

## **BAB IV**

### **HASIL DAN PEMBAHASAN**

#### **4.1 Tahapan Requirement Planning**

Pada tahap ini merupakan tahap untuk melakukan proses pengambilan data oleh peneliti untuk melakukan perancangan sistem pencarian destinasi wisata dan UMKM menggunakan algoritma Dijkstra di kecamatan Kemiling.

##### **4.1.1 Analisis Masalah**

Berdasarkan analisis masalah pada bab sebelumnya, ditemukan bahwa potensi wisata dan UMKM di Kecamatan Kemiling terhambat oleh keterbatasan informasi yang terfragmentasi dan promosi yang tidak efektif. Untuk mengatasi hal tersebut, telah dirancang dan dibangun sebuah solusi digital berupa aplikasi Android terintegrasi. Aplikasi ini secara langsung menjawab permasalahan dengan menyediakan platform terpusat di mana wisatawan dapat dengan mudah menemukan informasi deskripsi, lokasi, dan harga dari berbagai destinasi dan produk UMKM. Dengan demikian, aplikasi ini menjembatani kesenjangan informasi antara wisatawan dan pelaku usaha, meningkatkan visibilitas potensi lokal, dan menyediakan sarana digital untuk perencanaan perjalanan yang lebih efisien.

##### **4.1.2 Analisis Kebutuhan Sistem**

Berdasarkan analisis masalah yang dipaparkan diatas, dapat disimpulkan bahwa sistem pencarian destinasi wisata dan umkm yang ingin di bangun antara lain :

###### **a. Sisi client (Client-Side)**

Aplikasi dapat berjalan pada Smartphone Android dengan spesifikasi minimum RAM 2GB, memiliki fitur GPS, dan Memiliki Internet. Aplikasi akan dikembangkan menggunakan Android Studio dengan bahasa pemrograman Java, untuk fungsionalitas peta dan komunikasi data.

###### **b. Sisi Server (Server-Side)**

Sistem akan di hosting pada layanan web server online yang akan dipakai sebagai API (Application Programming Interface) yang menjadi jembatan antara aplikasi dan database yang di bangun menggunakan Bahasa pemrograman PHP dan sistem manajemen database MySQL.

#### **4.2 Tahapan Perancangan Sistem**

Pada tahap Dimana pembuat atau pengembang proyek dapat menjabarkan secara rinci apa saja yang akan di lakukan, dan mendesain struktur aplikasi. Mulai dari :

1. Pembuatan use case diagram

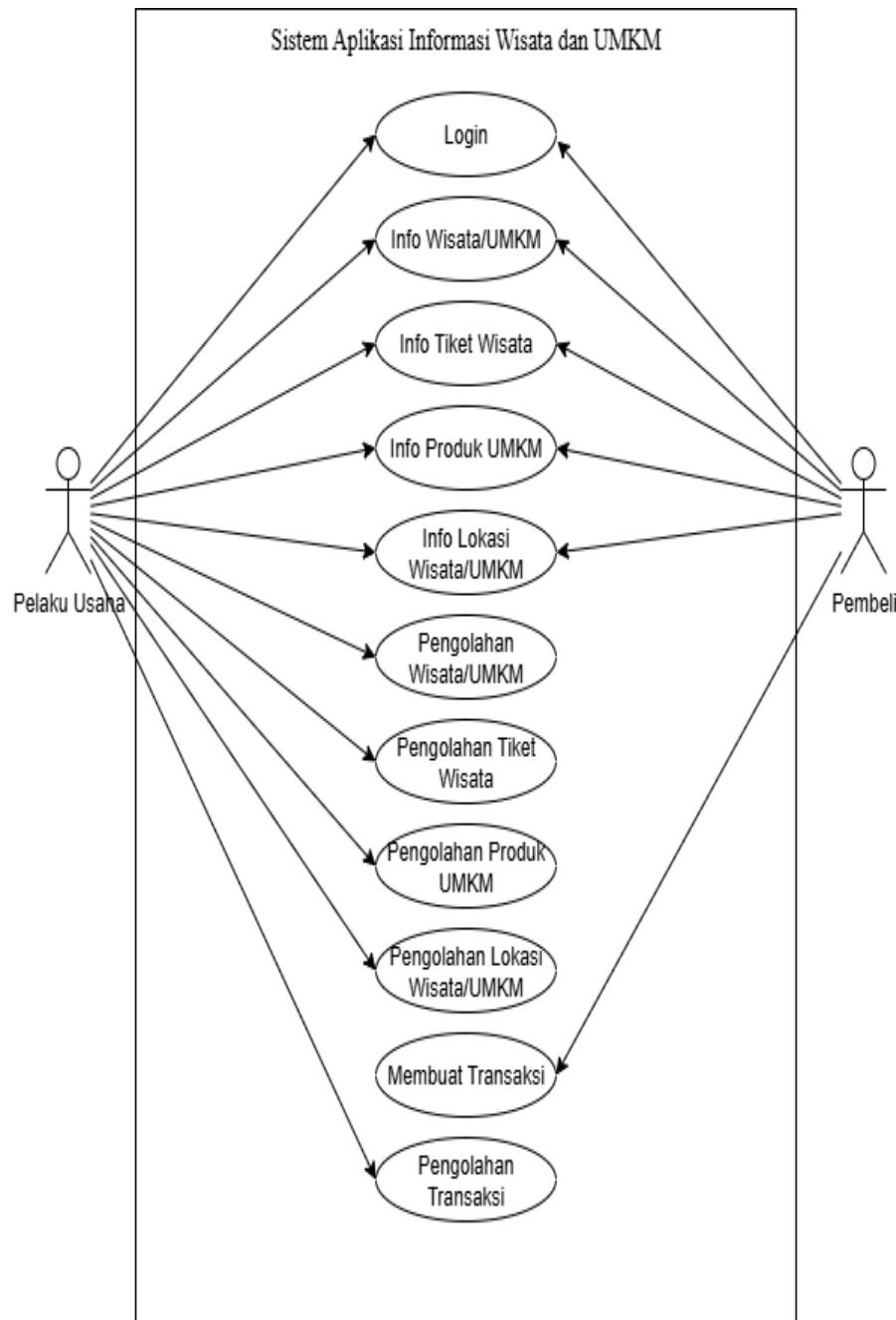
Pada tahap ini pembuatan use case diagram meliputi :

Perancangan Use case

**Tabel 4.2 Use Case**

Aktor	Deskripsi
Pengguna Umum (Wisatawan)	Aktor ini merupakan pengguna utama yang perannya sebagai konsumen. Pengguna dapat melakukan registrasi, login, mencari informasi wisata dan UMKM, melihat detail lokasi, serta mengelola profil pribadi nya.
Pelaku Usaha	Aktor ini adalah pengguna yang memiliki dan mengelola data destinasi wisata atau UMKM. Setelah melakukan login, Pelaku Usaha memiliki hak akses untuk menambah, mengubah, dan menghapus data yang terkait dengan usaha miliknya, seperti informasi usaha, produk, dan tiket.

Berdasarkan use case description diatas, dapat di lihat pada gambar 4.2 dibawah ini.



Gambar 4.1 Use Case

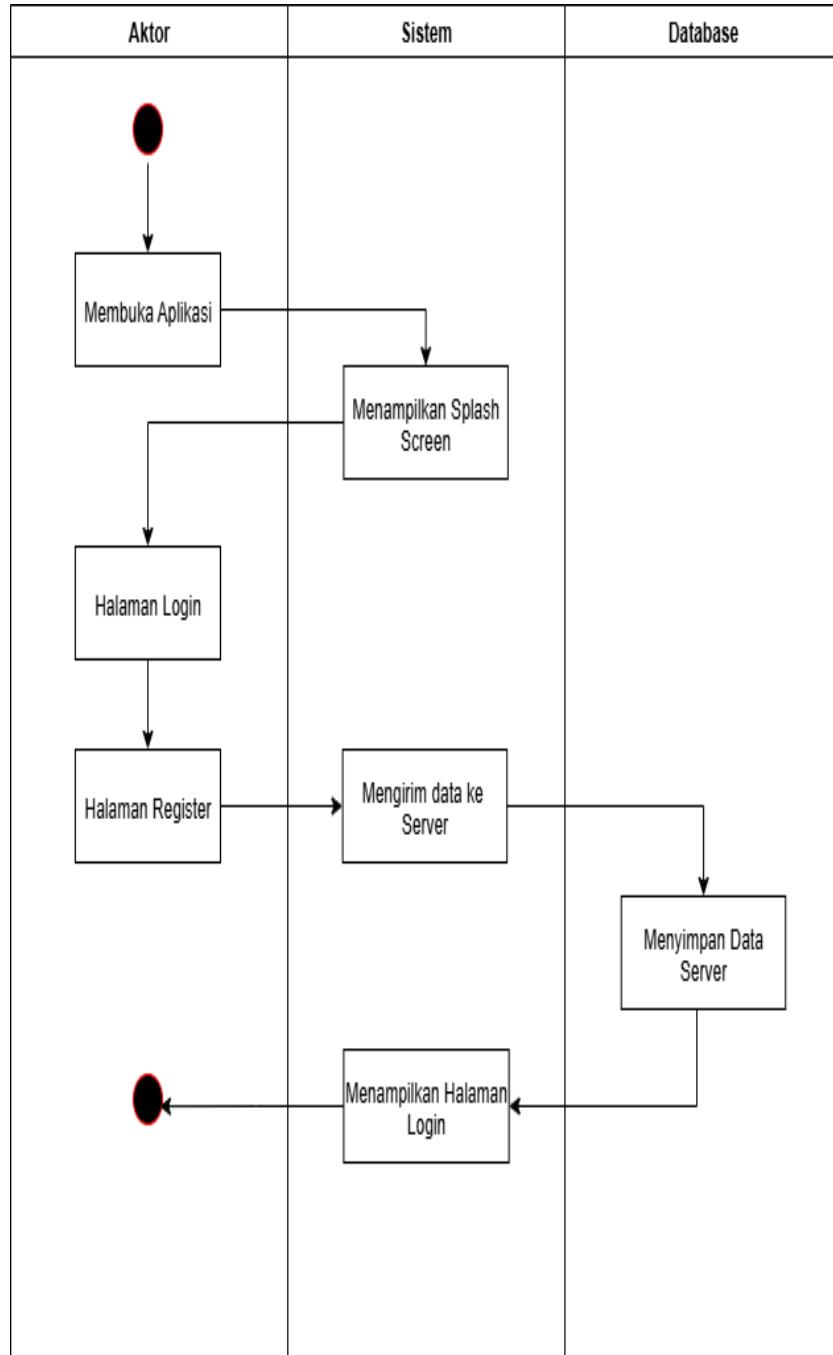
## 2. Pembuatan Activity diagram

Pada tahap ini peneliti membuat alur proses aktivitas sistem dan merencanakan setiap alur Activity Diagram.

### 1. Activity Diagram Registrasi

Diagram aktivitas ini merangkum alur kerja pengguna, mulai dari membuka aplikasi dan melakukan pendaftaran akun. Sehingga

setelah melakukan pendaftaran akun, pengguna bisa langsung login dengan akun yang sudah di daftarkan.

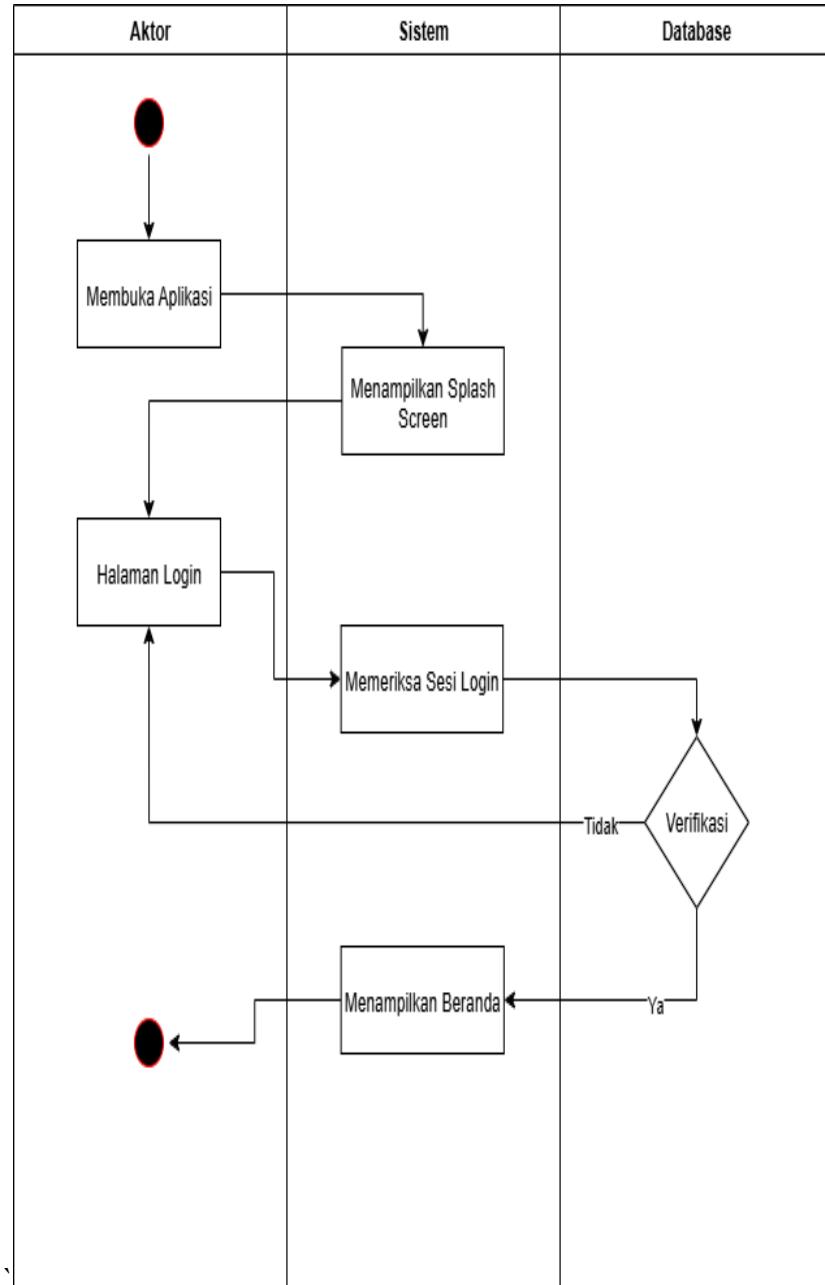


*Gambar 4.2 Activity Diagram Registrasi*

## 2. Activity Diagram Login

Diagram aktivitas ini merangkum alur kerja pengguna, mulai dari membuka aplikasi dan melakukan login dengan akun yang sudah

terdaftar. Sehingga setelah melakukan login akun, pengguna bisa langsung membuka aplikasi dan menampilkan halaman beranda.

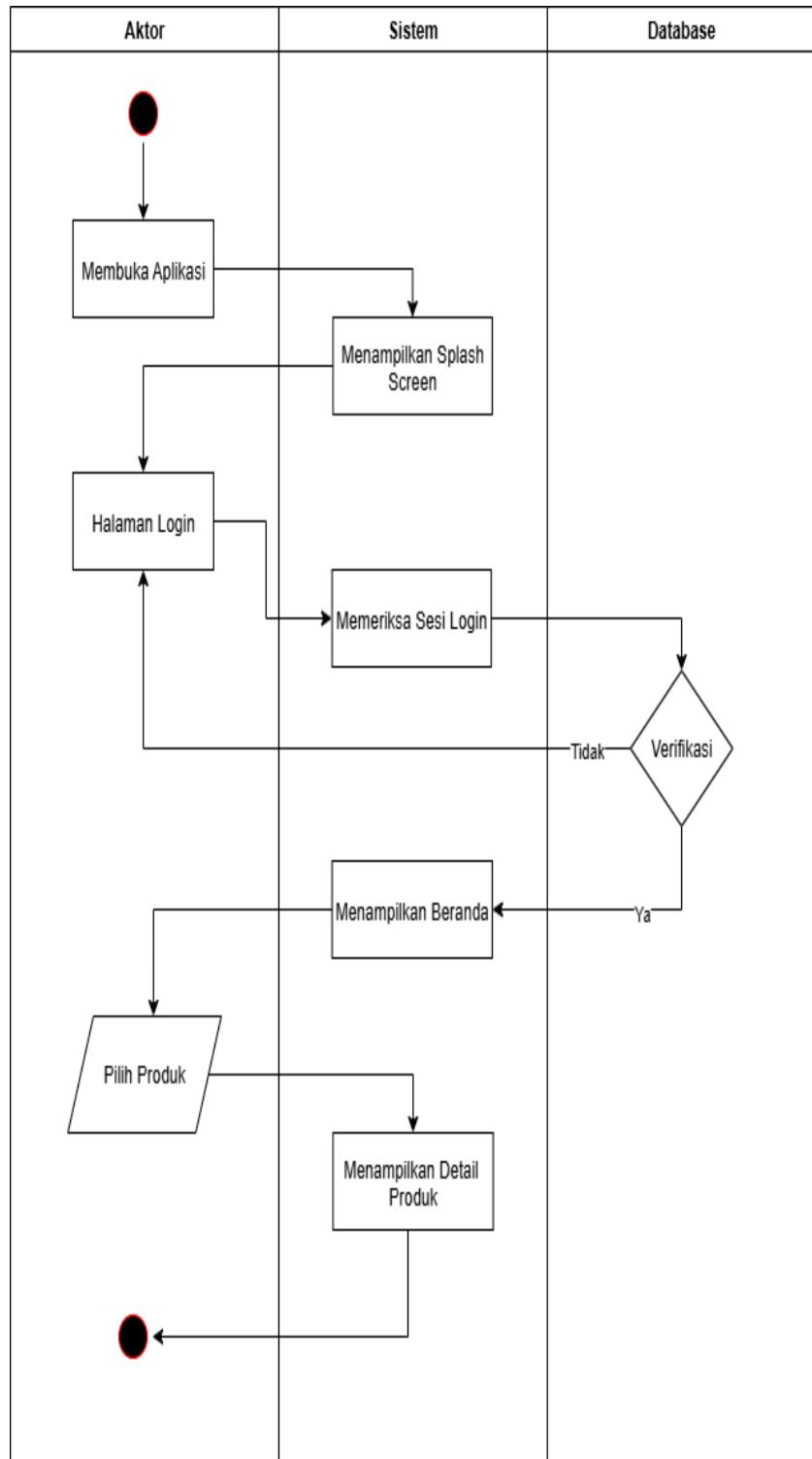


Gambar 4.3 Activity Diagram Login

### 3. Activity Detail Produk

Diagram aktivitas ini menggambarkan alur kerja pengguna untuk melihat detail dalam produk yang mereka pilih, yang akan

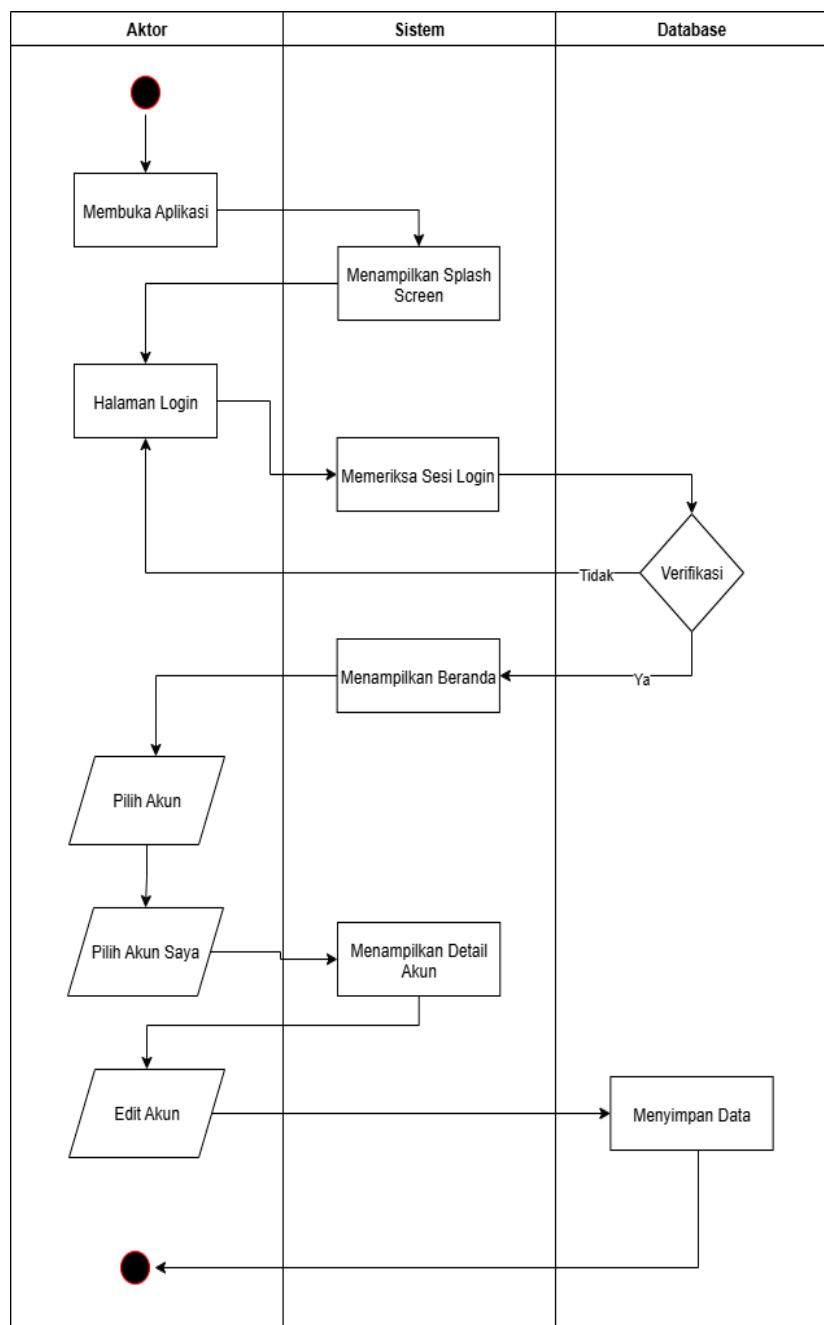
menampilkan deskripsi, gambar, dan rute yang akan di tempuh ke tempat lokasi produk tersebut.



*Gambar 4.4 Activity Detail Produk*

#### 4. Activity Diagram Edit Profil

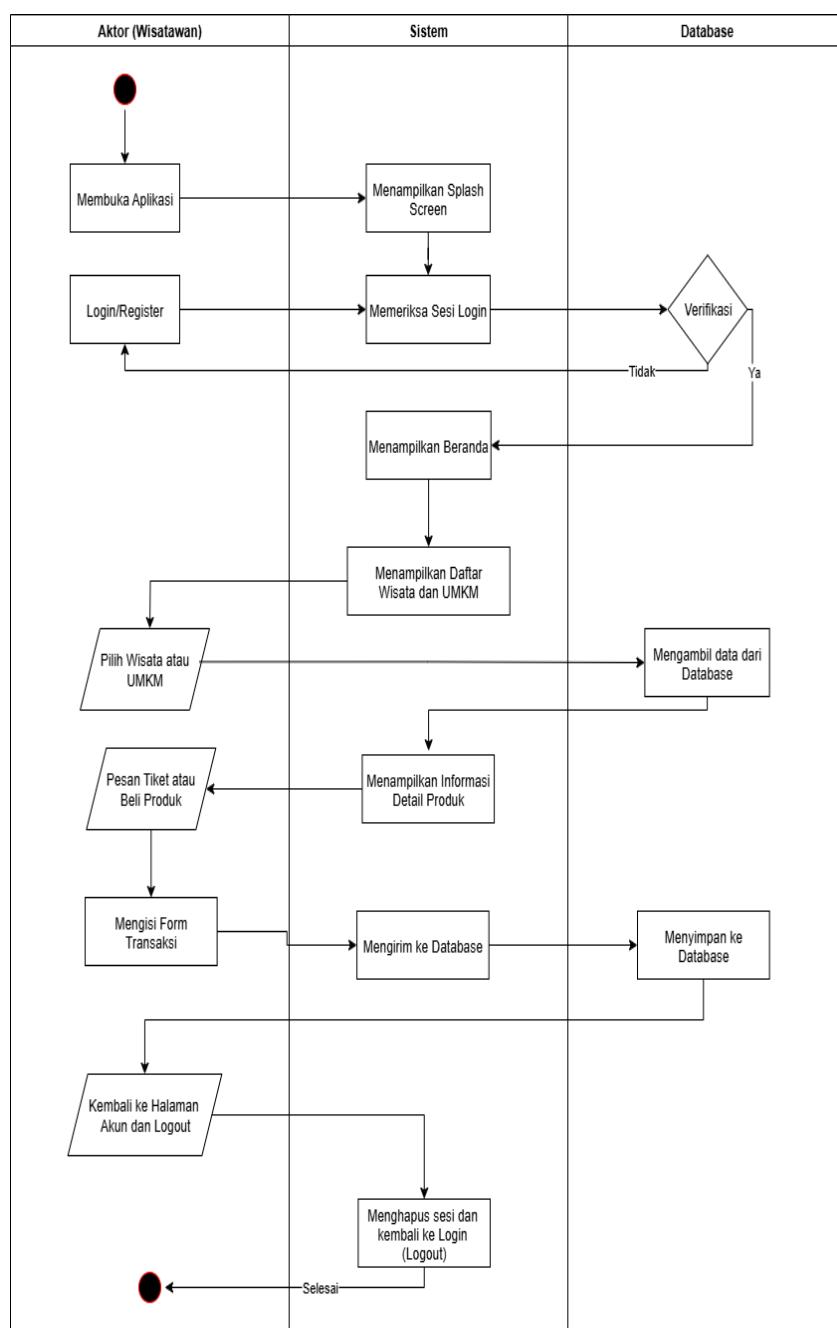
Diagram aktivitas ini menjelaskan alur kerja Pengguna, yang setelah login dapat mengelola profil miliknya di halaman "Akun saya". Fungsi utamanya adalah menyimpan data akun dengan mengisi form, di mana data tersebut akan diproses dan disimpan ke database.



*Gambar 4.5 Activity Diagram Edit Profil*

## 5. Activity Diagram Transaksi Wisatawan

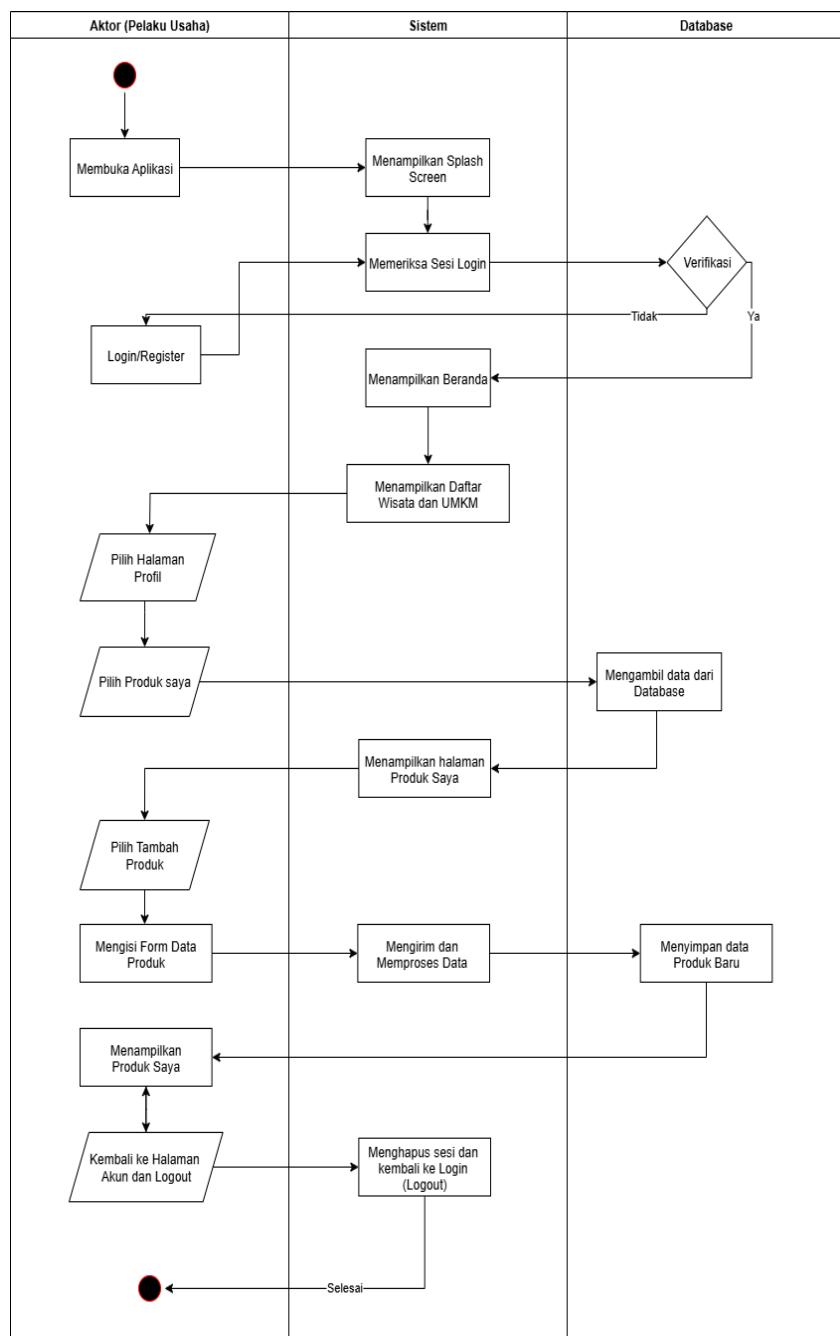
Diagram aktivitas ini menggambarkan alur kerja pengguna (wisatawan) ketika melakukan proses transaksi, seperti pemesanan tiket atau pembelian produk, melalui sistem aplikasi. Alur ini menunjukkan bagaimana sistem menangani dan merekam setiap transaksi yang dilakukan oleh pengguna.



Gambar 4.6 Activity Diagram Transaksi

## 6. Activity Diagram Tambah Produk

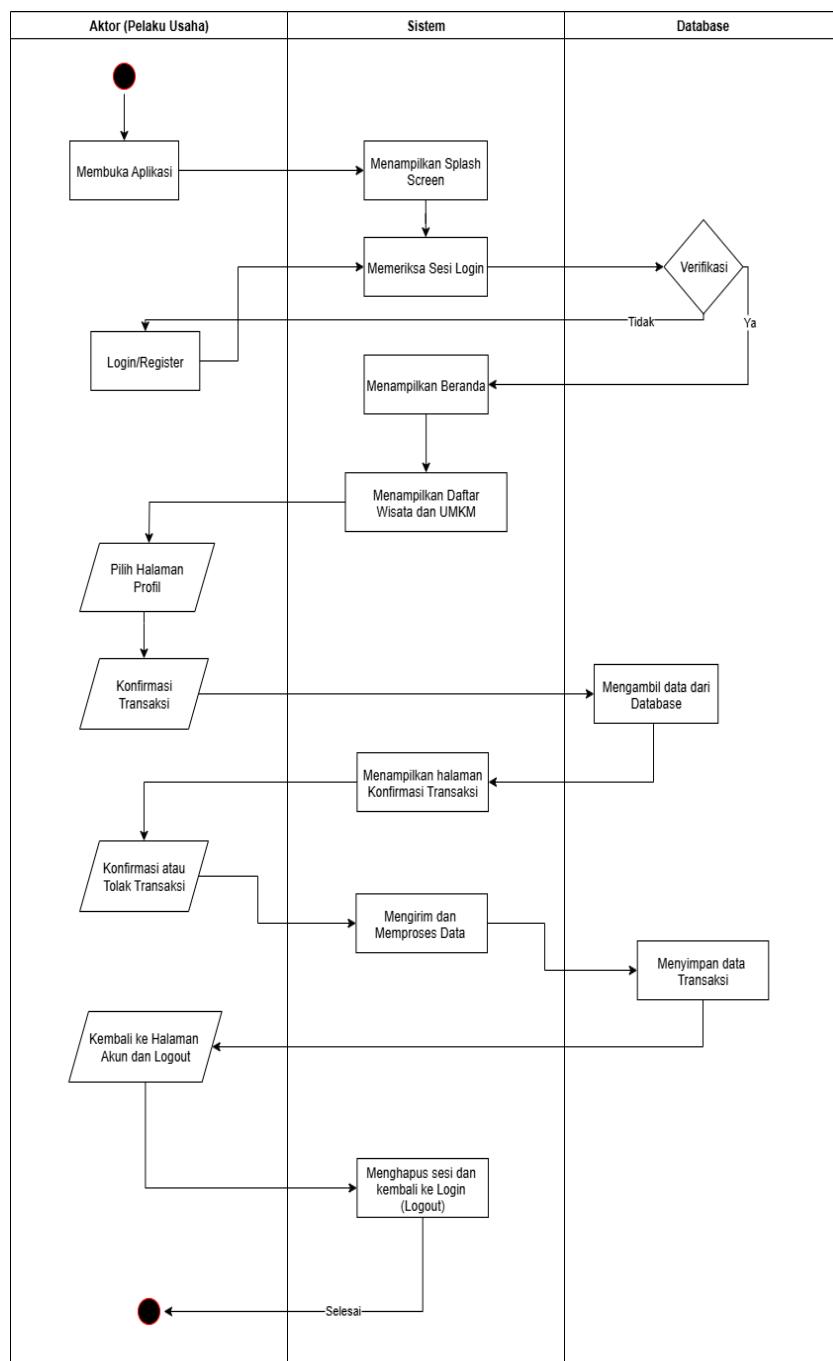
Diagram aktivitas ini menjelaskan alur kerja Pelaku Usaha, yang setelah login dapat mengelola konten miliknya di halaman "Produk Saya". Fungsi utamanya adalah menambah produk baru dengan mengisi form, di mana data tersebut akan diproses dan disimpan ke database.



Gambar 4.7 Activity Diagram Tambah Produk

## 7. Activity Diagram Konfirmasi Transaksi

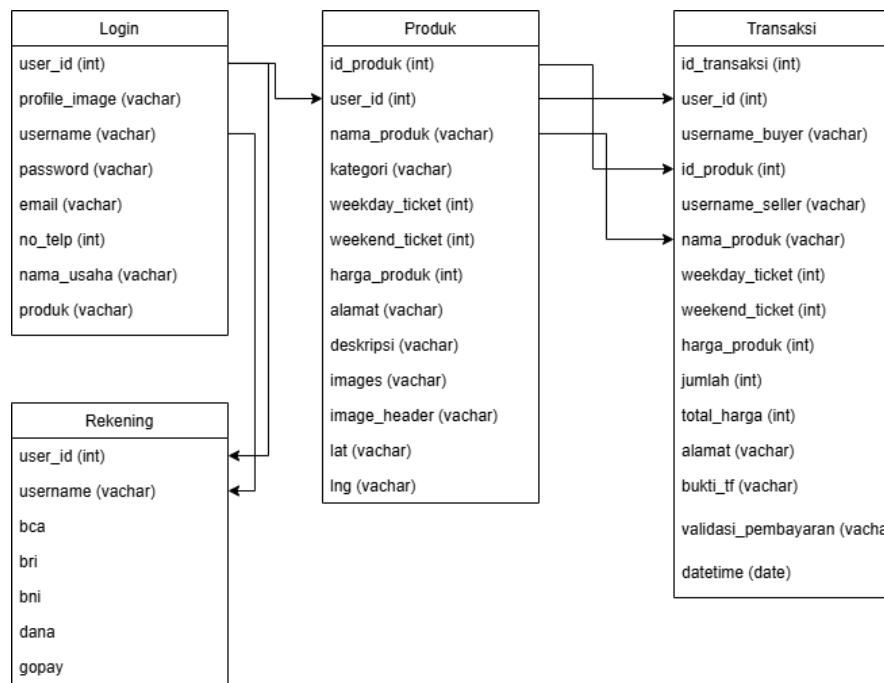
Diagram aktivitas ini menguraikan alur kerja bagi Pelaku Usaha ketika menerima dan mengonfirmasi pesanan atau tiket yang telah dibuat oleh wisatawan. Alur ini memastikan bahwa ada proses validasi dari sisi penyedia jasa sebelum sebuah transaksi dianggap selesai, dan memberikan kepastian bagi wisatawan.



Gambar 4.8 Konfirmasi Transaksi

### 3. Pembuatan Class Diagram

Class diagram ini menggambarkan struktur dasar dari sistem aplikasi dengan tiga komponen. Dengan class diagram ini menunjukkan arsitektur sistem yang modular, Dimana setiap class memiliki tanggung jawab yang spesifik.



Gambar 4.9 Class Diagram

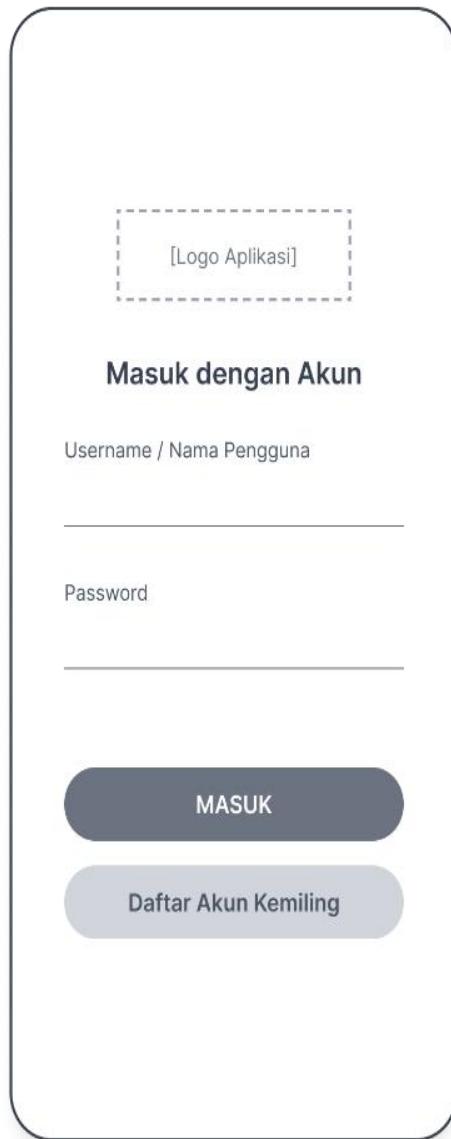
## 4.3 Tahapan Pembangunan dan Prototipe

Pada tahap ini, rancangan yang sudah di buat dapat di wujudkan dalam bentuk Aplikasi. Pada pembuatan aplikasi platform sistem terintegrasi berbasis mobile ini menggunakan android studio. Tahapan ini melakukan pengembangan prototipe awal berdasarkan desain yang sudah dibuat. Berikut ini adalah adalah desain awal dari aplikasi yang akan dibuat.

### 4.3.1 Pembangunan Desain Awal Aplikasi

#### a. Tampilan Login

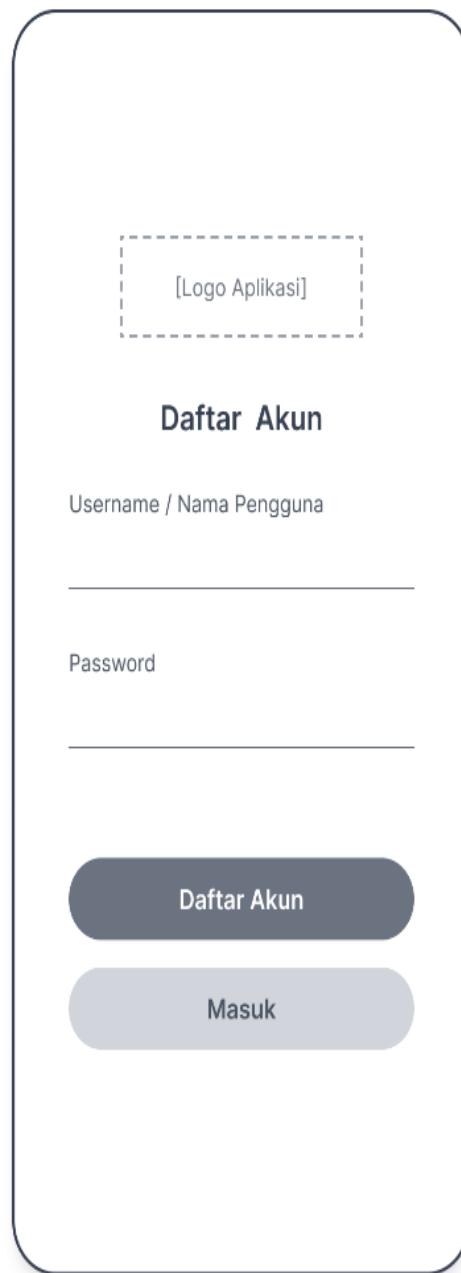
Tampilan ini adalah tampilan awal dari ketika membuka aplikasi, untuk masuk kebagian berikutnya, terdapat masukan data seperti username dan password, dan terdapat tombol untuk masuk dan melakukan proses login.



*Gambar 4.10 Tampilan Login*

b. Tampilan Registrasi

Tampilan ini merupakan bagian untuk dapat menambahkan akun user login konsumen, terdapat masukan data seperti usenname dan password serta terdapat tombol daftar untuk menambahkan akun ke database yang dapat di lihat pada gambar dibawah.



Gambar 4.11 Tampilan Daftar

c. Tampilan Beranda

Tampilan ini merupakan bagian antarmuka sistem yang menampilkan konten produk yang diambil dari database. Produk ini ditambahkan pengguna dari pelaku usaha dengan informasi deskripsi dan lokasi yang lengkap. Konten produk ini meliputi Wisata dan UMKM dan dilengkapi dengan navigasi utama yang terdiri dari “Home, Terdekat,

dan Akun”, untuk memfasilitasi perpindahan cepat antar fitur utama aplikasi.



Gambar 4.12 Tampilan Beranda

d. Tampilan Terdekat

Tampilan ini merupakan rancangan antarmuka yang dirancang untuk memberikan informasi Lokasi wisata dan UMKM berdasarkan

kedekatannya dengan posisi pengguna. Dengan tata letak ini memudahkan pengguna untuk memilih produk atau destinasi wisata terdekat yang ada di sekitar mereka, bisa di lihat pada gambar dibawah.



Gambar 4.13 Tampilan Terdekat

e. Halaman Akun

Halaman ini merupakan tampilan untuk mengelola profil, produk, dan Logout dari sesi. Dengan tata letak minimalis memastikan pengguna

yang dapat dengan mudah mengakses fungsi-fungsi penting terkait akun mereka, dan bilah navigasi bawa dengan ikon.

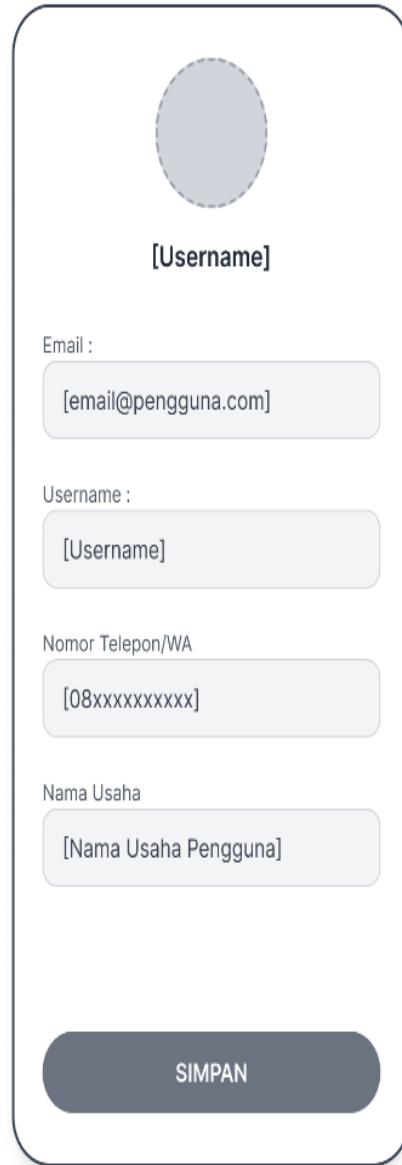


*Gambar 4.14 Tampilan Akun*

f. Tampilan Akun Saya

Tampilan ini merupakan antarmuka yang menampilkan dan mengelola informasi pribadi mereka, yang terdiri dari email, username, nomor telpon, dan nama usaha mereka, dan terdapat tombol simpan yang

berfungsi untuk menyimpan setiap perubahan yang mungkin dilakukan pengguna.



*Gambar 4.15 Tampilan Akun Saya*

g. Tampilan Produk Saya

Pada tampilan ini merupakan rancangan khusus untuk pelaku usaha dalam mengelola produk mereka. Dengan dua tombol sebagai penambah konten produk yang akan di upload, dan juga menampilkan beberapa daftar produk atau wisata yang diambil dari database

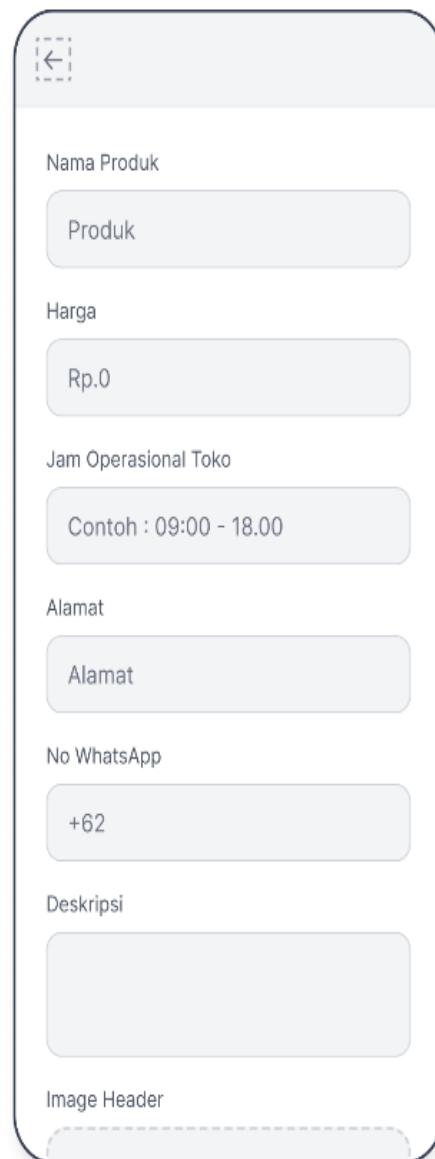
berdasarkan yang pelaku usaha upload. Serta pengguna dapat menghapus produk.



*Gambar 4.16 Tampilan Produk Saya*

h. Tampilan Tambah Produk

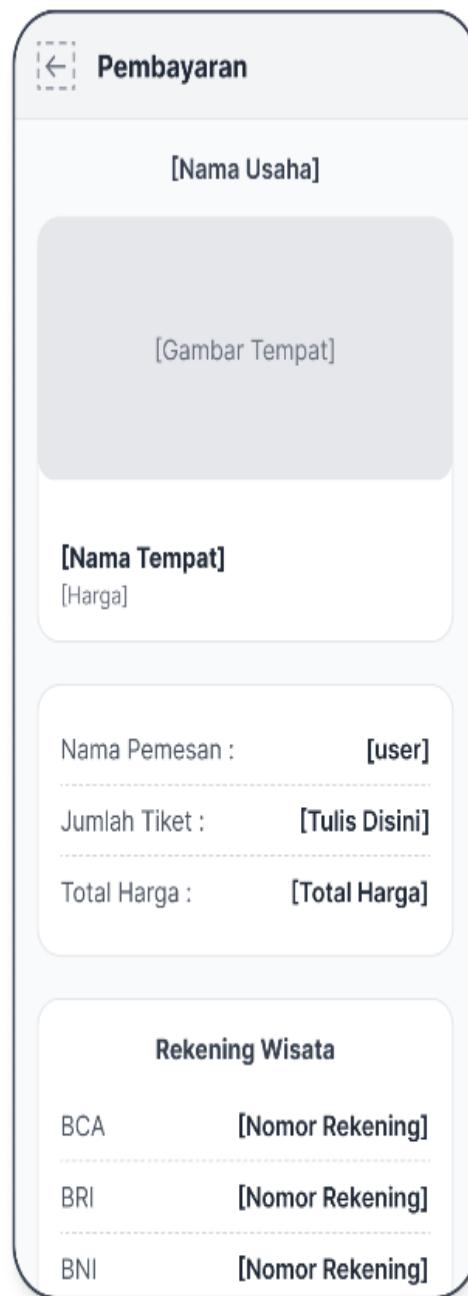
Tampilan ini merupakan sebuah formulir vertikal untuk memasukkan semua detail data yang di perlukan. Dengan desain ini memastikan semua informasi penting dapat di input oleh pelaku usaha secara sistematis dengan satu alur.



Gambar 4.17 Tampilan Tambah Produk

i. Tampilan Pembayaran

Rancangan antarmuka untuk Halaman Pembayaran. Halaman ini dirancang untuk memandu pengguna menyelesaikan proses transaksi setelah memesan tiket atau produk. Dengan desain ini memastikan semua informasi penting dapat di input oleh pengguna secara sistematis dengan satu alur.



Gambar 4.18 Tampilan Pembayaran

j. Tampilan Konfirmasi Transaksi

Gambar tersebut merupakan rancangan antarmuka untuk Halaman Konfirmasi Transaksi, yang dirancang khusus bagi Pelaku Usaha untuk mengelola pesanan yang masuk. Di bagian bawah setiap kartu, terdapat dua tombol aksi utama, yaitu "Konfirmasi" untuk menyetujui dan

"Tolak" untuk membatalkan pesanan, yang memungkinkan Pelaku Usaha untuk mengelola alur transaksi secara efisien.



Gambar 4.19 Tampilan Konfirmasi

#### 4.4 Penerapan Algoritma Dijkstra

Algoritma Dijkstra diimplementasikan untuk memenuhi tujuan utama sistem, yaitu memberikan rekomendasi tujuan berdasarkan rute perjalanan terpendek, bukan sekadar jarak garis lurus. Untuk menjalankan fungsinya, algoritma ini memerlukan model jaringan jalan dalam bentuk graf yang terdiri dari titik

(node) dan sisi (edge) yang memiliki bobot (jarak). Proses kerja Algoritma Dijkstra yang diimplementasikan dalam kode adalah sebagai berikut:

1. Inisialisasi Sistem menetapkan lokasi pengguna sebagai titik awal dengan jarak 0, sementara semua titik lain diberi jarak tak terhingga.
2. Selanjutnya Algoritma secara berulang memilih titik yang belum dikunjungi dengan jarak terkecil. Dari titik tersebut, algoritma memeriksa setiap tetangganya dan memperbarui jaraknya jika ditemukan rute yang lebih pendek. Proses ini menggunakan rumus inti:

Jika  $d(u) + w(u, v) < d(v)$ , maka  $d(v) := d(u) + w(u, v)$

3. Sehingga hasil akhir proses ini berlanjut ketika jarak terpendek dari titik awal ke semua titik lain ditemukan.

Contoh perhitungan manual :

Kita akan simulasikan perhitungan Dijkstra pada sebuah graf sederhana yang mempresentasikan beberapa lokasi di Kecamatan Kemiling.

Dengan model graf :

- a) Titik (Node): A (Lokasi Anda), B (Camp 91), C (Simpang Kedaung), D (Villa LDR), E (Taman Kupu-Kupu):
- b) Sisi (Edge) dan Bobot (Jarak dalam km):
  - 1) A → B : 5 km
  - 2) A → C : 2 km
  - 3) B → D : 6 km
  - 4) C → B : 1 km
  - 5) C → D : 4 km
  - 6) C → E : 7 km
  - 7) D → E : 3 km

**Tabel 4.41 Algoritma Dijkstra**

No	Node yang Belum Dikunjungi (Jarak)	Node Terpilih	Tetangga	Perhitungan jarak baru	Jarak final Diperbaiki
1.	A(0), B( $\infty$ ), C( $\infty$ ), D( $\infty$ ), E( $\infty$ )	-	-	-	-

2.	A(0), B( $\infty$ ), C( $\infty$ ), D( $\infty$ ), E( $\infty$ )	A	B C	$d(A)+w(A,B) = 0+5=5$ $d(A)+w(A,C) = 0+2=2$	$d(B)=5$ $d(C)=2$
3.	B(5), C(2), D( $\infty$ ), E( $\infty$ )	C	B D E	$d(C)+w(C,B) = 2+1=3$ $d(C)+w(C,D) = 2+4=6$ $d(C)+w(C,E) = 2+7=9$	$d(B)=3$ (lebih kecil dari 5) $d(D)=6$ $d(E)=9$
4.	B(3), D(6), E(9)	B	D	$d(B)+w(B,D) = 3+6=9$	(Tidak ada perubahan, 9 > 6)
5.	D(6), E(9)	D	E	$d(D)+w(D,E) = 6+3=9$	(Tidak ada perubahan, 9 = 9)
6.	E(9)	E	-	-	-

Sehingga hasil akhir berdasarkan tabel diatas menemukan total jarak terpendek dari lokasi anda (A) adalah:

- a. Jarak ke Camp 91 (B): 3 km (melalui Simpang Kedaung)
- b. Jarak ke Simpang Kedaung (C): 2 km
- c. Jarak ke Villa LDR (D): 6 km (melalui Simpang Kedaung)
- d. Jarak ke Taman Kupu-Kupu (E): 9 km (melalui Simpang Kedaung)

Dengan contoh ini menunjukkan bagaimana Dijkstra secara cerdas menemukan rute ke Camp 91 lebih pendek jika melewati simpang Kedaung ( $2+1=3$  KM) daripada rute langsung (5 km). Proses manual inilah yang dieksekusi oleh kode PHP di bawah ini secara otomatis dan cepat.

Berikut ini kode Implementasi dari Algoritma Dijkstra pada API (*Application Programming Interface*) menggunakan Bahasa pemrograman PHP (*Hypertext Preprocessor*):

**Tabel 4.42 Kode Algoritma Dijkstra**

```
<?php
function getDistanceMatrix($nodes, $apiKey) {
    $locations = [];
    $nodeKeys = array_keys($nodes); // Simpan urutan key/ID
```

```

foreach ($nodes as $node) {
    $locations[] = $node['lat'] . ',' . $node['lng'];
}
$locationStr = implode('|', $locations);

$url = "https://maps.googleapis.com/maps/api/distancematrix/json?origins=" . urlencode($locationStr) . "&destinations=" . urlencode($locationStr) . "&key={$apiKey}";

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_TIMEOUT, 20);
$response = curl_exec($ch);
curl_close($ch);

$result = json_decode($response, true);
$matrix = [];

if (isset($result['rows'])) {
    foreach ($result['rows'] as $i => $row) {
        $originKey = $nodeKeys[$i];
        $matrix[$originKey] = [];
        foreach ($row['elements'] as $j => $element) {
            $destKey = $nodeKeys[$j];
            if ($element['status'] == 'OK') {
                $matrix[$originKey][$destKey] = $element['distance']['value'] / 1000; // Jarak dalam KM
            } else {

```

```

    $matrix[$originKey][$destKey] = INF; // Tandai
    tidak terjangkau
    }
}
}
return $matrix;
}
return null;
}

// Implementasi Algoritma Dijkstra
function dijkstra_all_distances($graph, $startNode) {
    $distances = [];
    $queue = new SplPriorityQueue();
    foreach ($graph as $node => $edges) {
        $distances[$node] = INF;
    }
    $distances[$startNode] = 0;
    $queue->insert($startNode, 0);
    while (!$queue->isEmpty()) {
        $currentNode = $queue->extract();
        if ($distances[$currentNode] === INF || !isset($graph[$currentNode])) {
            continue;
        }
        foreach ($graph[$currentNode] as $neighbor => $weight) {
            $salt = $distances[$currentNode] + $weight;
            if ($salt < $distances[$neighbor]) {
                $distances[$neighbor] = $salt;
                $queue->insert($neighbor, -$salt);
            }
        }
    }
}

```

```

        }
    }
}

return $distances;
}

?>
```

Pengujian dilakukan menggunakan perangkat lunak *Postman* dengan mengirimkan permintaan *POST* yang berisi data *latitude* dan *longitude*. Sebagai studi kasus, digunakan titik koordinat Kampus IIB Darmajaya ("latitude": -5.377201, "longitude": 105.249535).

```
{
  "status": "success",
  "data": [
    {
      "id_produk": "141",
      "nama_produk": "Bukit Sakura",
      "kategori": "Tempat Wisata",
      "distance_route_km": 5.58
    },
    {
      "id_produk": "146",
      "nama_produk": "KDI Handmade",
      "kategori": "UMKM",
      "distance_route_km": 7.00
    },
    {
      "id_produk": "138",
      "nama_produk": "Kopi Gunawan",
      "kategori": "UMKM",
    }
  ]
}
```

```

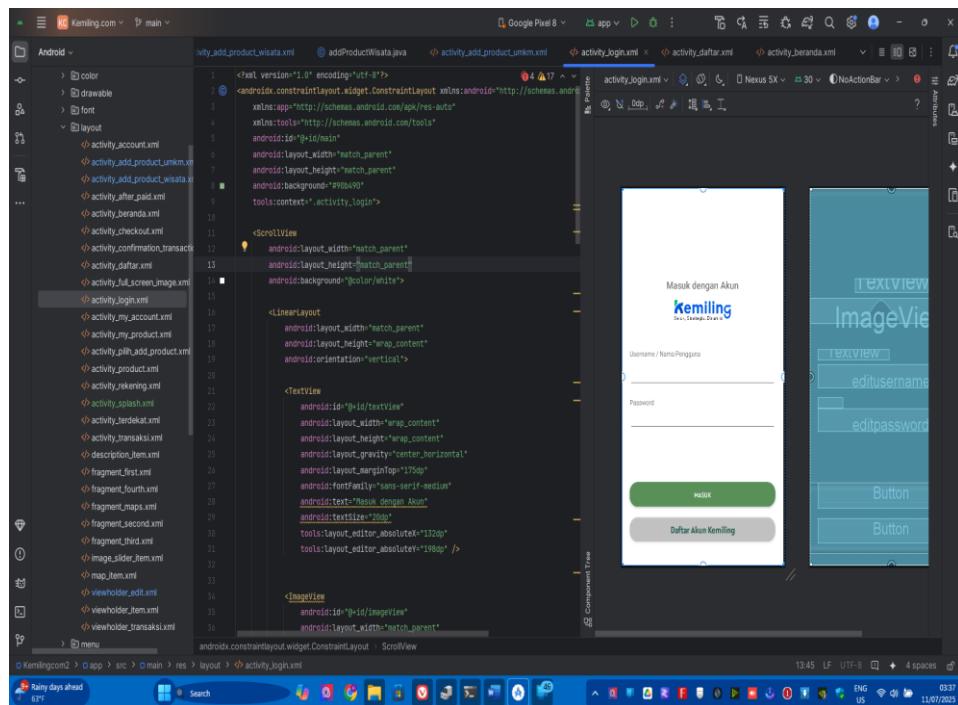
    "distance_route_km": 8.35
}
]
}

```

## 4.5 Tahapan Pengkodean dan Integrasi Sistem (Construction)

Pada tahap ini rancangan yang sudah dibuat akan diimplementasi ke dalam kode program yang sesuai dengan desain sistem. Dan akan dilakukan testing menggunakan blackbox untuk menguji fungsionalitas sistem yang sudah dibangun serta mengetahui apakah aplikasi sudah berfungsi dan sesuai dengan rancangan.

### a) Halaman Login



Gambar 4.20 Kode Desain Login

Pada gambar 4.20 diatas mendefinisikan struktur visual dan komponen antarmuka untuk halaman login aplikasi. Secara ringkas, kode ini berfungsi untuk menyusun semua komponen visual secara deklaratif guna menciptakan halaman login yang fungsional bagi pengguna.

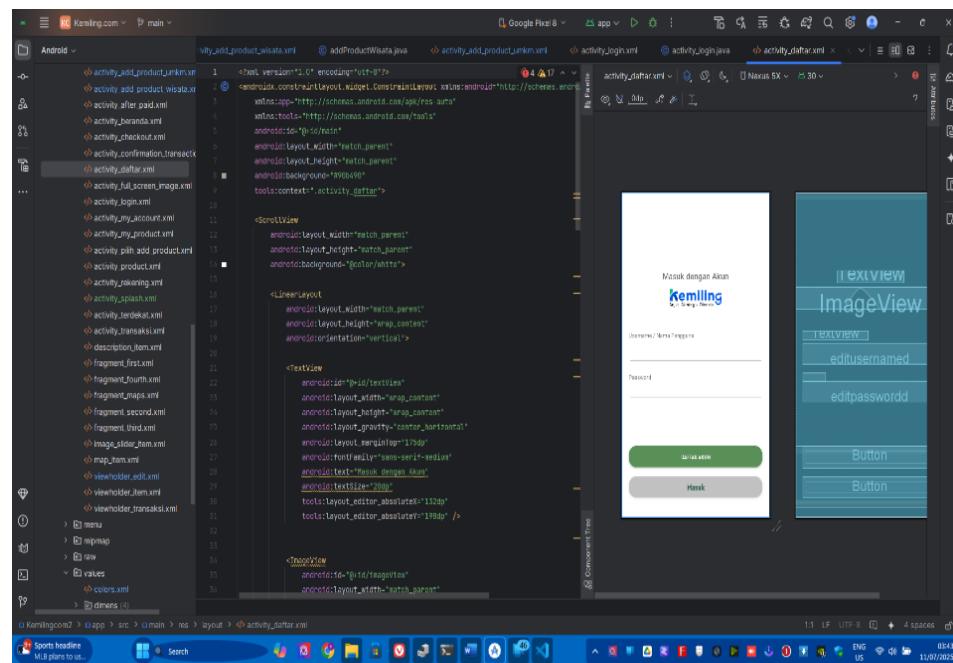
Gambar 4.21 Kode Login

Pada gambar 4.21 diatas berfungsi sebagai logika kode untuk antarmuka halaman login. Di dalamnya kode menghubungkan elemen visual dari file xml ke variabel kode java. Fungsi utamanya adalah menangani input dari interaksi pengguna lalu mengarahkannya ke halaman utama jika sesi masih aktif.

Gambar 4. 22 Api Login

Pada gambar 4.22 diatas adalah kode backend (API) yang berfungsi untuk memproses permintaan login dari aplikasi android. Setelah mendapatkan data username dan password dari pengguna yang dikirimkan, akan dilakukan kueri ke database untuk mencari pengguna berdasarkan data yang dikirim. Jika di temukan, maka akan diverifikasi dan tersimpan ke database.

### b) Halaman Daftar

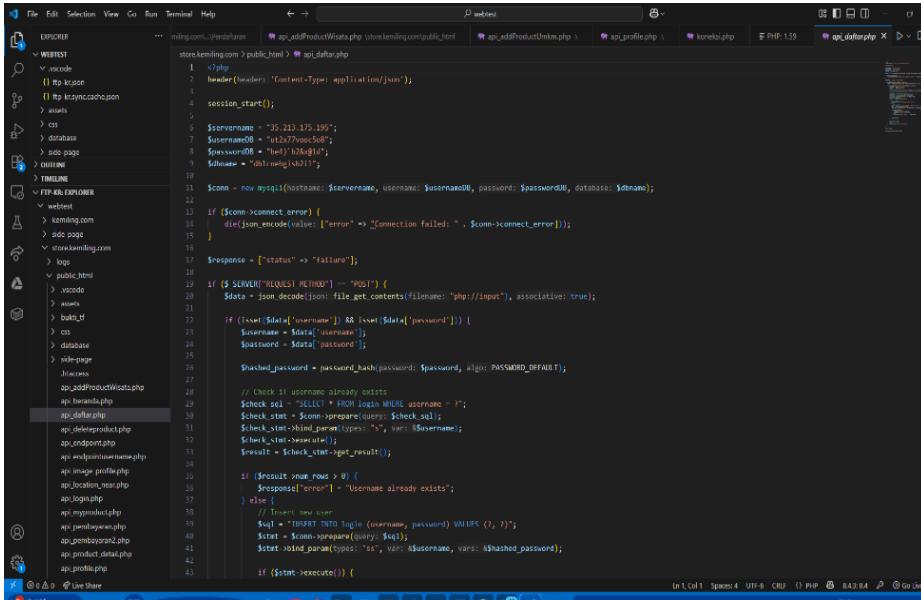


Gambar 4.23 Kode Desain Daftar

Pada gambar 4.23 diatas mendefinisikan struktur visual dan komponen antarmuka untuk halaman login aplikasi. Secara ringkas, kode ini berfungsi untuk menyusun semua komponen visual secara deklaratif guna menciptakan halaman Daftar yang fungsional bagi pengguna.

*Gambar 4.24 Kode Daftar*

Pada gambar 4.24 diatas berfungsi sebagai logika kode untuk antarmuka halaman daftar. Di dalamnya kode menghubungkan elemen visual dari file xml ke variabel kode java. Fungsi utamanya adalah menangani input dari interaksi pengguna lalu mengarahkannya ke halaman login jika akun berhasil terdaftar.



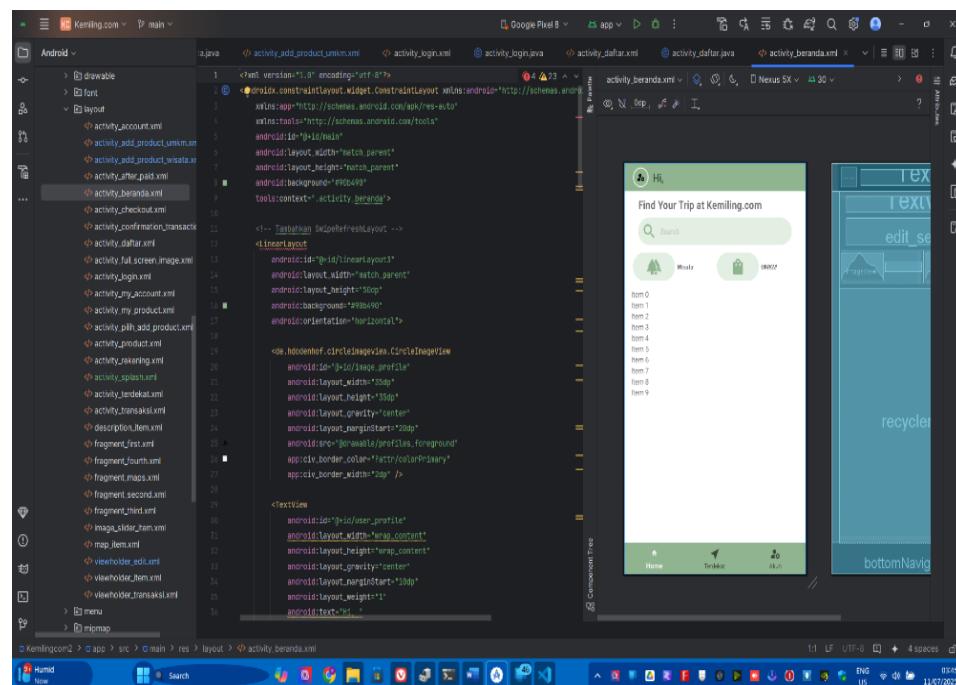
The screenshot shows a browser window with multiple tabs open, displaying a complex PHP exploit payload. The payload includes session management, database connection logic, and various SQL queries for user enumeration and password cracking. The code is heavily obfuscated with comments and whitespace.

```
file Edit Selection View Go Run Terminal Help ← → / index
EXPLORER ... maling.com.../sharehash ... api_addProductWisata.php store.lenkimg.com/public_html ... api_addProductUmkm.php ... api_changeProfile.php ... lenkimg.php F PHP 1.59 api_dishDetail.php
WEBTEST ... maling.com.../sharehash ... api_addProductWisata.php store.lenkimg.com/public_html ... api_dishDetail.php
store.lenkimg.com/public_html/api_dishDetail.php
1 <?php
2 header('Content-Type: application/json');
3
4 session_start();
5
6 $servername = "192.168.175.160";
7 $usernameDB = "root";
8 $passwordDB = "b62f3b40f0d";
9 $dbname = "dkitriang(dk1117)";
10
11 $conn = new mysqli($hostname, $username, $usernameDB, $passwordDB, $dbname);
12
13 if ($conn->connect_error) {
14 die(json_encode(['error' => "Connection failed: " . $conn->connect_error]));
15 }
16
17 $response = ["status" => "Failure"];
18
19 if ($SERVER["REQUEST_METHOD"] == "POST") {
20 $data = json_decode(file_get_contents("filename: \"php://input\"", associative: true));
21
22 if ($data[\"stat\"] == "Stat1[\"username\"]") {
23 $username = $data[\"stat\"]["username"];
24 $password = $data[\"stat\"]["password"];
25
26 $hashed_password = password_hash($password, algo: PASSWORD_DEFAULT);
27
28 // Check if username already exists
29 $check_sql = "SELECT * FROM login WHERE username = ?";
30 $check_stmt = $conn->prepare(query: $check_sql);
31 $check_stmt->bind_param(types: "s", var: $username);
32 $check_stmt->execute();
33 $result = $check_stmt->get_result();
34
35 if ($result->num_rows > 0) {
36 $response["error"] = "Username already exists";
37 } else {
38 // Insert new user
39 $sql = "INSERT INTO login (username, password) VALUES (?, ?)";
40 $stmt = $conn->prepare(query: $sql);
41 $stmt->bind_param(types: "ss", var: $username, var: $hashed_password);
42
43 if ($stmt->execute()) {
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
317
318
319
319
320
321
322
323
324
325
326
327
327
328
329
329
330
331
332
333
334
335
336
337
337
338
339
339
340
341
342
343
344
345
345
346
347
347
348
349
349
350
351
352
353
353
354
355
355
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
```

*Gambar 4.25 Kode Api Daftar*

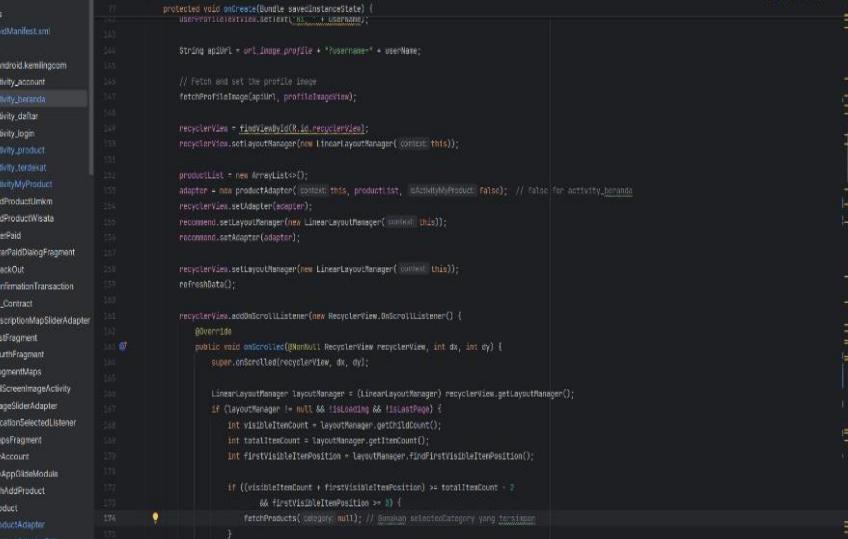
Pada gambar 4.25 diatas adalah kode backend (API) yang berfungsi untuk memproses permintaan daftar dari aplikasi android. Setelah mendapatkan data username dan password dari pengguna yang dikirimkan, akan dilakukan penyimpanan ke database berdasarkan data yang dikirim. Jika berhasil, maka akan diverifikasi dan tersimpan ke database

### c) Halaman Beranda



Gambar 4.26 Kode Desain Beranda

Pada gambar 4.26 diatas mendefinisikan struktur visual dan komponen antarmuka untuk halaman beranda aplikasi. Secara ringkas, kode ini berfungsi untuk menyusun semua komponen visual secara deklaratif guna menciptakan halaman beranda yang fungsional bagi pengguna.



```
public class activity_beranda extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_beranda);
        String imgProfile = urlImageProfile + "username=" + userName;
        fetchProfileImageCapture(imgProfile);
        recyclerView = findViewById(R.id.recycleView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setAdapter(adapter);
        recyclerView.setLinearLayoutManager(new LinearLayoutManager(this));
        recommend.setAdapter(adapter);
        recyclerView.setLinearLayoutManager(new LinearLayoutManager(this));
        recyclerView.setRecyclerListener(new RecyclerView.OnScrollListener() {
            @Override
            public void onScrolled(@NonNull RecyclerView recyclerView, int dx, int dy) {
                super.onScrolled(recyclerView, dx, dy);
                LinearLayoutManager layoutManager = (LinearLayoutManager) recyclerView.getLayoutManager();
                if (layoutManager != null & & layoutManager.isLastPage()) {
                    int visibleItemCount = layoutManager.getVisibleItemCount();
                    int totalItemCount = layoutManager.getItemCount();
                    int firstVisibleItemPosition = layoutManager.findFirstVisibleItemPosition();
                    if ((visibleItemCount + firstVisibleItemPosition) > totalItemCount - 2
                            && firstVisibleItemPosition > 0) {
                        fetchProducts(category: null); // Random select category yang tersisa
                    }
                }
            }
        });
    }
}
```

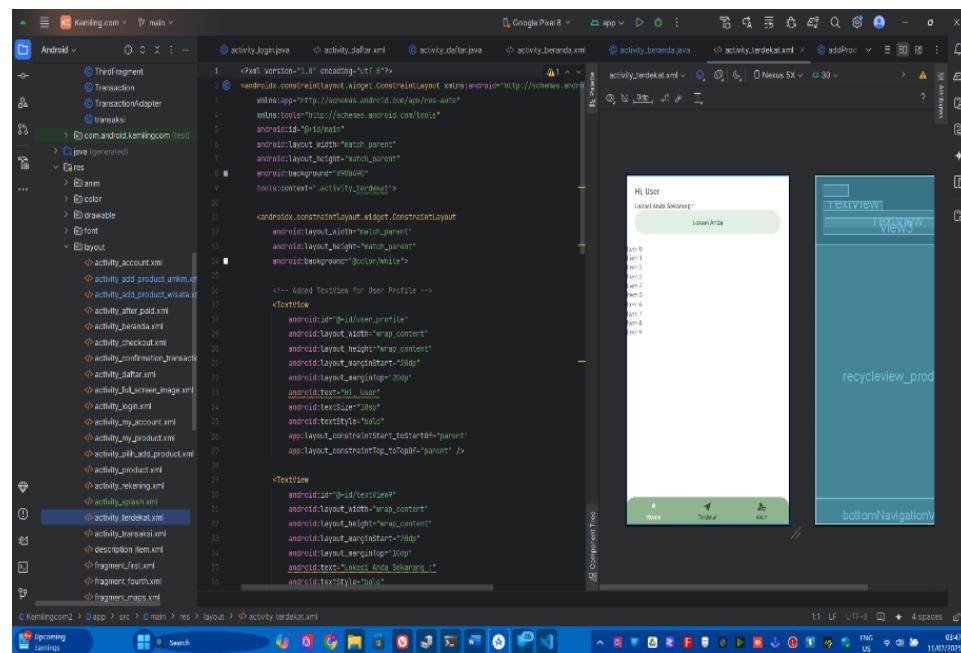
*Gambar 4.27 Kode Beranda*

Pada gambar 4.27 diatas berfungsi sebagai logika kode untuk antarmuka halaman beranda. Di dalamnya kode menghubungkan elemen visual dari file xml ke variabel kode java. Fungsi utamanya adalah menampilkan produk wisata dan umkm dari data yang di input oleh pengguna atau pelaku usaha.

*Gambar 4.28 Kode Api Beranda*

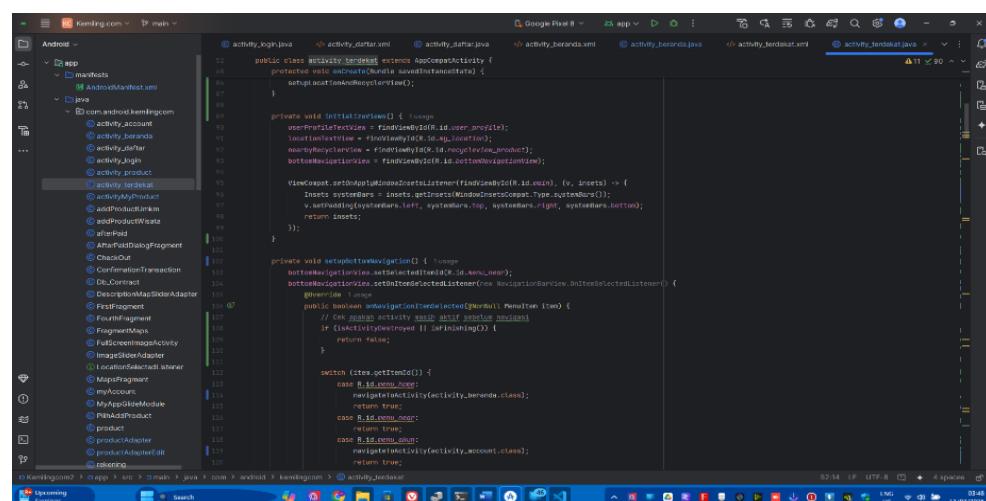
Pada gambar 4.28 diatas adalah kode backend (API) yang berfungsi untuk memproses permintaan get data dari database dan di tampilkan ke halaman beranda aplikasi.

#### d) Halaman Terdekat



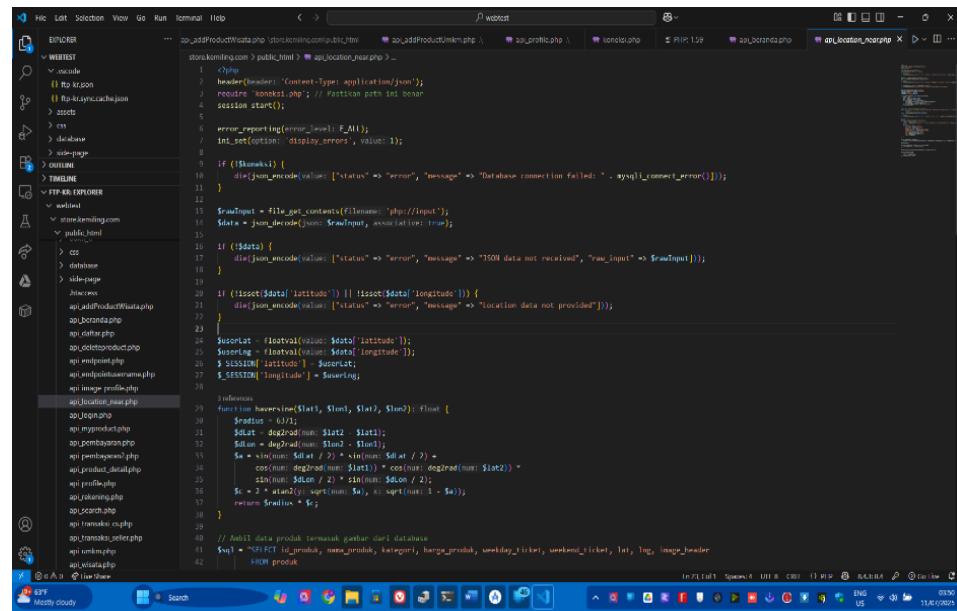
Gambar 4.29 Kode Desain Terdekat

Pada gambar 4.29 diatas mendefinisikan struktur visual dan komponen antarmuka untuk halaman terdekat aplikasi. Secara ringkas, kode ini berfungsi untuk menyusun semua komponen visual secara deklaratif guna menciptakan halaman terdekat yang fungsional bagi pengguna.



Gambar 4.30 Kode Terdekat

Pada gambar 4.30 diatas berfungsi sebagai logika kode untuk antarmuka halaman terdekat. Di dalamnya kode menghubungkan elemen visual dari file xml ke variabel kode java. Fungsi utamanya adalah menampilkan produk wisata dan umkm dari data yang di input oleh pengguna atau pelaku usaha. Sebelum di tampilkan, data di proses oleh algoritma sehingga data yang tertampil adalah data produk terdekat dari pengguna.



```

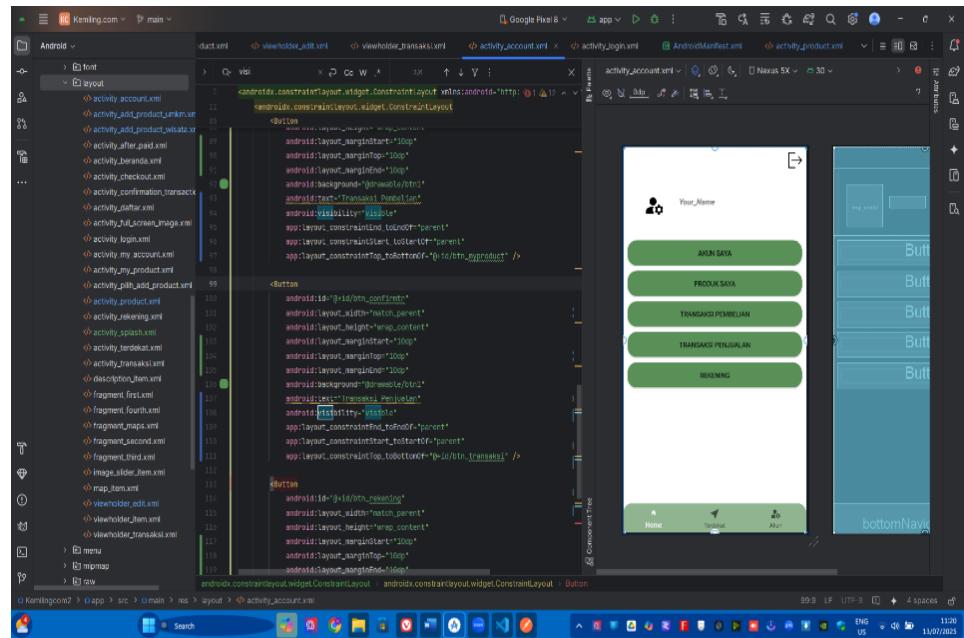
<?php
header('Content-Type: application/json');
require 'koneksi.php'; // Pangkan path ini buat session start();
session_start();
error_reporting(error_level: E_ALL);
ini_set(option: display_errors, value: 1);
if ($koneksi) {
    die(json_encode(value: ["status" => "error", "message" => "Database connection failed: " . mysqli_connect_error()]));
}
$rawInput = file_get_contents(filename: 'php://input');
$data = json_decode(json: $rawInput, association: true);
if ($data) {
    die(json_encode(value: ["status" => "error", "message" => "JSON data not received", "raw_input" => $rawInput]));
}
if (!isset($data['latitude']) || !isset($data['longitude'])) {
    die(json_encode(value: ["status" => "error", "message" => "location data not provided"]));
}
$UserLat = floatval(value: $data['latitude']);
$UserLong = floatval(value: $data['longitude']);
$_SESSION['latitude'] = $userLat;
$_SESSION['longitude'] = $userLong;
$_SESSION['longitude'] = $userLong;
function haversine($lat1, $lon1, $lat2, $lon2): float {
    $radius = 6371;
    $lat1 = deg2rad(deg: $lat1);
    $lon1 = deg2rad(deg: $lon1);
    $lat2 = deg2rad(deg: $lat2);
    $lon2 = deg2rad(deg: $lon2);
    $a = sin(radians(deg1 / 2) * sin.radians(deg2)) + cos(radians(deg1 / 2) * cos(radians(deg2 / 2)) *
        cos(radians(deg1 / 2) * sin.radians(deg2 / 2));
    $c = 2 * atan2(sqrt(num: $a), sqrt(num: 1 - $a));
    $k = 2 * $radius * $c;
    return $k;
}
// ambil data produk termasuk ganteng dari database
$sql = "SELECT id_produk, nama_produk, kategori, harga_produk, weekend_ticket, weekday_ticket, tgl, img_header
        FROM produk";

```

Gambar 4.31 Kode API Terdekat

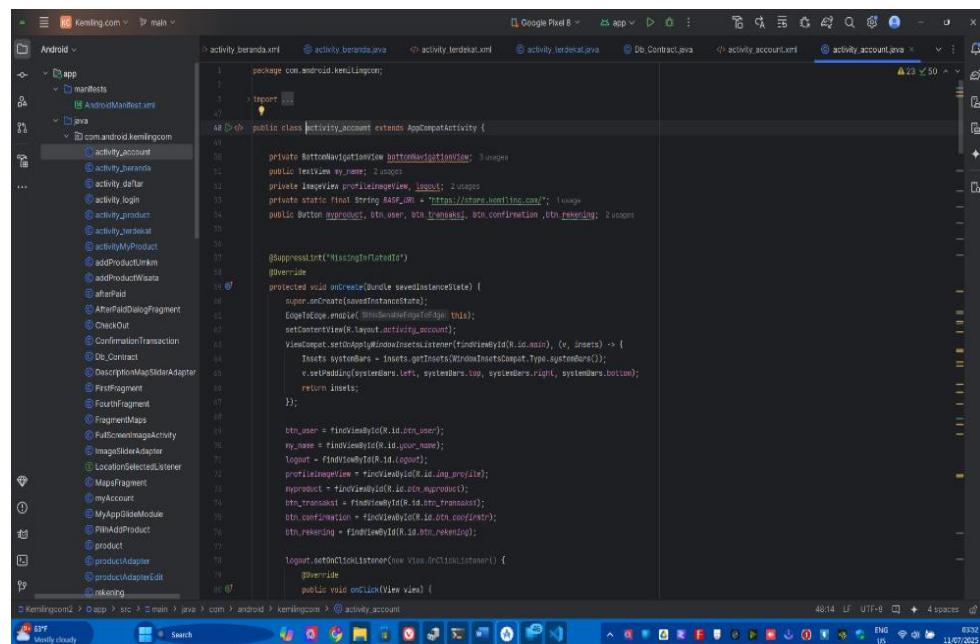
Pada gambar 4.31 diatas adalah kode backend (API) yang berfungsi untuk memproses permintaan get data dari database dan di tampilkan ke halaman terdekat aplikasi. Kode ini terimplementasi algoritma dijkstra, sehingga data yang di tampilkan adalah hasil dari penerapan algoritma dijktra, dan merupakan data terdekat dari pengguna secara *real-time*.

- Halaman Akun



*Gambar 4.32 Kode Desain Akun*

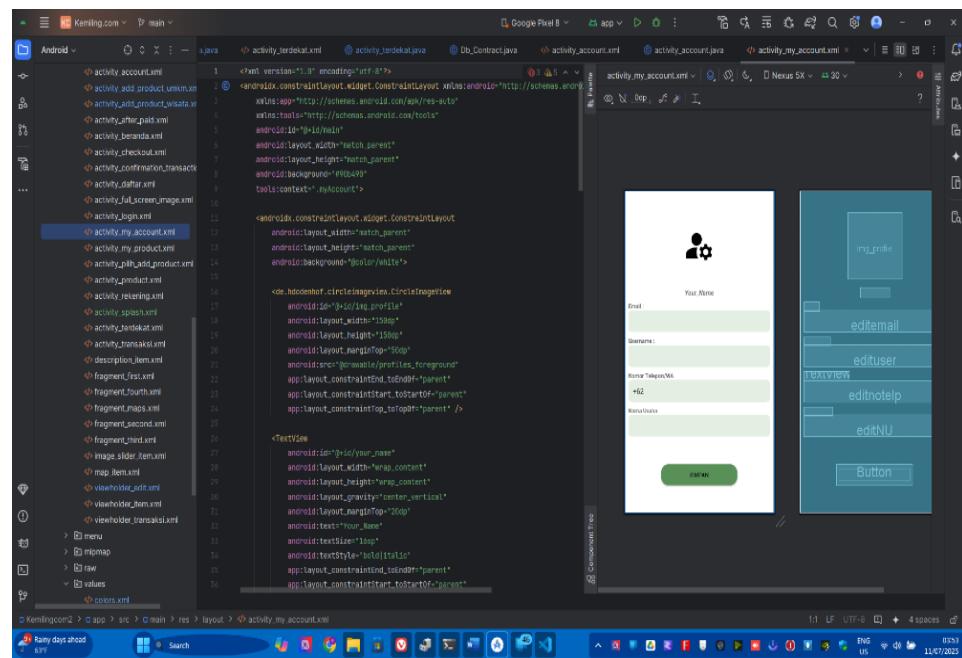
Pada gambar 4.32 diatas mendefinisikan struktur visual dan komponen antarmuka untuk halaman akun aplikasi. Secara ringkas, kode ini berfungsi untuk menyusun semua komponen visual secara deklaratif guna menciptakan halaman akun yang fungsional bagi pengguna.



*Gambar 4.33 Kode akun*

Pada gambar 4.33 diatas berfungsi sebagai logika kode untuk antarmuka halaman akun. Di dalamnya kode menghubungkan elemen visual dari file xml ke variabel kode java. Fungsi utamanya adalah menampilkan beberapa tombol pilihan untuk melakukan perubahan data, baik data akun ataupun data transaksi hingga produk dan akan tersimpan ke dalam database.

#### f) Halaman Akun Saya



Gambar 4.34 Kode Desain Akun Saya

Pada gambar 4.34 diatas mendefinisikan struktur visual dan komponen antarmuka untuk halaman akun saya aplikasi. Secara ringkas, kode ini berfungsi untuk menyusun semua komponen visual secara deklaratif guna menciptakan halaman akun saya yang fungsional bagi pengguna.

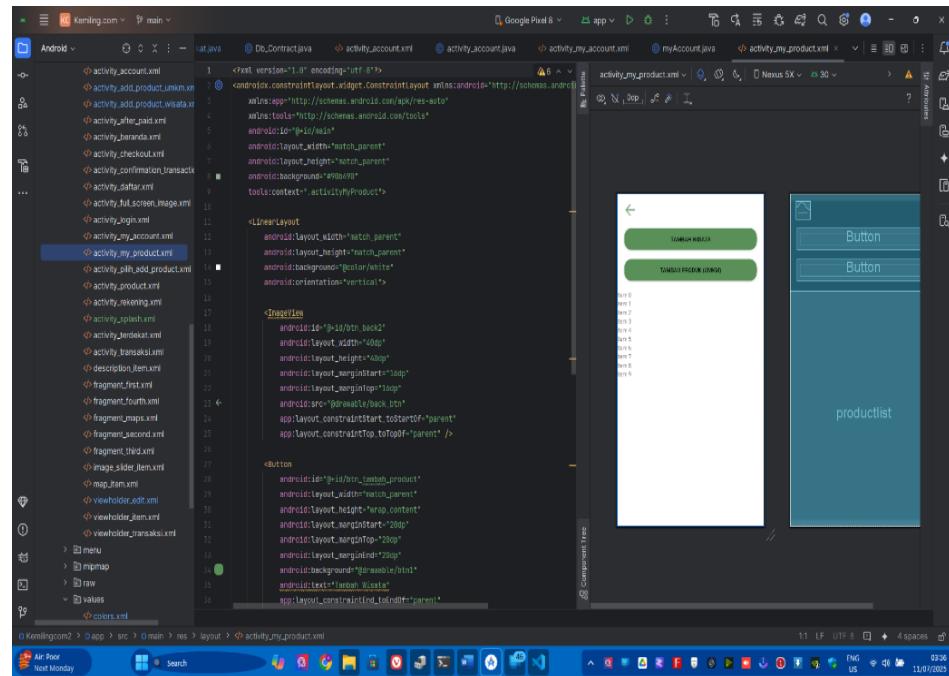
*Gambar 4.35 Kode Akun Saya*

Pada gambar 4.35 diatas berfungsi sebagai logika kode untuk antarmuka halaman akun saya. Di dalamnya kode menghubungkan elemen visual dari file xml ke variabel kode java. Fungsi utamanya adalah menampilkan beberapa field untuk melakukan perubahan data akun dan akan tersimpan ke dalam database.

*Gambar 4.36 Kode API Akun Saya*

Pada gambar 4.36 diatas adalah kode backend (API) yang berfungsi untuk memproses permintaan data dari database dan di tampilkan ke halaman akun saya di aplikasi. Kode ini dapat menerima perubahan data dan menyimpan ke database.

### g) Halaman Produk Saya



Gambar 4.37 Kode Desain Produk Saya

Pada gambar 4.37 diatas mendefinisikan struktur visual dan komponen antarmuka untuk halaman produk saya aplikasi. Secara ringkas, kode ini berfungsi untuk menyusun semua komponen visual secara deklaratif guna menciptakan halaman produk saya yang fungsional bagi pengguna.

```

public class ActivityMyProduct extends AppCompatActivity {
    private void fetchProducts(int userId) {
        String url = "https://store.kemiling.com/api_myproduct.php?user_id=" + userId;
        Log.d("tag", "fetchProducts", "msg: " + "fetching from URL " + url);
        JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(
            Request.Method.GET,
            url,
            null,
            new Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {
                    try {
                        JSONArray dataArray = response.getJSONArray("data");
                        productList.clear(); // Clear previous data
                        for (int i = 0; i < dataArray.length(); i++) {
                            JSONObject productObject = dataArray.getJSONObject(i);

                            int id = productObject.getInt("id");
                            String title = productObject.getString("name");
                            int price = productObject.getInt("price");

                            // Safe parsing category & ticket
                            String category = productObject.has("category") && !productObject.isNull("category")
                                ? productObject.getString("name_category").trim()
                                : "";

                            int weekday_ticket = productObject.has("name_weekday_ticket") && !productObject.isNull("name_weekday_ticket")
                                ? productObject.getInt("name_weekday_ticket")
                                : 0;

                            int weekend_ticket = productObject.has("name_weekend_ticket") && !productObject.isNull("name_weekend_ticket")
                                ? productObject.getInt("name_weekend_ticket")
                                : 0;
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            }
        );
    }
}

```

Gambar 4.38 Kode Produk Saya

Pada gambar 4.38 diatas berfungsi sebagai logika kode untuk antarmuka halaman produk saya. Di dalamnya kode menghubungkan elemen visual dari file xml ke variabel kode java. Fungsi utamanya adalah menampilkan data produk yang sudah di upload oleh pengguna (pelaku usaha) baik dari wisata maupun umkm.

```

$ curl -X POST https://store.kemiling.com/api_myproduct.php
{
    "data": [
        {
            "id": 1,
            "name": "Kaos Distro",
            "price": 15000,
            "category": "Clothing"
        },
        {
            "id": 2,
            "name": "Topi Distro",
            "price": 20000,
            "category": "Accessories"
        }
    ]
}

```

```

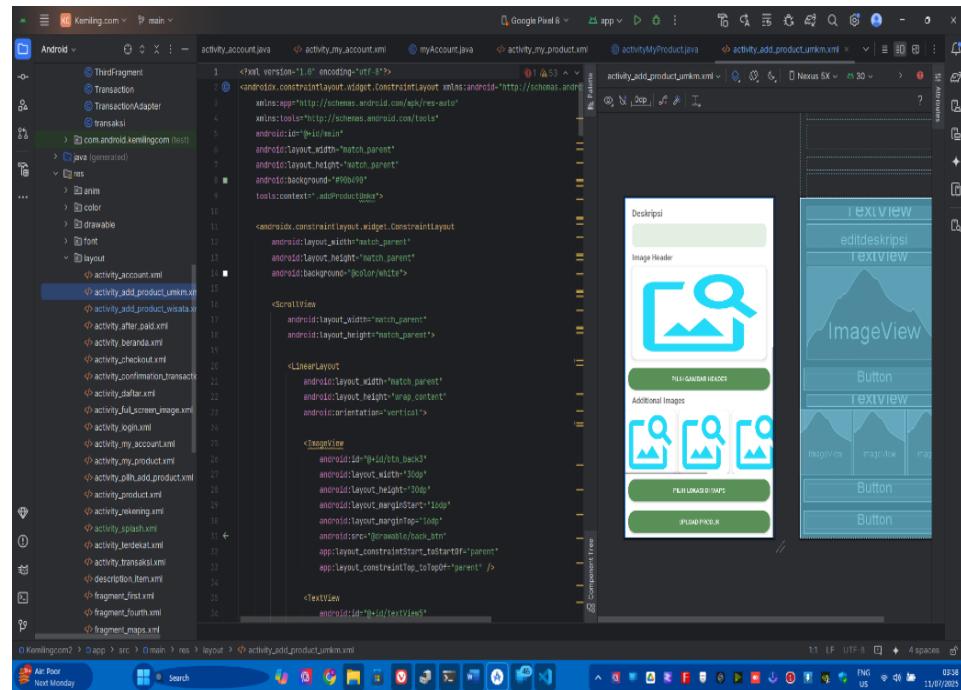
$ curl -X POST https://store.kemiling.com/api_myproduct.php
{
    "data": [
        {
            "id": 1,
            "name": "Kaos Distro",
            "price": 15000,
            "category": "Clothing"
        },
        {
            "id": 2,
            "name": "Topi Distro",
            "price": 20000,
            "category": "Accessories"
        }
    ]
}

```

Gambar 4.39 Kode Api Produk Saya

Pada gambar 4.39 diatas adalah kode backend (API) yang berfungsi untuk memproses permintaan data dari database dan di tampilkan ke halaman produk saya di aplikasi. Kode ini dapat mengambil data dari database dan akan di tampilkan ke aplikasi android.

#### h) Halaman Tambah Produk



Gambar 4.40 Kode Tambah Produk

Pada gambar 4.40 diatas mendefinisikan struktur visual dan komponen antarmuka untuk halaman tambah produk di aplikasi. Secara ringkas, kode ini berfungsi untuk menyusun semua komponen visual secara deklaratif guna menciptakan halaman tambah produk yang fungsional bagi pengguna.

The screenshot shows an Android Studio interface with the following details:

- Project Structure:** The project is named "Kemling.com" and contains an "app" module.
- Java Files:** Several Java files are visible in the "src/main/java/com/kemling/app" directory:
  - activity\_my\_account.xml
  - MyAccount.java
  - activity\_my\_product.xml
  - activityMyProduct.java
  - activity\_add\_product\_umkm.xml
  - addProductUmkm.java
- Manifest Files:** The "AndroidManifest.xml" file is located in the "src/main/AndroidManifest.xml" directory.
- Code Preview:** The main code block displays Java code for handling user authentication and product management. It includes methods for fetching user ID from SharedPreferences, making network requests using Volley, and updating SharedPreferences with user ID. It also handles JSON response parsing and error catching.

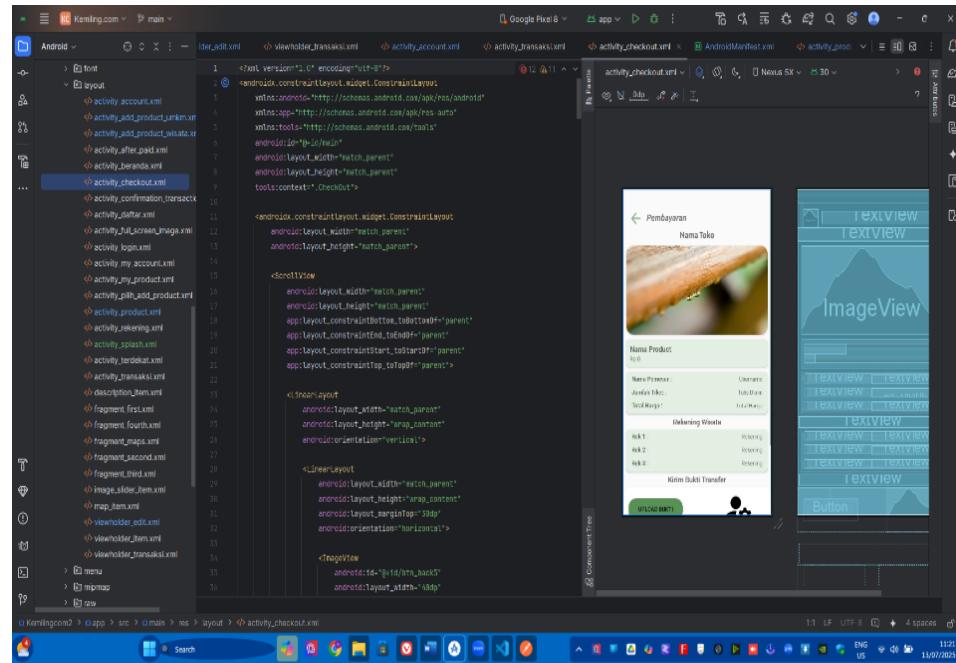
*Gambar 4.41 Kode Tambah Produk*

Pada gambar 4.41 diatas berfungsi sebagai logika kode untuk antarmuka halaman tambah produk. Di dalamnya kode menghubungkan elemen visual dari file xml ke variabel kode java. Fungsi utamanya adalah mengupload data produk ke database oleh pengguna (pelaku usaha) baik dari wisata maupun umkm.

Gambar 4.42 Kode Api Tambah Produk

Pada gambar 4.42 diatas adalah kode backend (API) yang berfungsi untuk memproses pengiriman data dari aplikasi ke database. Kode ini dapat mengirim data ke database dan akan ditampilkan ke aplikasi android.

#### i) Halaman Pembayaran



Gambar 4.43 Desain Halaman Pembayaran

Pada gambar 4.43 diatas mendefinisikan struktur visual dan komponen antarmuka untuk halaman pembayaran di aplikasi. Secara ringkas, kode ini berfungsi untuk menyusun semua komponen visual secara deklaratif guna menciptakan halaman pembayaran yang fungsional bagi pengguna.

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** KelingCom v. man
- Toolbar:** Google Pixel 8, app, etc.
- Side Navigation:** Android, showing a tree view of activity files like activity\_main.xml, activity\_checkout.xml, etc.
- Main Editor:** Checkout.java (selected tab)
- Code Content:** The code for the Checkout class, which extends AppCompatActivity. It includes methods for onActivityResult and a private helper method encodeImageToBase64.

```
public class Checkout extends AppCompatActivity {

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == PRICK_IMAGE_REQUEST && resultCode == RESULT_OK && data != null && data.getData() != null) {
            Uri imageUri = data.getData();
            Log.d(TAG, "imageUri: " + imageUri.toString());
            Uri imageUriEncoded = encodeImageToBase64(imageUri);
            buktiTF = encodeImageToBase64(imageUri);
            Log.d(TAG, "base64 String: " + buktiTF);
        }
    }

    private String encodeImageToBase64(Uri imageUri) {
        try {
            Bitmap bitmap = MediaStore.Images.Media.getBitmap(this.getContentResolver(), imageUri);
            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            bitmap.compress(Bitmap.CompressFormat.JPEG, quality, outputStream);
            byte[] byteArray = outputStream.toByteArray();
            // Log panjang byte array untuk debug
            Log.d(TAG, "Byte array length: " + byteArray.length);

            // Encode ke Base64 menggunakan NO_WRAP
            String base64String = Base64.encodeToString(byteArray, Base64.NO_WRAP);
            Log.d(TAG, "Base64 String length: " + base64String.length());
            return base64String;
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

- Bottom Navigation:** A navigation bar with icons for Home, Search, etc.
- Taskbar:** Shows the current project structure: KelingCom > App > src > main > Java > com > android > kelingcom > Checkout
- System Tray:** Shows battery level (80%), signal strength, and system status.

*Gambar 4.44 Kode Pembayaran*

Pada gambar 4.44 diatas berfungsi sebagai logika kode untuk antarmuka halaman Pembayaran. Di dalamnya kode menghubungkan elemen visual dari file xml ke variabel kode java. Fungsi utamanya adalah mengupload data transaksi ke database oleh pengguna dan dapat di verifikasi oleh pelaku usaha.

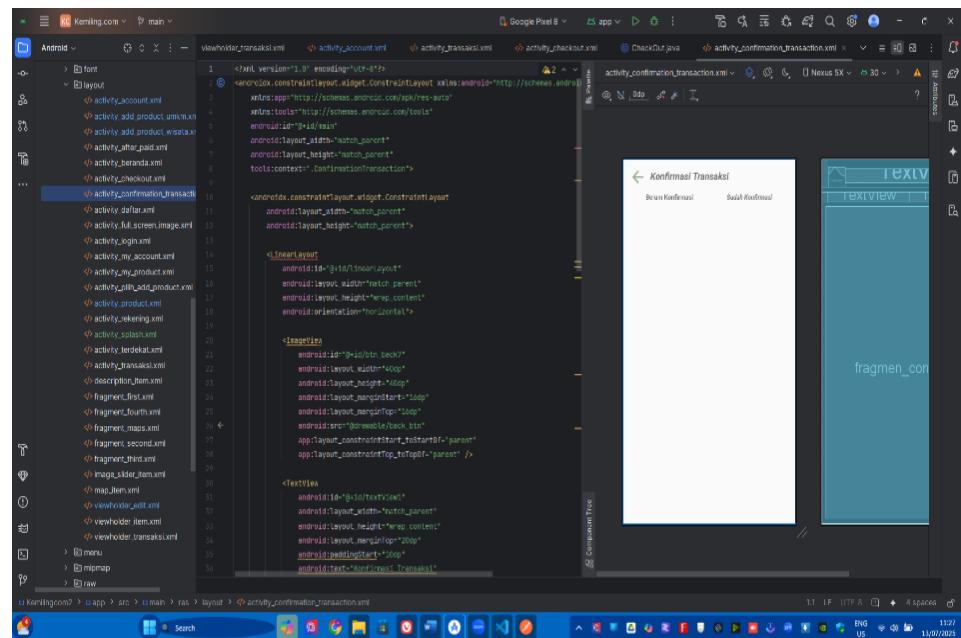
```
File Edit Selection View Go Run Terminal Help ⌘-P webkit ⌘-F Live Share

EXPLORER
  WEBIST
    - ap_index.php
    - ap_login.php
    - ap_logout.php
    - ap_myprofile.php
    - ap_pantaiyaran.php
    - ap_pantaiyaran.php
    - ap_product_detail.php
    - ap_profile.php
    - ap_rekening.php
    - ap_sewa.php
    - ap_transaksi_cus.php
    - ap_transaksi_seller.php
    - ap_unregister.php
    - ap_wisata.php
    - gelar_kemiling.php
    - index.php
    - knowns.php
    - php_wisata.php
    - privacy_policy.php
    - savelocation.php
    - test11.php
    - .htaccess
    - api
  WEBSITE
    - store_kemiling_com/public_html
      store_kemiling_com > public_html > ap_pantaiyaran.php
      19 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
      20   }
      21     if ($request->getUploadCount() > 8) {
      22       $pesan_error = "Maaf, maksimal upload file adalah 8 file";
      23       $response['status'] = 'error';
      24       $response['message'] = $pesan_error;
      25       error_log("Incomplete data: " . json_encode($data));
      26     } else {
      27       $stmt_insert = $db->prepare("INSERT INTO `tbl_pantaiyaran`(`username`, `password`, `id_produk`, `username_seller`, `nama_produk`, `weekday_ticket`, `weekend_ticket`, `harga_produk`, `jumlah`, `total_harga`, `alamat`, `bukti_tf`) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
      28       $stmt_insert->execute([
      29         $username,
      30         $password,
      31         $id_produk,
      32         $username_seller,
      33         $nama_produk,
      34         $weekday_ticket,
      35         $weekend_ticket,
      36         $harga_produk,
      37         $jumlah,
      38         $total_harga,
      39         $alamat,
      40         $bukti_tf
      41       ]);
      42       $response['status'] = 'success';
      43       $response['message'] = "Berhasil menyimpan pesanan: " . $stmt_insert->error;
      44     }
      45     $stmt_insert->close();
      46   } else {
      47     $response['status'] = 'error';
      48     $response['message'] = "Produk tidak ditemukan!";
      49   }
      50 }
      51 $stmt_produkt->close();
      52 } else {
      53   $response['status'] = 'error';
      54   $response['message'] = "Data tidak lengkap!";
      55   error_log("Incomplete data: " . json_encode($data));
      56 }
      57 }
```

*Gambar 4.45 Kode API Pembayaran*

Pada gambar 4.45 diatas adalah kode backend (API) yang berfungsi untuk memproses pengiriman data dari aplikasi ke database. Kode ini dapat mengirim data ke database dan akan di tampilkan ke aplikasi android pada halaman transaksi.

j) Halaman Konfirmasi Transaksi



Gambar 4.46 Desain Halaman Konfirmasi

Pada gambar 4.46 diatas mendefinisikan struktur visual dan komponen antarmuka untuk halaman konfirmasi transaksi di aplikasi. Secara ringkas, kode ini berfungsi untuk menyusun semua komponen visual secara deklaratif guna menciptakan halaman konfirmasi transaksi yang fungsional bagi pengguna.

```

public class ConfirmationTransaction extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Default fragment
        if (savedInstanceState == null) {
            switchFragment(new ThirdFragment());
            highlightButton(btNextConfirm);
        }
    }

    btConfirm.setOnClickListener(v -> {
        switchFragment(new FourthFragment());
        highlightButton(btNextConfirm);
    });

    btBack.setOnClickListener(v -> {
        switchFragment(new ThirdFragment());
        highlightButton(btNextConfirm);
    });

    // Back button
    btBack.setOnClickListener(v -> {
        onBackPressed();
    });
}

private void saveInstanceState(Fragment fragment) {
    FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
    transaction.replace(R.id.fragment_container, fragment);
    transaction.addToBackStack(null);
    transaction.commit();
}

private void switchFragment(Fragment fragment) {
    FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
    transaction.replace(R.id.fragment_container, fragment);
    transaction.addToBackStack(null);
    transaction.commit();
}

private void highlightButton(Button button) {
    btNextConfirm.setTextColor(Color.BLACK);
    btNextConfirm.setBackgroundColor(Color.BLUE);
    button.setTextColor(Color.parseColor("#00FFFF"));
    button.setText("Cancel");
}
}

```

Gambar 4.47 Kode Konfirmasi

Pada gambar 4.47 diatas berfungsi sebagai logika kode untuk antarmuka halaman konfirmasi transaksi. Di dalamnya kode menghubungkan elemen visual dari file xml ke variabel kode java. Fungsi utamanya adalah mengupload data transaksi ke database oleh pengguna dan dapat di verifikasi oleh pelaku usaha.

```

if (Request::method === 'GET') {
    echo json_encode(['status' => 'success']);
} else {
    $transaksi = json_decode(file_get_contents('php://input'));
    $rawPostData = file_get_contents('php://input');
    error_log(message: "Raw POST data: " . $rawPostData);
    parse_str($rawPostData, $postVars);
    $id_transaksi = intval($postVars['id_transaksi']) ?: null;
    $validasi_pembayaran = intval($postVars['validasi_pembayaran']) ?: null;
    $validasi_pembayaran = intval($postVars['validasi_pembayaran']) ?: null;

    error_log(message: "id_transaksi: " . $id_transaksi);
    error_log(message: "validasi_pembayaran": $validasi_pembayaran);

    if ($id_transaksi === null || $validasi_pembayaran === null) {
        echo json_encode(['status' => 'error', 'message' => 'id_transaksi and validasi_pembayaran are required.']);
        exit();
    }

    $sql = "UPDATE transaksi SET validasi_pembayaran = ? WHERE id_transaksi = ?";
    $stmt = $conn->prepare($query, $sql);
    $stmt->bind_param('ii', $validasi_pembayaran, $id_transaksi);

    if ($stmt->execute()) {
        echo json_encode(['status' => 'success', 'message' => 'Transaction updated successfully.']);
    } else {
        echo json_encode(['status' => 'error', 'message' => 'Failed to update transaction.']);
    }
}

$stmt->close();
$conn->close();
}

```

Gambar 4.48 Kode API Konfirmasi

Pada gambar 4.48 diatas adalah kode backend (API) yang berfungsi untuk memproses pengiriman data dari aplikasi ke database. Kode ini dapat mengirim data ke database tentang hasil konfirmasi transaksi yang dilakukan pelaku usaha.

#### **4.6 Tahapan Implementasi (Cutover)**

Hasil dari seluruh proses perancangan dan pengembangan adalah sebuah purwarupa aplikasi Android fungsional yang siap untuk digunakan, menyediakan platform terintegrasi bagi wisatawan untuk menemukan destinasi wisata dan produk UMKM, serta bagi pelaku usaha untuk mengelola konten mereka. Fitur-fitur utama seperti pendaftaran pengguna, penjelajahan konten, dan penambahan produk telah diuji dan berjalan sesuai dengan hasil yang diharapkan. Secara khusus, implementasi Algoritma Dijkstra berhasil menyediakan fungsionalitas rekomendasi tujuan terdekat berdasarkan jarak rute perjalanan yang efisien.

##### **4.6.1 Halaman Login**

Implementasi halaman Login sebagai halaman awal pada saat aplikasi dibuka yang mengharuskan pengguna memasukkan username dan password sebelum membuka aplikasi.

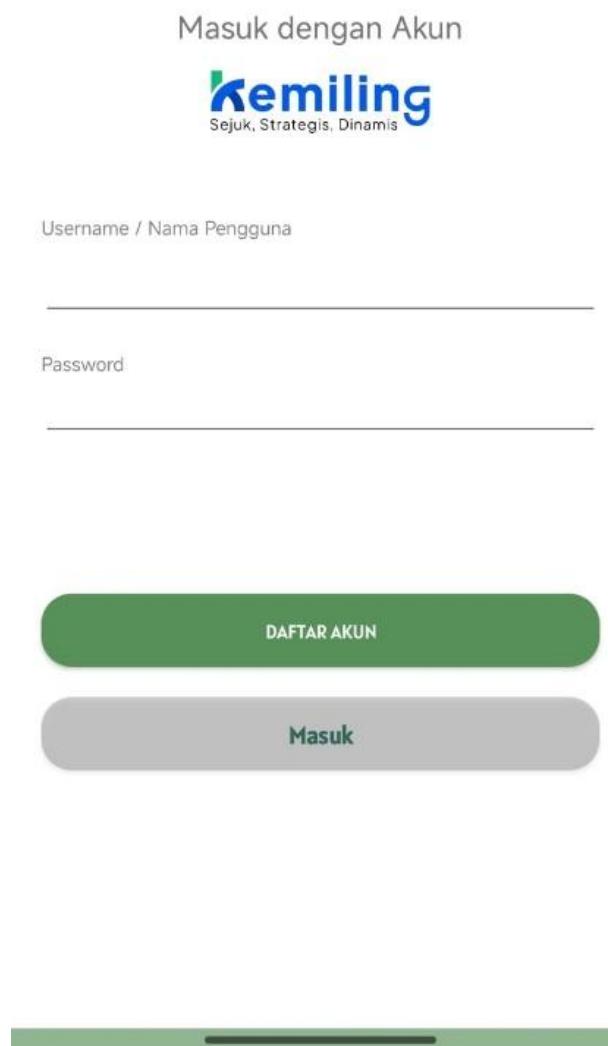


Gambar 4.49 Halaman Login

#### 4.6.2 Halaman Daftar

Implementasi halaman Daftar yang dirancang sebagai alur pendaftaran bagi pengguna baru. Dengan hanya menampilkan kolom input

username dan password serta dua tombol aksi seperti “Daftar Akun” yang berfungsi sebagai tombol yang memproses data pendaftaran.

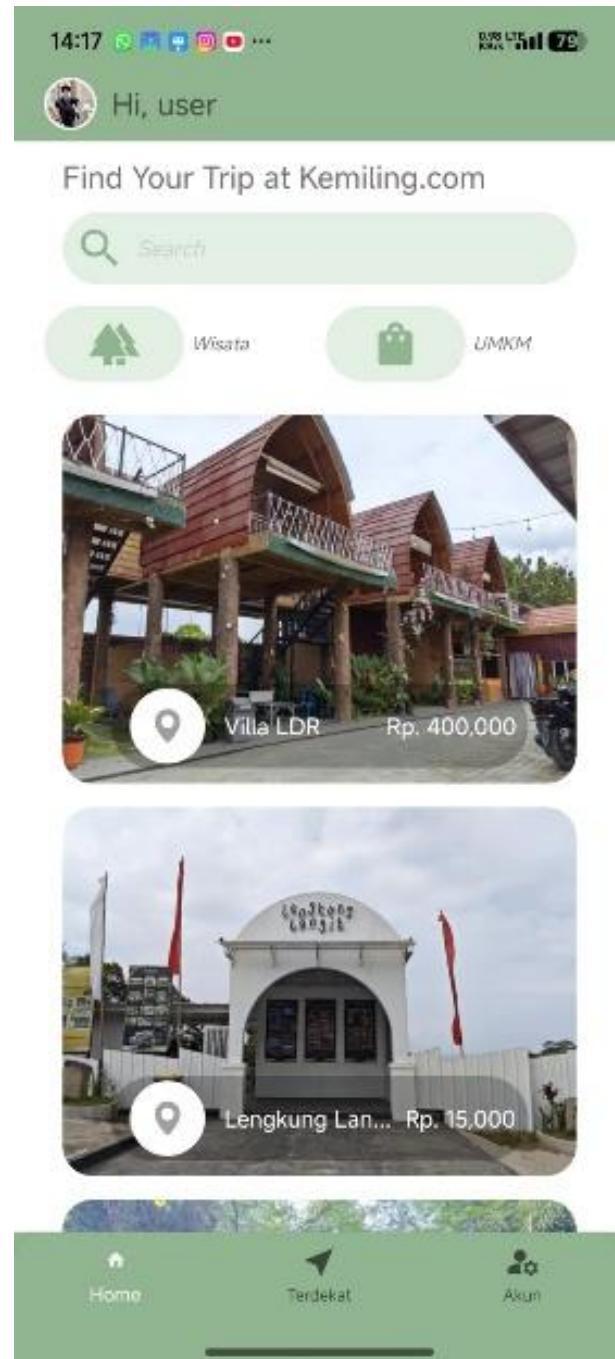


Gambar 4.50 Halaman Daftar Akun

#### 4.6.3 Halaman Beranda

Implementasi halaman Beranda yang dirancang sebagai halaman utama yang menampilkan produk destinasi wisata dan UMKM dari pelaku

usaha kecamatan Kemiling. Dan pengguna juga bisa memilih produk atau destinasi wisata yang ingin mereka tuju.

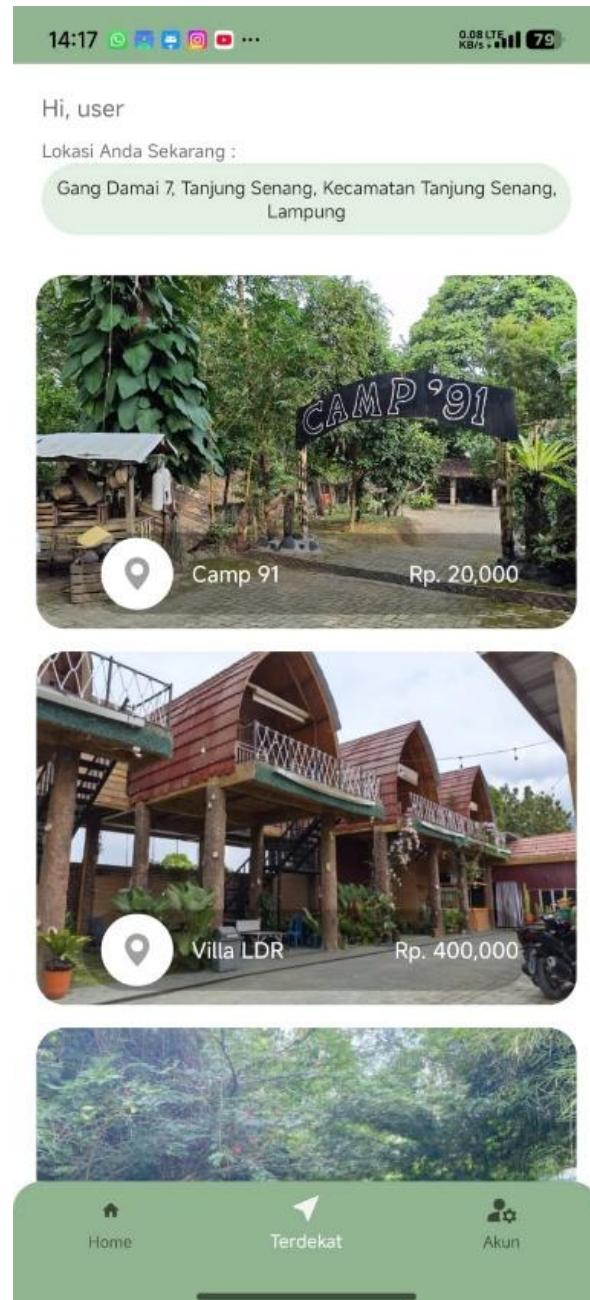


Gambar 4.51 Halaman Beranda

#### 4.6.4 Halaman Terdekat

Implementasi halaman Terdekat sebagai halaman yang akan menampilkan produk atau destinasi wisata terdekat dari tempat

pengguna secara real-time. Dengan menggunakan kalkulasi algoritma Dijkstra sebagai dasar jarak rute terpendek.



Gambar 4.52 Halaman Terdekat

#### 4.6.5 Halaman Akun

Implementasi halaman Akun yang dirancang untuk pengguna mengubah data profile mereka, menambahkan produk, atau mengecek

transaksi yang sudah di buat. Serta dapat mengkonfirmasi transaksi sebagai pelaku usaha.



Gambar 4.53 Halaman Akun

#### 4.6.6 Halaman Produk Detail

Implementasi halaman Produk detail yang dirancang untuk memberikan informasi kepada pengguna setelah mereka

memilih produk. Serta memiliki dua tombol untuk melakukan pemesanan dan melihat lokasi produk.



Gambar 4.54 Halaman Produk Detail

#### 4.6.7 Halaman Pemesanan/Pembayaran

Implementasi halaman Pembayaran yang dirancang untuk memfasilitasi proses transaksi setelah pengguna memilih atau memesan tiket. Dengan

menyediakan pengunggah bukti pembayaran melalui tombol “Upload Bukti”. Setelah itu pengguna dapat melakukan konfirmasi pembayaran mereka dengan dua tombol terakhir dibawah.



Gambar 4.55 Halaman Pemesanan

#### 4.6.8 Halaman Lokasi

Implementasi halaman Lokasi sebagai hasil integrasi sistem untuk menampilkan peta interaktif yang berpusat pada koordinat destinasi

tujuan, yang ditandai dengan pin berwarna merah. Serta ada tombol untuk melihat menggunakan Aplikasi pihak ketiga dan tombol selesai untuk kembali ke halaman sebelumnya.

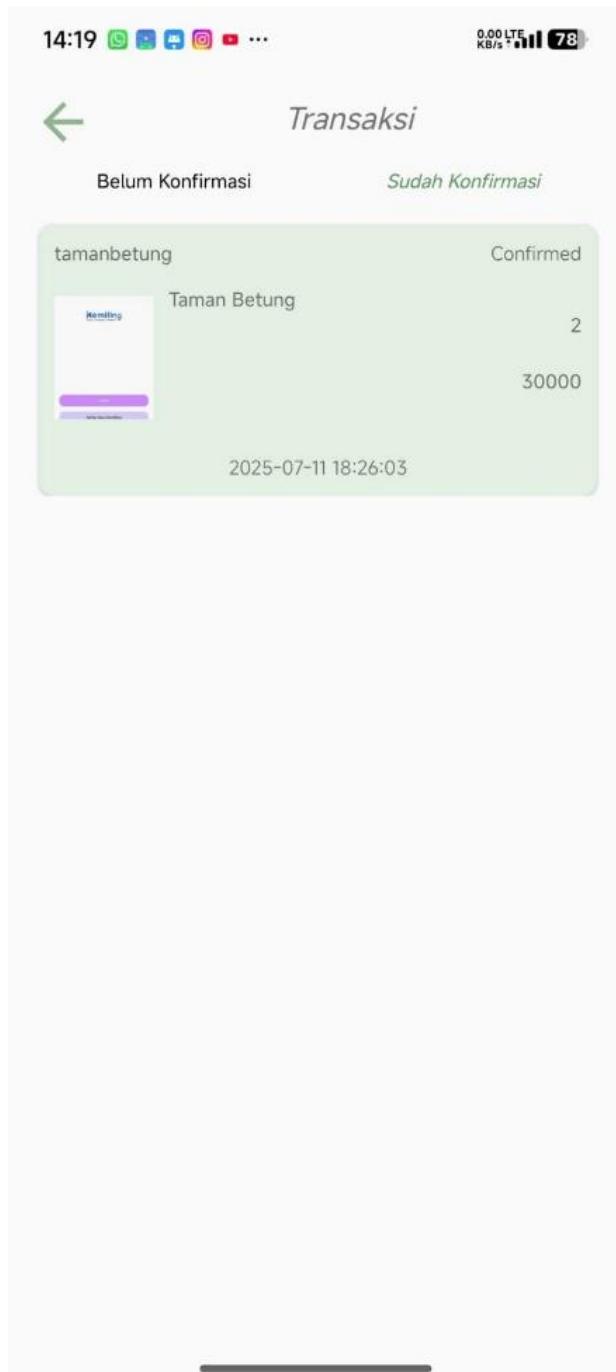


Gambar 4.55 Halaman Lokasi

#### 4.6.9 Halaman Status Transaksi

Implementasi halaman status Transaksi yang dirancang agar pengguna dapat melihat status pesanan atau transaksi mereka. Implementasi ini

memberikan transparansi kepada pengguna mengenai alur pemesanan mereka.

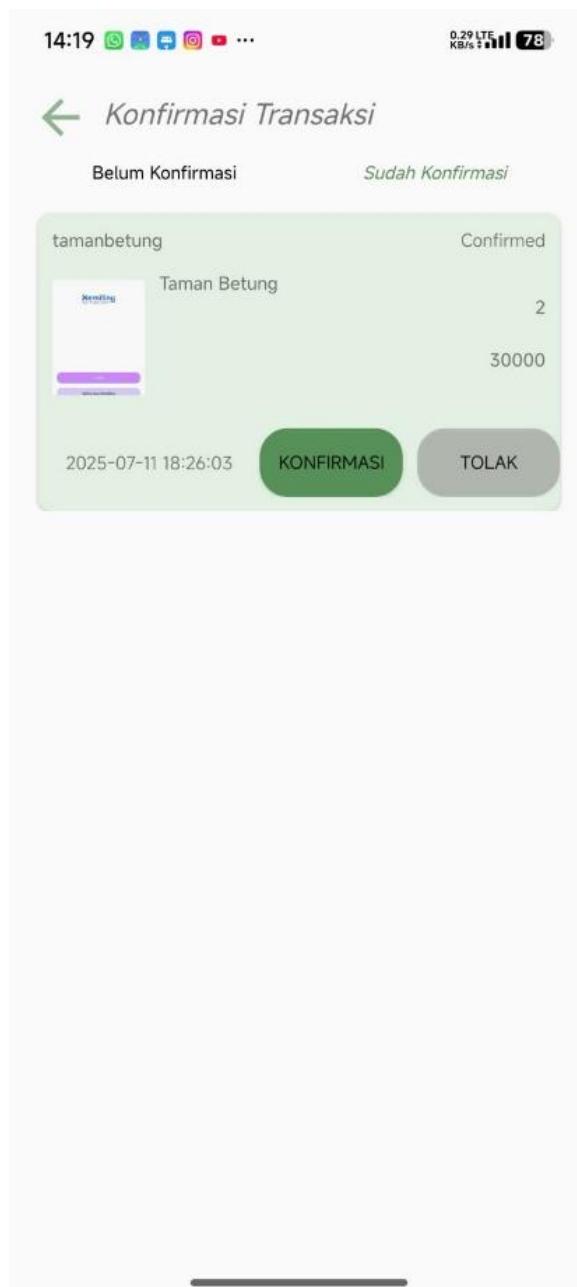


Gambar 4.56 Halaman Status Transaksi

#### 4.6.10 Halaman Konfirmasi Transaksi

Implementasi halaman ini merupakan pusat pengelolaan pesanan yang masuk. Untuk mempermudah manajemen, halaman ini dilengkapi

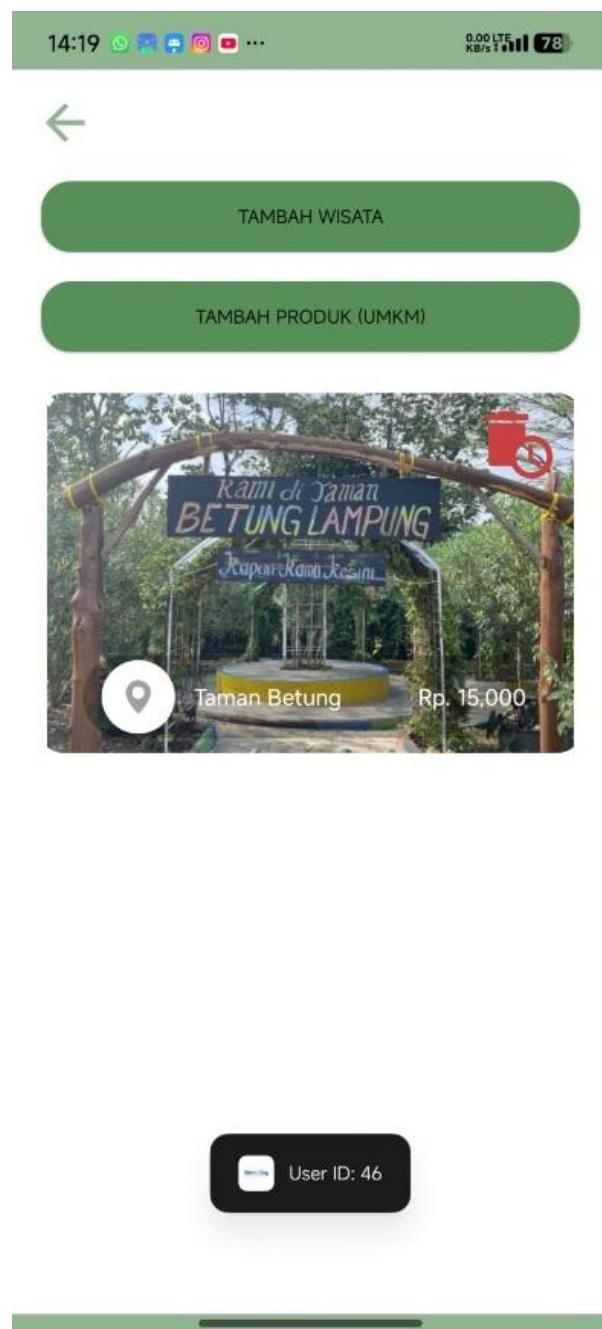
dengan sistem tab yang memisahkan transaksi berdasarkan statusnya, yaitu "Belum Konfirmasi" dan "Sudah Konfirmasi". Fungsi inti dari halaman ini adalah tombol aksi "Konfirmasi" dan "Tolak", yang memungkinkan Pelaku Usaha untuk menyetujui atau menolak pesanan secara langsung, sehingga alur validasi transaksi dapat berjalan dengan efisien.



Gambar 4.57 Halaman Konfirmasi Transaksi

#### 4.6.11 Halaman Produk Saya

Implementasi halaman ini dirancang sebagai dasbor bagi pelaku usaha untuk mengelola konten mereka. Dibagian atas terdapat dua tombol yang dapat menambahkan produk konten baru, baik Wisata maupun produk UMKM. Dan memungkinkan pengguna untuk menghapus data produk mereka.



Gambar 4.58 Halaman Produk Saya

#### 4.6.12 Halaman Tambah Produk

Implementasi halaman ini dirancang sebagai sebuah formulir untuk pelaku usaha bisa mendaftarkan destinasi wisata atau produk umkm baru. Dengan mencakup kolom input yang terstruktur dan beberapa tombol untuk pilih titik lokasi dan menyimpan data produk ke *Database*.

The screenshot displays the 'Tambah Produk' (Add Product) page of a mobile application. At the top, there are two status bars showing the time as 14:19 and signal strength. The main interface consists of several input fields and buttons:

- Image Header:** A placeholder with a magnifying glass icon and the text 'Image Header'.
- Deskripsi Wisata:** A placeholder with the text 'Deskripsi Wisata'.
- Pilih Gambar Header:** A green button labeled 'PILIH GAMBAR HEADER'.
- Additional Images:** Three icons representing additional images.
- Pilih Lokasi Maps:** A blue button labeled 'PILIH LOKASI MAPS'.
- Upload All Data:** A blue button labeled 'UPLOAD ALL DATA'.

Below these controls are the input fields:

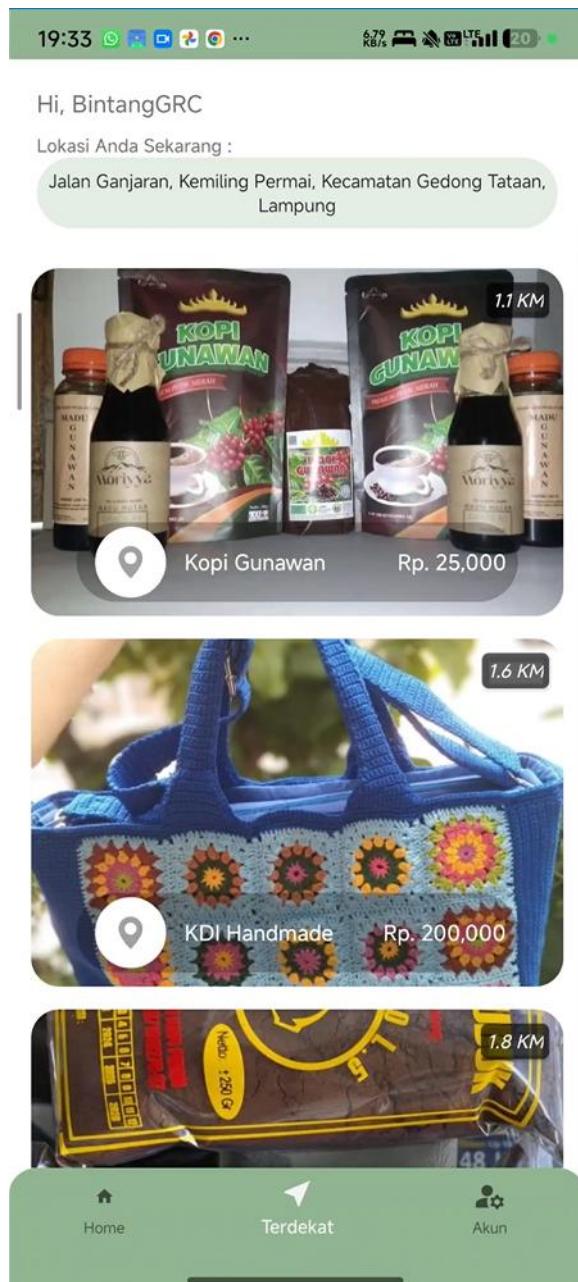
- Nama Wisata:** Placeholder 'Wisata'.
- Harga Weekday:** Placeholder 'Rp.0'.
- Harga Weekend:** Placeholder 'Rp.0'.
- Jam Operasional:** Placeholder 'Contoh : 09:00 - 18:00'.
- Alamat:** Placeholder 'Alamat'.
- No WhatsApp:** Placeholder '+62'.
- Deskripsi:** Placeholder 'Deskripsi Wisata'.
- Image Header:** A green button labeled 'Image Header'.

Gambar 4.59 Halaman Tambah Produk

#### 4.6.13 Halaman Hasil Algoritma Dijkstra

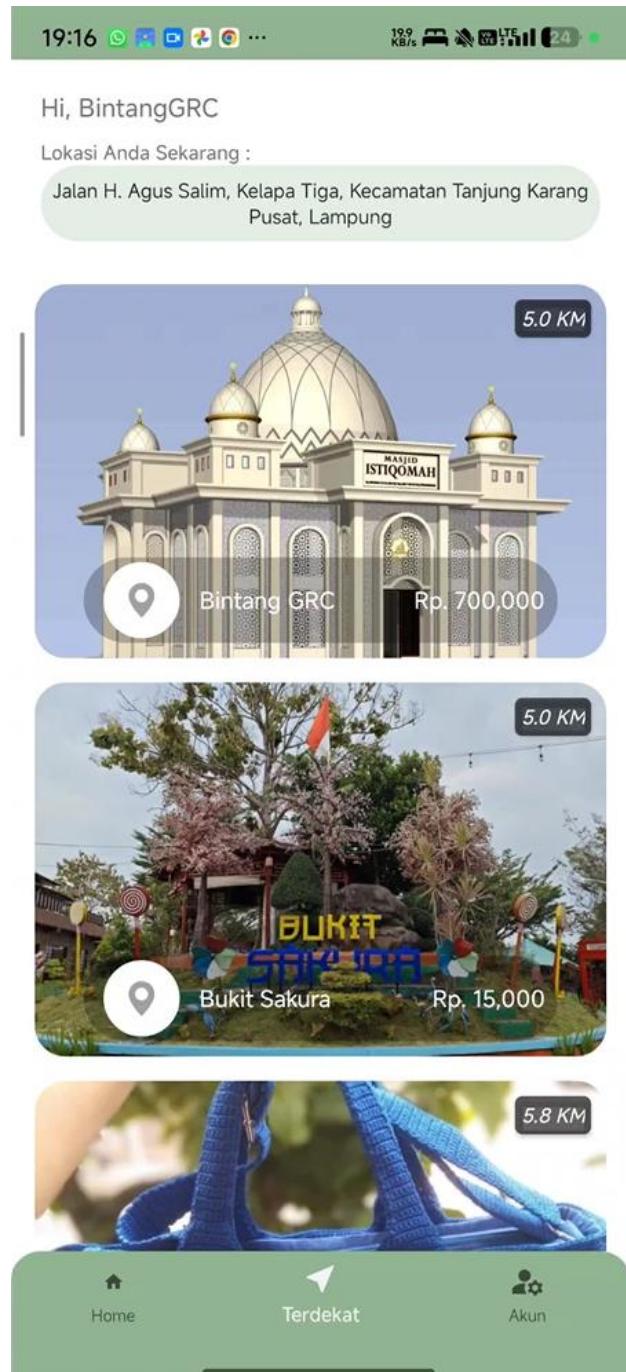
Berdasarkan implementasi Algoritma Dijkstra pada kode PHP yang sudah di terapkan sebelumnya, sistem berhasil menampilkan

rekomendasi terdekat dari lokasi pengguna secara *real-time*, dan berikut ini merupakan beberapa contoh rekomendasi tujuan dari berbagai tempat.



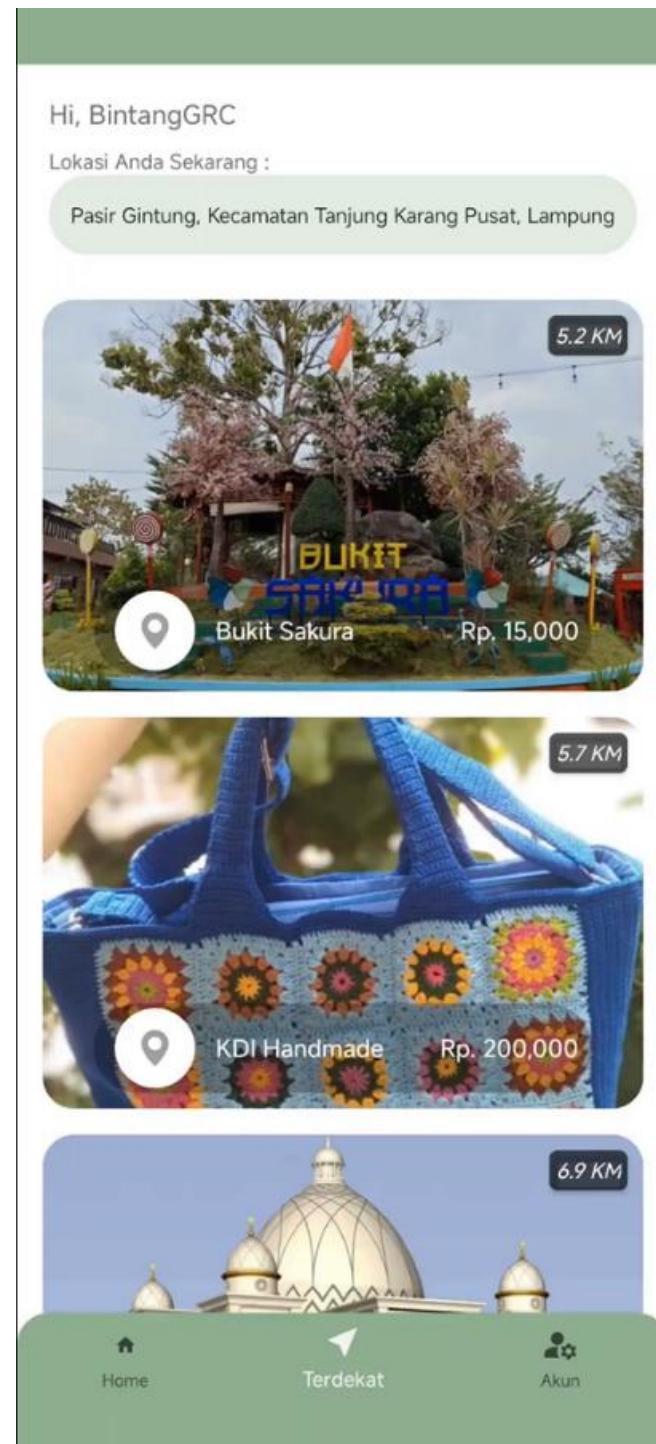
Gambar 4.60 Lokasi Jalan Ganjaran Kemiling

Dari lokasi ini, algoritma merekomendasikan Kopi Gunawan (1.1 KM) sebagai destinasi terdekat, diikuti oleh KDI Handmade (1.6 KM), menunjukkan hasil pengurutan jarak yang benar.



Gambar 4.62 Lokasi Jalan H. Agus Salim

Dari lokasi ini, algoritma dijkstra merekomendasikan Bintang GRC dan Bukit Sakura sebagai destinasi terdekat.



Gambar 4.63 Lokasi Pasar Gintung

Dan dari lokasi ini, algoritma dikstra merekomendasikan Bukit Sakura sebagai destinasi terdekat yang dapat di kunjungi pengguna berdasarkan lokasi pengguna saat itu.

#### 4.7 Pengujian Sistem

Berikut ini adalah tabel pengujian sistem yang sudah berjalan, tabel ini mencakup scenario pengujian dengan hasil yang di harapkan untuk memastikan setiap fungsi berjalan dengan baik sesuai rancangan.

**Tabel 4.7 Pengujian Sistem**

Skenario Pengujian	Hasil Pengujian dan Hasil yang Diharapkan	Kesimpulan
Registrasi Pengguna Baru	 <p>Deskripsi : Berhasil menampilkan halaman Register sesuai yang diharapkan</p>	Sesuai dengan yang diharapkan
Login Pengguna	 <p>Deskripsi : Berhasil menampilkan halaman Login sesuai yang diharapkan</p>	Sesuai dengan yang diharapkan
Login dengan Data Salah	 <p>Deskripsi : Berhasil menampilkan halaman Login dengan data yang salah dan sesuai yang diharapkan</p>	Sesuai dengan yang diharapkan

Melihat Daftar Produk		Deskripsi : Berhasil menampilkan halaman Register sesuai yang diharapkan	Sesuai dengan yang diharapkan
Melihat Algoritma Bekerja		Deskripsi : Berhasil menampilkan halaman Terdekat tempat algoritma bekerja sesuai yang di harapkan	Sesuai dengan yang diharapkan
Melihat Profil Pengguna		Deskripsi : Berhasil menampilkan halaman Profil sesuai yang diharapkan	Sesuai dengan yang diharapkan

#### 4.8 Distribusi Aplikasi

Untuk mempermudah proses penyebaran aplikasi, berkas instalasi diunggah ke layanan penyimpanan awan Google drive, Sehingga pengguna dapat mengunduh dan menggunakan aplikasi. Aplikasi dapat di download pada link berikut.

Link : [https://drive.google.com/drive/folders/1WanW7MtLZ5NuVY-eODb3iewMW\\_i9zy2S?usp=sharing](https://drive.google.com/drive/folders/1WanW7MtLZ5NuVY-eODb3iewMW_i9zy2S?usp=sharing)

atau di scan pada barcode yang mengarahkan ke tempat aplikasi di upload dibawah ini.



*Gambar 4.64 Barcode Aplikasi*