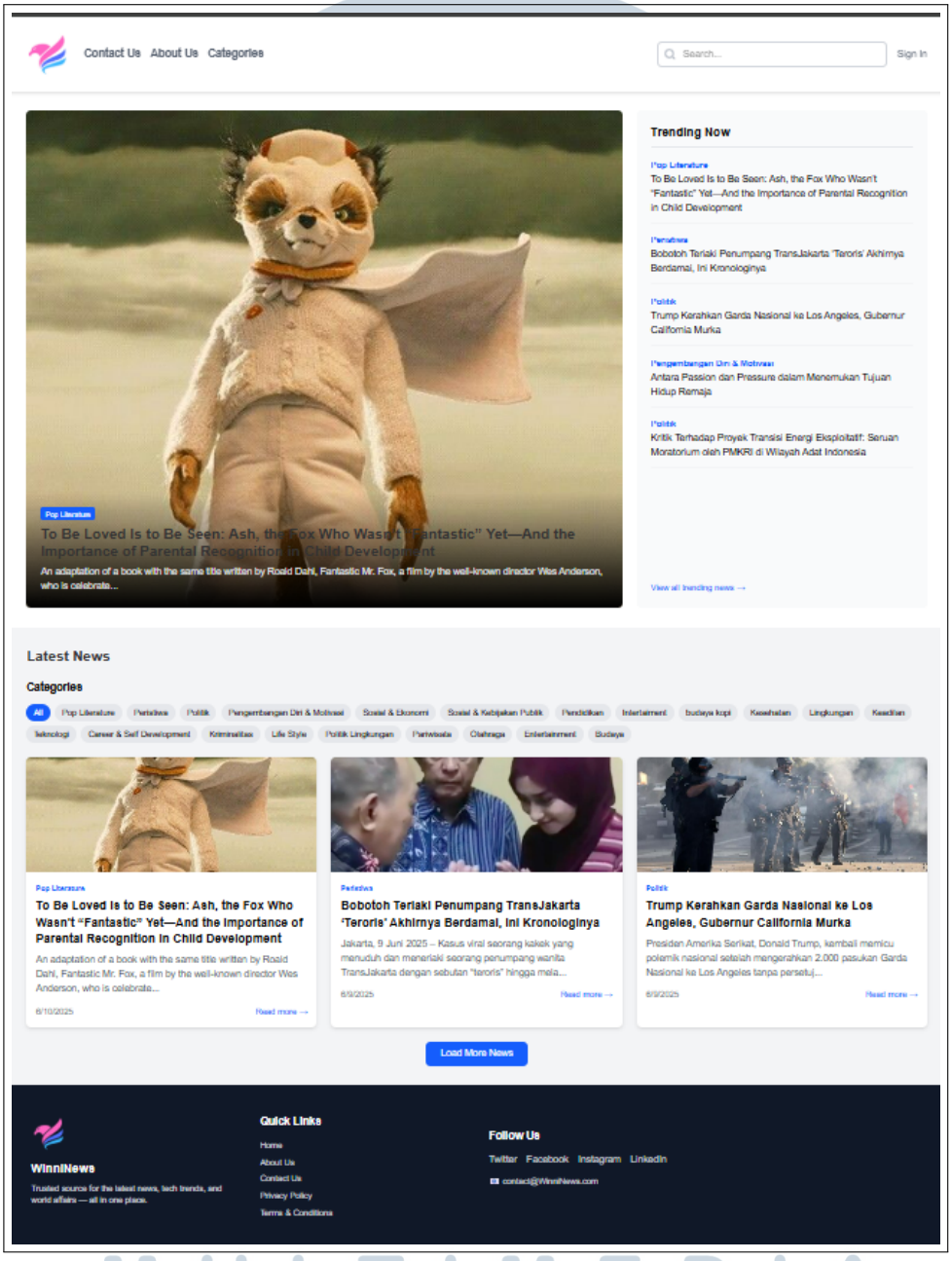
Selain itu, saya juga melakukan pengujian antarmuka pengguna, mencakup uji responsivitas pada berbagai ukuran layar serta pengujian lintas browser (cross-browser testing) guna memastikan tampilan dan fungsi aplikasi tetap konsisten di berbagai perangkat dan platform. Proses ini disertai dengan identifikasi dan perbaikan bug yang ditemukan, serta optimasi performa sistem agar aplikasi berjalan lebih cepat, efisien, dan ramah pengguna.

3.3.8 Finalisasi

Pada tahap finalisasi, saya melakukan monitoring terhadap performa sistem secara menyeluruh untuk memastikan seluruh fitur berjalan dengan baik tanpa kendala berarti. Proses ini mencakup pengujian akhir pada alur pengguna, validasi data, kecepatan respons aplikasi, serta kestabilan integrasi antara frontend dan backend.

Berdasarkan hasil monitoring, saya melakukan optimasi akhir baik dari sisi tampilan maupun performa teknis sistem, seperti perbaikan kecil pada UI, peringanan beban pemrosesan data, dan penyempurnaan query database. Tahap ini memastikan aplikasi siap untuk digunakan oleh pengguna akhir dengan kualitas yang optimal.

3.3.9 Implementasi



Gambar 3.7. Landing Page

Landing page pada gambar 3.7 merupakan halaman utama dari sebuah portal berita bernama WinniNews, yang dirancang untuk menyajikan informasi terkini dalam berbagai kategori. Di bagian paling atas terdapat navigasi utama dengan menu seperti Contact Us, About Us, Categories, kolom pencarian, dan tombol Sign in untuk pengguna.

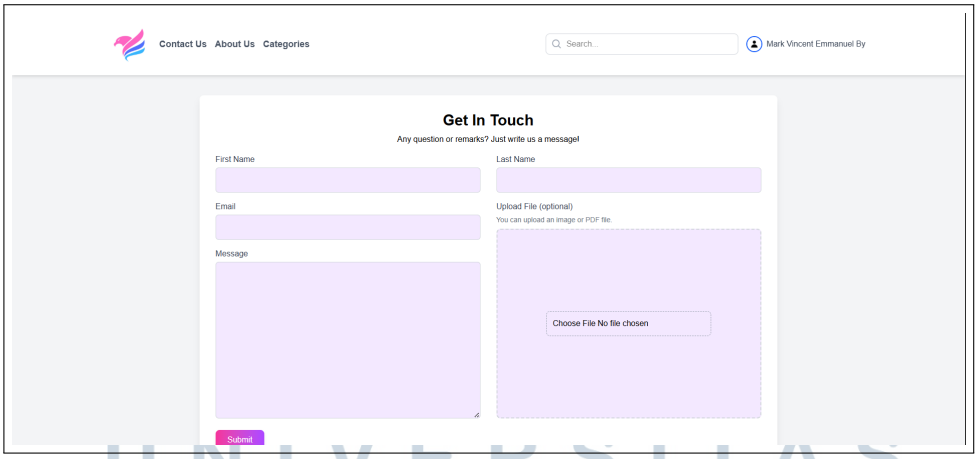
Bagian utama halaman menampilkan berita unggulan dengan visual yang mencolok (hero section), diikuti oleh daftar berita yang sedang trending di sisi kanan. Di bawahnya terdapat segmen Latest News yang menyajikan berita terbaru, dilengkapi filter kategori seperti Politik, Teknologi, Pendidikan, hingga Kesehatan. Setiap berita ditampilkan dengan gambar thumbnail, judul, ringkasan, kategori, dan tanggal publikasi. Pada bagian paling bawah terdapat footer yang berisi informasi tentang WinniNews, tautan cepat, dan kontak media sosial. Tampilan ini bersifat responsif dan mengutamakan kemudahan akses informasi bagi pengguna.

Gambar 3.8. Detail Page

Halaman ini menampilkan detail berita. Pada bagian atas terdapat navigasi utama dengan menu Contact Us, About Us, Categories, kolom pencarian, serta informasi pengguna yang sedang login. Di bawahnya, terdapat judul artikel berita yang ditampilkan secara menonjol, lengkap dengan informasi kategori, tanggal terbit, dan nama penulis.

Isi artikel disajikan secara lengkap dengan gambar utama di bagian atas sebagai ilustrasi, lalu diikuti oleh isi teks berita yang ditulis dalam beberapa paragraf panjang. Artikel ini mengangkat topik tentang kebijakan pendidikan oleh Dedi Mulyadi di Jawa Barat. Di bagian bawah konten utama, tersedia fitur komentar yang memungkinkan pembaca memberikan tanggapan, namun hanya jika sudah login. Jika belum ada komentar, sistem akan menampilkan pesan default bahwa belum ada komentar yang masuk.

Pada bagian paling bawah, halaman ini juga menampilkan artikel terkait (Related Articles) dalam bentuk kartu horizontal yang terdiri dari gambar, judul, dan tautan Read More. Ini bertujuan untuk meningkatkan keterlibatan pengguna dan memperpanjang waktu kunjungan di dalam platform dengan menyarankan bacaan relevan lainnya. Secara keseluruhan, halaman ini dirancang secara informatif dan interaktif dengan fokus pada pengalaman membaca yang nyaman dan terstruktur.



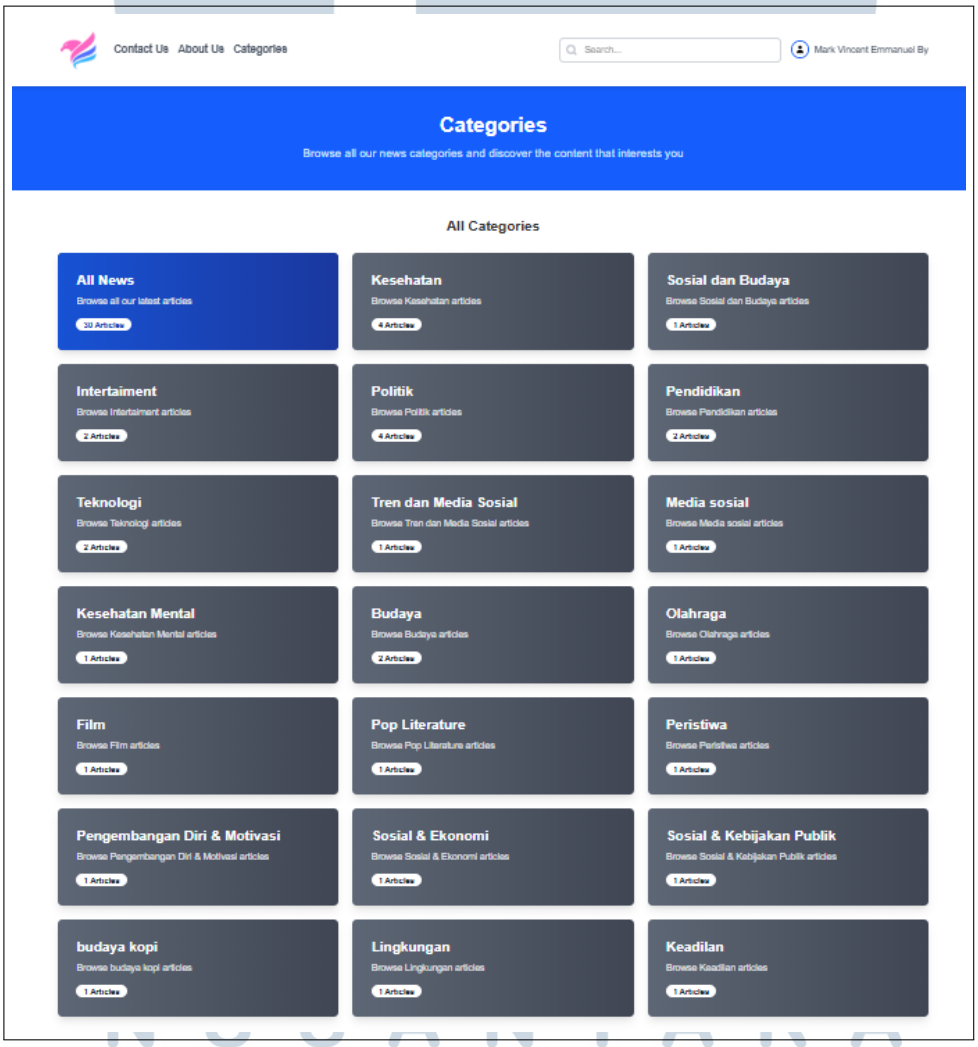
Gambar 3.9. ContacUs Page

Gambar 3.9 menampilkan halaman Contact Us. Halaman ini berfungsi sebagai formulir komunikasi bagi pengunjung yang ingin mengirim pertanyaan, saran, atau pesan lainnya kepada tim pengelola situs. Di bagian atas terdapat navigasi utama yang mencakup tautan ke Contact Us, About Us, Categories, kolom pencarian, dan profil pengguna yang sedang login.

Formulir ini terdiri dari beberapa kolom input, yaitu: First Name, Last

Name, Email, dan Message. Selain itu, terdapat fitur unggahan file (Upload File) yang bersifat opsional, memungkinkan pengguna melampirkan dokumen atau gambar (seperti bukti atau lampiran pendukung lainnya). Teks pendamping di bawah kolom upload memberi tahu pengguna bahwa file yang diperbolehkan berupa gambar atau file PDF.

Desain form disusun dengan tata letak dua kolom yang rapi dan responsif, memberikan pengalaman pengguna yang baik dan mudah dipahami. Setelah form diisi, pengguna dapat mengirim pesannya melalui tombol Submit yang tersedia di bagian bawah. Halaman ini berperan penting dalam membangun komunikasi dua arah antara pengunjung dan pengelola platform.



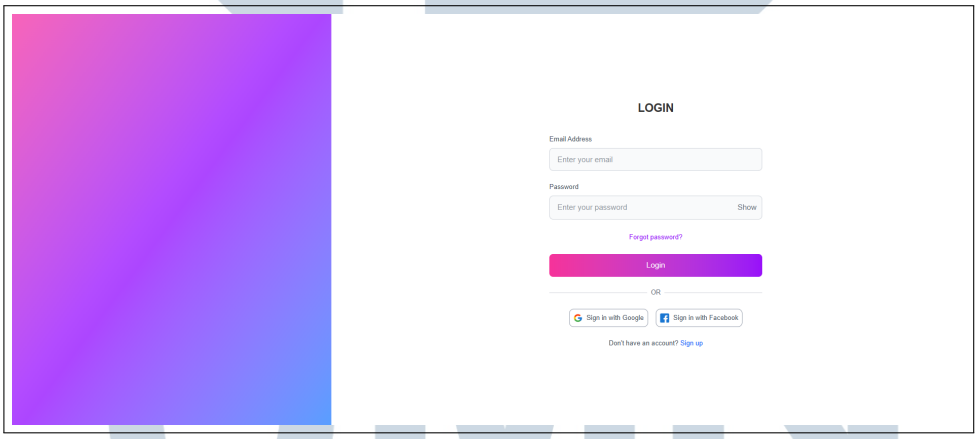
Gambar 3.10. Categories Page

Gambar 3.10 menampilkan halaman kategori berita. Halaman ini dirancang untuk memudahkan pengguna menjelajahi artikel berdasarkan topik atau tema

tertentu yang telah diklasifikasikan. Di bagian atas terdapat header biru mencolok dengan judul ”Categories” serta deskripsi singkat yang mengajak pengguna untuk menemukan konten yang mereka minati.

Setiap kategori ditampilkan dalam bentuk kartu berwarna gelap dengan nama kategori, deskripsi singkat (misalnya: “Browse Kesehatan articles”), dan jumlah artikel yang tersedia dalam kategori tersebut. Terdapat berbagai kategori yang mencakup topik seperti Kesehatan, Sosial dan Budaya, Politik, Teknologi, Pendidikan, Film, Pop Literature, hingga kategori unik seperti Budaya Kopi. Kategori All News berada di posisi teratas dan tampil menonjol dengan warna berbeda, menunjukkan total jumlah artikel secara keseluruhan (30 articles).

Tata letak grid yang digunakan memungkinkan pengguna untuk dengan cepat memindai dan memilih kategori yang sesuai dengan minat mereka. Desain halaman ini tidak hanya memperkaya pengalaman pengguna, tetapi juga meningkatkan navigasi dan keterlibatan dengan konten yang lebih relevan secara tematik.

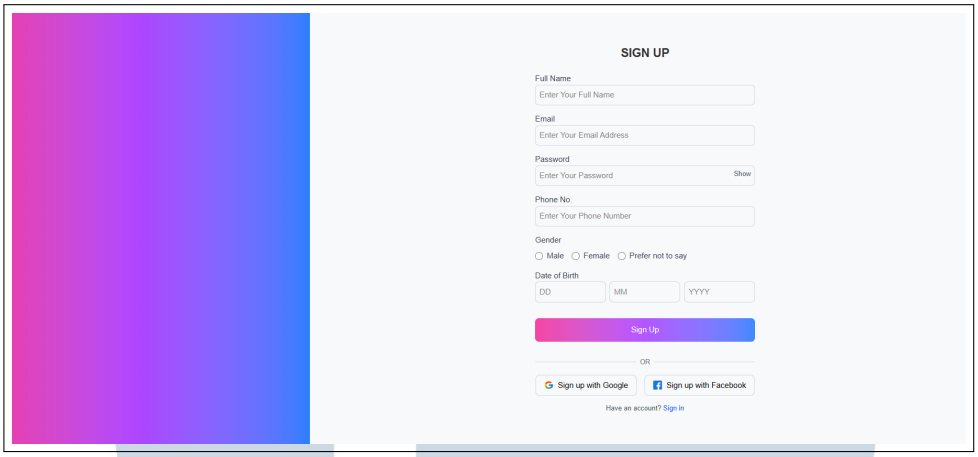


Gambar 3.11. Login Page

Gambar 3.11 merupakan halaman Login dari sebuah aplikasi web modern. Desainnya minimalis dan elegan, memanfaatkan gradasi warna ungu dan pink yang mencolok di sisi kiri, memberikan kesan profesional sekaligus menarik secara visual. Di sisi kanan, pengguna dapat mengisi formulir login yang terdiri dari dua input: alamat email dan kata sandi. Terdapat juga fitur “Show” untuk menampilkan atau menyembunyikan sandi serta tautan “Forgot password?” untuk membantu pengguna yang lupa kata sandi.

Selain itu terdapat link untuk sign up bagi pengguna baru yang belum memiliki akun. Penempatan elemen yang rapi serta pilihan warna yang kontras membuat halaman ini sangat mudah digunakan dan ramah bagi pengguna dari

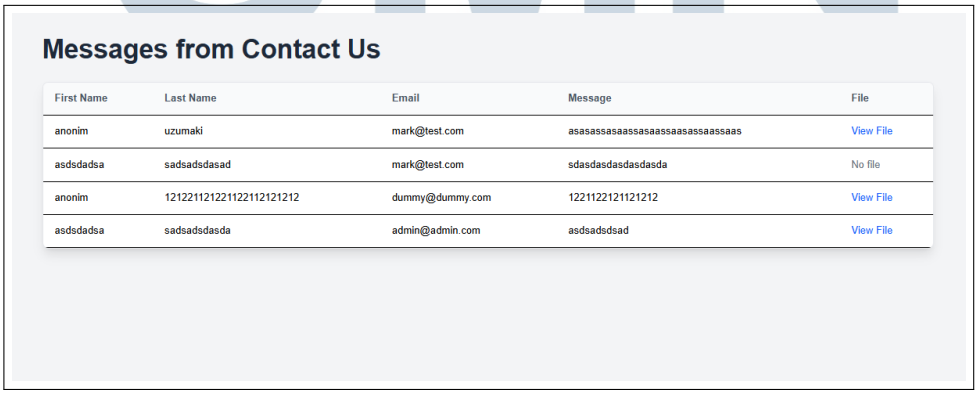
berbagai latar belakang.



Gambar 3.12. Register Page

Gambar 3.12 merupakan halaman Register dari sebuah aplikasi web modern yang didesain dengan tampilan minimalis dan elegan. Gradasi warna ungu dan biru di sisi kiri memberikan kesan visual yang menarik sekaligus profesional. Di sisi kanan halaman, pengguna disajikan formulir pendaftaran yang lengkap, terdiri dari beberapa input seperti nama lengkap, email, kata sandi, nomor telepon, jenis kelamin, serta tanggal lahir.

Formulir ini juga dilengkapi dengan fitur “Show” untuk menampilkan atau menyembunyikan kata sandi, sehingga mempermudah pengguna dalam menghindari kesalahan pengetikan. Selain itu, Penataan elemen yang terstruktur dengan baik, penggunaan warna yang kontras, serta navigasi yang jelas membuat halaman ini sangat ramah pengguna dan mudah digunakan, baik oleh pengguna baru maupun pengguna berpengalaman.



Gambar 3.13. Login Page

Gambar 3.13 merupakan halaman yang digunakan untuk menampilkan

pesan yang dikirimkan oleh pengguna melalui halaman Contact Us. Desain halaman ini sederhana dan terstruktur dengan baik, menampilkan data dalam bentuk tabel yang mudah dibaca. Setiap baris pada tabel berisi informasi seperti nama depan, nama belakang, email, pesan yang dikirim, serta tautan untuk melihat file yang mungkin dilampirkan.

Kolom ”File” memberikan informasi apakah pengguna menyertakan file atau tidak, dengan opsi “View File” untuk melihat berkas yang diunggah. Tabel ini mempermudah admin dalam memantau dan merespons pesan yang masuk secara efisien. Penataan elemen yang rapi serta pemisahan data berdasarkan kolom menjadikan halaman ini fungsional dan mudah dipahami.

3.4 Kendala dan Solusi yang Ditemukan

Selama melaksanakan praktik kerja magang di PT WINNICODE GARUDA TEKNOLOGI, terdapat beberapa kendala teknis dan non-teknis yang saya hadapi dalam proses pengembangan sistem. Namun, dengan bimbingan dari supervisor dan pembelajaran mandiri, setiap kendala tersebut dapat diselesaikan dengan solusi yang tepat. Berikut adalah beberapa kendala yang ditemui beserta solusinya:

1. **Kendala dalam Integrasi API Eksternal**
   Pada saat mengintegrasikan API eksternal untuk mengambil data berita dari situs Winnicode, saya mengalami kesulitan dalam memahami dokumentasi API dan format autentikasi yang digunakan.

   Solusi: Saya mempelajari dokumentasi API secara mendalam, melakukan eksperimen melalui Postman, dan berdiskusi dengan supervisor untuk memahami skema otentikasi *Bearer Token* serta struktur respons API.

2. **Masalah Validasi Data dan Upload File**
   Saat membuat fitur registrasi pengguna dan formulir kontak, terjadi error ketika pengguna mengunggah file berukuran besar atau tidak sesuai format.

   Solusi: Saya memperketat proses validasi di sisi backend menggunakan Laravel Validator, membatasi ukuran file maksimal, serta menambahkan filter tipe MIME yang diperbolehkan. Selain itu, saya juga menambahkan notifikasi yang informatif di sisi frontend.

3. **Penggunaan Laravel Sanctum untuk Autentikasi**
   Penggunaan Laravel Sanctum untuk manajemen token autentikasi cukup

membingungkan pada awalnya, terutama saat mengatur middleware dan token session.

Solusi: Saya membaca dokumentasi resmi Laravel serta mengikuti tutorial dari sumber terpercaya. Selain itu, saya juga melakukan debugging menggunakan Postman untuk memastikan alur login dan token berhasil dikembalikan dan digunakan dalam setiap permintaan.

4. **Pengelolaan Waktu dan Penyesuaian dengan Target**
   Beberapa kali terjadi kendala dalam membagi waktu antara belajar mandiri, tanggung jawab magang, dan pengerjaan proyek yang cukup kompleks.

   Solusi: Saya membuat jadwal kerja harian dan menetapkan prioritas tugas. Saya juga melakukan diskusi rutin dengan pembimbing lapangan untuk memastikan progress tetap sesuai dengan target yang ditentukan.

Dengan menghadapi berbagai kendala tersebut, saya memperoleh pengalaman langsung dalam problem solving di dunia kerja serta meningkatkan kemampuan teknis maupun komunikasi secara signifikan.