4. IMPLEMENTASI SISTEM

4.1. Perangkat Lunak yang digunakan

Aplikasi Nunut pada penelitian ini dibuat dengan menggunakan Flutter SDK versi 3.3.6 Aplikasi Nunut dalam penelitian ini dibuat menggunakan Flutter SDK versi 3.3.6 dengan bahasa pemrograman Dart dan Cursor sebagai *source code editor*. Data-data pada aplikasi disimpan dalam database Firebase.

4.2. Langkah-Langkah Implementasi

Berikut merupakan langkah-langkah yang dilakukan dalam melakukan implementasi program :

1. Melakukan instalasi SDK Flutter Dart.
2. Membuat tampilan desain *user interface* aplikasi dengan fitur terbaru menggunakan Figma.
3. Mengimplementasikan desain yang telah dibuat dari Figma kedalam Flutter.
4. Membuat konfigurasi logika fitur dan integrasi database untuk menyimpan data kedalam Firebase.
5. Melakukan modifikasi API yang sudah ada untuk mengambil data yang ada dalam database.

4.3. Instalasi Perangkat Lunak

Pada bagian ini akan dibahas mengenai perangkat lunak yang diperlukan untuk membuat dan mengimplementasikan program, yaitu instalasi Flutter.

4.3.1. Instalasi Flutter SDK

Berikut adalah langkah-langkah untuk melakukan instalasi Flutter SDK :

1. Mengunduh *bundle* instalasi Flutter SDK dari situs web https://docs.flutter.dev/get-started/install.
2. Mengekstrak file yang telah diunduh ke lokasi yang ditentukan.
3. Menambahkan path lokasi folder bin Flutter SDK ke dalam *system environment variables* di sistem operasi Windows (*Environment Variables > Path*).
4. Unduh dan instal ekstensi Dart dan Flutter pada editor kode Cursor.

4.4. Implementasi Program

Pada bagian ini akan menampilkan implementasi program berupa *source code* berdasarkan pembahasan dan diagram aktivitas yang telah dibuat pada BAB 3. Hubungan antara program dan rincian pembahasan diagram aktivitas dapat dilihat pada tabel 4.1.

Tabel 4. 1 Tabel Hubungan Program dan Activity Diagram

Nama	Segmen Program	Activity Diagram
Lupa Password	Segment Program 4.1	Gambar 3.2
OTP & Recaptcha	Segment Program 4.2	Gambar 3.3
Referral	Segment Program 4.3	Gambar 3.4
Buat Permintaan Tumpangan	Segment Program 4.4	Gambar 3.5
Halaman Request Tumpangan	Segment Program 4.5	Gambar 3.5
List Permintaan Tumpangan User	Segment Program 4.6	Gambar 3.6
List Tawaran Tumpangan	Segment Program 4.7	Gambar 3.7
Forum Chat	Segment Program 4.8	Gambar 3.8
List Chat Nunut	Segment Program 4.9	Gambar 3.8
Negosiasi Harga	Segment Program 4.10	Gambar 3.7
Google Maps API	Segment Program 4.11	Gambar 3.5

4.4.1. Lupa Password

Fitur ini memungkinkan user untuk mengubah kata sandi nya ketika user tidak dapat mengingat kata sandi yang sebelumnya pernah digunakan. *Source code* untuk fitur ini dapat dilihat pada Segmen Program 4.1.

Segmen Program 4. 1 Implementasi Lupa Password

```
Future passwordReset() async {
    try {
      await FirebaseAuth.instance.sendPasswordResetEmail(email:
email.text.trim());
      Fluttertoast.showToast(
```

```
        msg: "Link reset password telah dikirim, mohon cek email anda
untuk mereset password",
        toastLength: Toast.LENGTH_SHORT,
        gravity: ToastGravity.BOTTOM,
        timeInSecForIosWeb: 4,
        backgroundColor: Colors.green,
        textColor: Colors.white,
        fontSize: 16.0
      );
      Future.delayed(
        Duration(seconds: 2),
        () {
          Navigator.pushNamedAndRemoveUntil(context, '/login', (route)
=> false);
        },
      );
    } on FirebaseAuthException catch (e) {
      Fluttertoast.showToast(
        msg: e.message.toString(),
        toastLength: Toast.LENGTH_SHORT,
        gravity: ToastGravity.BOTTOM,
        timeInSecForIosWeb: 4,
        backgroundColor: Colors.red,
        textColor: Colors.white,
        fontSize: 16.0
      );
    }
  }
}

Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: nunutPrimaryColor,
      body: Center(
        child: Container(
          margin: const EdgeInsets.only(right: 24, left: 24),
          padding: const EdgeInsets.all(16),
          height: 700,
          decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(16),
            color: Colors.white,
          ),
          child: SingleChildScrollView(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.center,
```

```
              mainAxisSize: MainAxisSize.min,
              children: [
                Container(
                  margin: const EdgeInsets.only(top: 24, bottom: 16),
                  child: Image.asset(
                    "assets/icon.png",
                    width: 100,
                    height: 100,
                  ),
                ),
                Padding(
                  padding: const EdgeInsets.fromLTRB(0, 0, 0, 20),
                  child: NunutText(title: "Lupa Password", isTitle:
true, size: 32),
                ),
                InkWell(
                  child: NunutText(
                    title: "Masukkan email anda untuk mengubah
password",
                    size: 15,
                    color: greyFontColor,
                    fontWeight: FontWeight.w800,
                    textAlign: TextAlign.center,
                  ),
                ),
                NunutTextFormField(
                  title: "",
                  hintText: "Example@email.com",
                  obsecureText: false,
                  controller: email,
                  width: 1.5,
                ),

                NunutButton(
                  title: "Ubah Password",
                  widthButton: 200,
                  onPressed: passwordReset,
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
```

4.4.2. OTP & Recaptcha

Fitur ini memungkinkan user meningkatkan keamanan akunnya dengan adanya kewajiban bagi pengguna untuk memverifikasi nomor teleponnya melalui OTP dan Recaptcha yang diberikan oleh sistem. *Source code* untuk fitur ini dapat dilihat pada Segmen Program 4.2.

Segmen Program 4. 2 Implementasi OTP & Recaptcha

```
void handleSubmit(BuildContext context, pin, verificationId) {
   AuthService.checkVerifyPhone(otp: pin, verifyId:
verificationId).then((value) {
     print(value);
     if (value == true) {
       Navigator.pop(context);
       showDialog(
         context: context,
         builder: (BuildContext context) {
           return PopUpVerificationSuccess(
             title: "Nomor Telepon",
             subtitle: "Berhasil Terverifikasi!",
             onPressed: () {
               Navigator.pushNamedAndRemoveUntil(context, '/main',
(route) => false);
             },
           );
         },
       );
     } else {
       print("Gagal");
     }
   });
 }

static Future sentOtp({
   required String phone,
   required Function errorStep,
   required Function nextStep,
 }) async {
   await auth.setSettings(appVerificationDisabledForTesting: true);
   await auth
       .verifyPhoneNumber(
     timeout: Duration(seconds: 30),
     phoneNumber: "+62$phone",
     verificationCompleted: (phoneAuthCredential) async {
```

```
      print('complete');
      return;
    },
    verificationFailed: (error) async {
      print(error);
      return;
    },
    codeSent: (verificationId, forceResendingToken) async {
      print(verificationId);
      nextStep(verificationId);
    },
    codeAutoRetrievalTimeout: (verificationId) async {
      return;
    },
  )
      .onError((error, stackTrace) {
    errorStep();
  });
 }


 static Future checkVerifyPhone({required String otp, required String
verifyId}) async {
  final cred =
      PhoneAuthProvider.credential(verificationId: verifyId, smsCode:
otp);


  try {
    UserService().verifyPhone(uid: config.user.id!);
    return true;
  } on FirebaseAuthException catch (e) {
    return e.message.toString();
  } catch (e) {
    return e.toString();
  }
 }

Widget build(BuildContext context) {
    final args = ModalRoute.of(context)!.settings.arguments as
Map<String, dynamic>;
    final String verificationId = args['verificationId'];
    final String phone = args['phone'];

    final defaultPinTheme = PinTheme(
```

```
      width: 56,
      height: 60,
      textStyle: const TextStyle(
        fontSize: 22,
        color: Colors.black,
        fontWeight: FontWeight.bold,
      ),
      decoration: BoxDecoration(
        color: nunutLightColor,
        borderRadius: BorderRadius.circular(8),
        border: Border.all(color: Colors.transparent),
      ),
    );
    return Scaffold(
      backgroundColor: nunutPrimaryColor,
      body: Center(
        child: Container(
          margin: const EdgeInsets.only(right: 24, left: 24),
          padding: const EdgeInsets.all(16),
          height: 700,
          decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(16),
            color: Colors.white,
          ),
          child: SingleChildScrollView(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              mainAxisSize: MainAxisSize.min,
              children: [
                Container(
                  margin: const EdgeInsets.only(top: 24, bottom: 16),
                  child: Image.asset(
                    "assets/icon.png",
                    width: 100,
                    height: 100,
                  ),
                ),
                Padding(
                  padding: const EdgeInsets.fromLTRB(0, 0, 0, 20),
                  child: NunutText(title: "OTP Verification", isTitle:
true, size: 32),
                ),
                InkWell(
                  child: NunutText(
                    title: "Kode telah dikirim ke nomor",
```

```
                    size: 15,
                    color: greyFontColor,
                    fontWeight: FontWeight.w600,
                    textAlign: TextAlign.center,
                  ),
                ),
                InkWell(
                  child: NunutText(
                    title: "+62 $phone",
                    size: 15,
                    color: greyFontColor,
                    fontWeight: FontWeight.w800,
                    textAlign: TextAlign.center,
                  ),
                ),
                SizedBox(height: 24),
                Pinput(
                  length: 6,
                  defaultPinTheme: defaultPinTheme,
                  focusedPinTheme: defaultPinTheme.copyWith(
                    decoration: defaultPinTheme.decoration!.copyWith(
                      border: Border.all(color: nunutPrimaryColor,
width: 2),
                    ),
                  ),
                  onCompleted: (pin) {
                    handleSubmit(context, pin, verificationId);
                  }
                ),
                SizedBox(width: 340,)
              ]
            )
          )
        )
      )
    );
  }
```

4.4.3. Referral

Fitur ini memungkinkan user untuk menarik pengguna untuk menggunakan aplikasi Nunut yang terbaru dan mengajak temannya untuk menggunakan aplikasi Nunut. Fitur ini akan memberikan *reward* berupa saldo Nunut Pay sebesar Rp 5,000 kepada pengguna yang mengklaim kode referral

temannya dan pemilik kode referral tersebut. Source code untuk fitur ini dapat dilihat pada Segmen Program 4.3.

Segmen Program 4. 3 Implementasi Referral

```
Future<void> _initializeReferralCode() async {
   try {
     String? code = await referralApi.checkCode(config.user.id!);
     String? sisaClaim = await referralApi.checkClaim(config.user.id!);


     if (code != "") {
       setState(() {
         referralCode = code;
         sisa = sisaClaim;
       });
     } else {
       code = await referralApi.generateCode(config.user.id!);
       sisaClaim = await referralApi.checkClaim(config.user.id!);
       setState(() {
         referralCode = code;
         sisa = sisaClaim;
       });
     }
   } catch (error) {
     print("Error initializing referral code: $error");
   }
 }

class ProfilePageMenu {
 String title;
 String icon;
 String identifier;
 ProfilePageMenu({required this.title, required this.icon, required
this.identifier});
}

class ReferralApi {
 FirebaseFirestore firestore = FirebaseFirestore.instance;
 Random random = Random();
 String generatedCode = '';
  Future<String> generateCode(String user_id) async {
 // Generate a random 5-character code
   const String chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
   generatedCode = List.generate(5, (index) =>
```

```
chars[random.nextInt(chars.length)]).join();

   // Check if the code is unique
   bool isUnique = await checkCodeUnique(generatedCode);

   // If not unique, generate another one
   if (!isUnique) {
     return generateCode(user_id);
   } else {
     saveReferralCode(generatedCode, user_id);
   }

   return generatedCode;
 }


 Future<void> claimReferral(String user_id, String code, BuildContext
context) async {
   try {
     final DocumentSnapshot result = await
firestore.collection('referral').doc(user_id).get();
     final DocumentSnapshot walletClaimer = await
firestore.collection('wallet').doc(user_id).get();
     final QuerySnapshot querySnapshot = await
firestore.collection('referral').where('code', isEqualTo: code).get();

     String userCode = "";

     if (querySnapshot.docs.isEmpty) {
       showSnackBar(context, "Kode Referral tidak valid!");
     } else {
       int? claimerIndex;


       querySnapshot.docs.forEach((doc) {
         userCode = doc.id;
       });

       final DocumentSnapshot walletUser = await
firestore.collection('wallet').doc(userCode).get();


       for (int i = 1; i <= 5; i++) {

         if (result['claimer$i'] == userCode) {
```

```
            showSnackBar(context, "Kode Referral sudah pernah
digunakan!");
            print("Code already claimed");
            break;
          }
          if (result['claimer$i'] == "") {
            claimerIndex = i;
            showSnackBar(context, "Kode referral berhasil di klaim");
            if (claimerIndex != null) {
              int sisaClaim = int.parse(result['sisaClaim']);
              sisaClaim -= 1;
              print(sisaClaim);


              await firestore
                .collection("referral")
                .doc(user_id)
                .update({"claimer$claimerIndex": userCode, "sisaClaim":
sisaClaim.toString()});


              await firestore
                .collection("wallet")
                .doc(user_id)
                .update({"balance": (walletClaimer['balance']) + 5000});


              await firestore
                .collection("wallet")
                .doc(userCode)
                .update({"balance": (walletUser['balance']) + 5000});

              print("Document updated successfully");
              break;
            } else {
              showSnackBar(context, "Sisa klaim kode referral sudah
habis!");
              print("All claimers are already filled");
            }
          }
        }
      }
    } catch (e) {
      showSnackBar(context, "Terjadi kesalahan. Silakan coba lagi.");
      print("Error: $e");
```

```
  }
 }


 void showSnackBar(BuildContext context, String message) {
   ScaffoldMessenger.of(context).showSnackBar(
     SnackBar(
       content: Text(message),
     ),
   );
 }


 Future<String> checkCode(String user_id) async {
   final DocumentSnapshot result = await
firestore.collection('referral').doc(user_id).get();

   if (result.data() == null) {
     return "";
   } else {
     return result['code'];
   }
 }

 Future<String> checkClaim(String user_id) async {
   final DocumentSnapshot result = await
firestore.collection('referral').doc(user_id).get();

   if (result.data() == null) {
     return "";
   } else {
     return result['sisaClaim'];
   }
 }

 Future<bool> checkCodeUnique(String code) async {
   final QuerySnapshot result = await
firestore.collection('referral').get();
   final List<DocumentSnapshot> documents = result.docs.where((element)
=> element['code'] == code).toList();

   if (documents.isEmpty) {
     return true;
   } else {
     return false;
```

```
   }
 }

 Future<Widget> showClaimReferral(BuildContext context, String user_id)
async {
   List<ProfilePageMenu> profilePageMenu = [];
   List<ProfilePageMenu> profilePageMenu2 = [];

   final String myCode = user_id;
   final DocumentSnapshot result = await
firestore.collection('referral').doc(user_id).get();
   final List<String> list = [];


   for (int i = 1; i <= 5; i++) {
     if (result['claimer$i'] != "") {
       final DocumentSnapshot userName = await
firestore.collection('users').doc(result['claimer$i']).get();

       profilePageMenu.add(ProfilePageMenu(
         title: userName['name'],
         icon: "assets/icons/referral.png",
         identifier: "claimReferral",
       ));
     }
   }

   final QuerySnapshot querySnapshot = await
firestore.collection('referral').get();
   String userIdInDocs;

   querySnapshot.docs.forEach((doc) {
     if (doc.data() != null && (doc.data() as
Map).containsValue(user_id)) {
       userIdInDocs = doc.id.toString();
       list.add(doc.id.toString());
     }
   });
   for (int i = 0; i < list.length; i++){
     final DocumentSnapshot userName = await
firestore.collection('users').doc(list[i]).get();
       profilePageMenu2.add(ProfilePageMenu(
         title: userName['name'],
         icon: "assets/icons/referral.png",
         identifier: "DiclaimReferral",
```

```
      ));
   }
   return Dialog(....)
}

Future<void> saveReferralCode(String code, String user_id) async {
   final data = <String, String>{
     "code": code,
     "claimer1": "",
     "claimer2": "",
     "claimer3": "",
     "claimer4": "",
     "claimer5": "",
     "sisaClaim": "5",
   };

   firestore
   .collection("referral")
   .doc(user_id)
   .set(data)
   .onError((e, _) => print("Error writing document: $e"));
 }
}

Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: Stack(
          children: [
            Align(
              alignment: Alignment.topCenter,
              child: Image.asset(
                "assets/lingkaranKuning_2.png",
                width: MediaQuery.of(context).size.width * 0.815,
              ),
            ),
            Positioned(
              top: MediaQuery.of(context).size.height * 0.085,
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  IconButton(
                    onPressed: () {
                      Navigator.pop(context);
                    },
```

```
                    icon: Icon(Icons.arrow_back),
                  ),
                  SizedBox(height: 10),
                  Container(
                    margin: EdgeInsets.only(left: 18),
                    child:
                        NunutText(title: "Referral", size: 32, isTitle:
true),
                  ),
                ],
              ),
            ),
            Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              mainAxisSize: MainAxisSize.min,
              children: [
                Container(
                  margin: const EdgeInsets.only(top: 124, bottom: 16),
                ),
                SizedBox(height: 70),
                Container(
                  padding: EdgeInsets.symmetric(horizontal: 30),
                  child: NunutTextFormField(
                    title: "Kode Referral Mu",
                    hintText: referralCode!,
                    hintStyle: TextStyle(
                      color: Colors.black,
                      fontWeight: FontWeight.w800,
                    ),
                    obsecureText: false,
                    textAlign: TextAlign.center,
                    controller: referral_code,
                    color: nunutPrimaryColor,
                    readOnly: true,
                    isSuffixIcon: false,
                  ),
                ),
                SizedBox(height: 20),
                Container(
                  padding: EdgeInsets.symmetric(horizontal: 30),
                  child: NunutTextFormField(
                    title: "Klaim Referral Teman",
                    hintText: "Masukkan kode referral teman",
                    obsecureText: false,
                    textAlign: TextAlign.center,
```

```
                    controller: claim_referral,
                    color: nunutPrimaryColor,
                    readOnly: false,
                    isSuffixIcon: false,
                  ),
                ),
                SizedBox(height: 5),
                NunutButton(
                  title: "Klaim Kode",
                  widthButton: 200,
                  onPressed: () {
                    if (claim_referral.text == referralCode) {
                      ScaffoldMessenger.of(context).showSnackBar(
                        SnackBar(
                          content: Text(
                              "Kode referral tidak boleh sama dengan
kode referral kamu"),
                        ),
                      );
                    } else if (claim_referral.text == "") {
                      ScaffoldMessenger.of(context).showSnackBar(
                        SnackBar(
                          content: Text("Kode referral tidak boleh
kosong"),
                        ),
                      );
                    } else {
                      if (sisa == "0") {
                        ScaffoldMessenger.of(context).showSnackBar(
                          SnackBar(
                            content:
                                Text("Sisa klaim referral kamu sudah
habis"),
                          ),
                        );
                      } else {
                        showDialog(
                          context: context,
                          builder: (BuildContext context) {
                            return AlertDialog(
                              title: Text("Klaim Kode Referral"),
                              content: Text(
                                  "Apakah kamu yakin ingin mengklaim
kode referral ini?"),
                              actions: [
```

```
                                TextButton(
                                  onPressed: () {
                                    Navigator.pop(context);
                                  },
                                  child: Text("Tidak"),
                                ),
                                TextButton(
                                  onPressed: () async {
                                    await referralApi.claimReferral(
                                        config.user.id!,
                                        claim_referral.text,
                                        context);
                                    _initializeReferralCode();
                                    Navigator.pop(context);
                                  },
                                  child: Text("Ya"),
                                ),
                              ],
                            );
                          },
                        );
                      }
                    }
                  },
                ),
                SizedBox(height: 30),
                Container(
                  padding: EdgeInsets.symmetric(horizontal: 30),
                  alignment: Alignment.centerLeft,
                  child: Text(
                    "Sisa Klaim : $sisa",
                    style: TextStyle(
                      color: Colors.black,
                      fontSize: 15,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ),
                SizedBox(height: 10),
                FutureBuilder(
                  future:
                      referralApi.showClaimReferral(context,
config.user.id!),
                  builder: (context, snapshot) {
                    if (snapshot.hasData) {
```

```
                    return snapshot.data!;
                  } else {
                    return Container();
                  }
                },
              ),
              SizedBox(height: 50),
            ],
          ),
        ],
      ),
    ),
  );
}
```

4.4.4. Buat Permintaan Tumpangan

Fitur ini memungkinkan user untuk membuat permintaan tumpangan yang mereka inginkan dari Universitas Kristen Petra ke tujuan yang diinginkan maupun sebaliknya. *Driver* dapat melihat semua permintaan pengguna yang telah dibuat pada halaman list permintaan. Source code untuk fitur ini dapat dilihat pada Segmen Program 4.4.

Segmen Program 4. 4 Implementasi Buat Permintaan Tumpangan

```
Future<String> createRoom(String user_id, String roomTitle) async {
    try {
      String roomId = firestore.collection('chats').doc().id;
      DocumentSnapshot userName =
          await firestore.collection('users').doc(user_id).get();

      List<String> memberIds = [
        '$user_id',
      ];

      // Check if 'image' field exists in the user document
      String imageUrl = userName.exists &&
              userName.data() is Map<String, dynamic> &&
              (userName.data() as Map<String,
dynamic>).containsKey('image')
          ? userName['image']
          :
'https://firebasestorage.googleapis.com/v0/b/nunut-da274.appspot.com/o/a
vatar.png?alt=media&token=62dfdb20-7aa0-4ca4-badf-31c282583b1b';
```

```
      await firestore.collection('chats').doc(roomId).set({
        'chatRoomTitle': roomTitle,
        'id': roomId,
        'isRequest': true,
        'memberIds': memberIds,
        'createdBy': user_id,
        'members': {
          memberIds[0]: {
            'imageUrl': imageUrl,
            'isDriver': false,
            'isNunut': false,
            'name': userName['name']
          },
        }
      });

      return roomId;
    } catch (e) {
      print("Error creating room: $e");
      return e.toString();
    }
  }

static Future PostRideScheduleUser(
      date, time, meeting_point_id, destination_id, user_id, chatRoomId)
async {
    var url = Uri.parse(config.baseUrl + '/ride-schedule-user/');
    var meetingPoint = await
mapLocationApi.getMapListById(meeting_point_id);
    var destination = await
mapLocationApi.getMapListById(destination_id);
    var body = {
      'date': date,
      'time': time,
      'meeting_point': meetingPoint.toJson(),
      'destination': destination.toJson(),
      'note': "Testing",
      'user_id': user_id,
      'is_active': true,
      'chatRoomId': chatRoomId,
    };
    var bodyJson = json.encode(body);

    var response = await http.post(
      url,
```

```
      headers: {
        'Content-Type': 'application/json',
        'Accept': 'application/json',
        'Authorization': 'Bearer ${config.user.token}',
      },
      body: bodyJson,
    );

    Result result =
Result.fromJson(json.decode(response.body.toString()));
    if (result.status == 200) {
      return true;
    } else {
      return false;
    }
  }

Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: Container(
          padding: EdgeInsets.all(24),
          margin: EdgeInsets.only(top: 50),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              NunutText(
                title: "Minta Tumpangan Baru",
                color: Colors.black,
                size: 20,
                fontWeight: FontWeight.bold,
              ),
              SizedBox(height: 15),
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Flexible(
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        NunutText(
                          title: "Tanggal Berangkat",
                          fontWeight: FontWeight.bold,
                          size: 12,
                        ),
```

```
                        SizedBox(height: 10),
                        TextFormField(
                          style: TextStyle(
                            color: Colors.black,
                            fontSize: 12,
                          ),
                          cursorColor: Colors.black,
                          obscureText: false,
                          controller: _dateController,
                          onTap: () async {
                            FocusScope.of(context)
                                .requestFocus(new FocusNode());
                            DateTime? pickedDate = await showDatePicker(
                                context: context,
                                initialDate: DateTime.now(),
                                firstDate: DateTime.now(),
                                lastDate: DateTime(2101));

                            if (pickedDate != null) {
                              String formattedDate = DateFormat('MMMM
dd, yyyy')
                                  .format(pickedDate);
                              setState(() {
                                _dateController.text =
                                    formattedDate; //set output date to
TextField value.
                              });
                            }
                          },
                          decoration: InputDecoration(
                            contentPadding: EdgeInsets.symmetric(
                                horizontal: 10, vertical: 10),
                            isDense: true,
                            hintText: "dd/mm/yyyy",
                            hintStyle: TextStyle(
                              color: Colors.grey,
                              fontSize: 12,
                            ),
                            filled: true,
                            fillColor: Colors.grey[300],
                            border: OutlineInputBorder(
                              borderRadius: BorderRadius.circular(50),
                              borderSide: const BorderSide(
                                width: 0,
                                style: BorderStyle.none,
```

```
                              ),
                            ),
                            focusedBorder: OutlineInputBorder(
                              borderRadius: BorderRadius.circular(50),
                              borderSide: const BorderSide(
                                width: 0,
                                style: BorderStyle.none,
                              ),
                            ),
                          ),
                        ),
                      ],
                    ),
                  ),
                  SizedBox(width: 20),
                  Flexible(
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        NunutText(
                          title: "Jam Berangkat",
                          fontWeight: FontWeight.bold,
                          size: 12,
                        ),
                        SizedBox(height: 10),
                        TextFormField(
                          style: TextStyle(
                            color: Colors.black,
                            fontSize: 12,
                          ),
                          cursorColor: Colors.black,
                          obscureText: false,
                          controller: _timeController,
                          onTap: () async {
                            FocusScope.of(context)
                                .requestFocus(new FocusNode());
                            TimeOfDay? picked = await showTimePicker(
                              context: context,
                              initialTime: TimeOfDay.now(),
                              builder: (BuildContext context, Widget?
child) {
                                return MediaQuery(
                                  data: MediaQuery.of(context)
                                      .copyWith(alwaysUse24HourFormat:
true),
```

```
                                child: child!,
                              );
                            },
                          );
                          if (picked != null) {
                            setState(() {
                              _timeController.text = picked.hour
                                      .toString()
                                      .padLeft(2, '0') +
                                  ":" +
                                  picked.minute.toString().padLeft(2,
'0');
                            });
                          }
                        },
                        decoration: InputDecoration(
                          contentPadding: EdgeInsets.symmetric(
                              horizontal: 10, vertical: 10),
                          isDense: true,
                          hintText: "00:00",
                          hintStyle: TextStyle(
                            color: Colors.grey,
                            fontSize: 12,
                          ),
                          filled: true,
                          fillColor: Colors.grey[300],
                          border: OutlineInputBorder(
                            borderRadius: BorderRadius.circular(50),
                            borderSide: const BorderSide(
                              width: 0,
                              style: BorderStyle.none,
                            ),
                          ),
                          focusedBorder: OutlineInputBorder(
                            borderRadius: BorderRadius.circular(50),
                            borderSide: const BorderSide(
                              width: 0,
                              style: BorderStyle.none,
                            ),
                          ),
                        ),
                      ),
                    ],
                  ),
                ),
```

```
                ],
              ),
              SizedBox(height: 15),
              NunutText(
                title: "Meeting Point",
                fontWeight: FontWeight.bold,
                size: 12,
              ),
              SizedBox(height: 10),
              TextFormField(
                style: TextStyle(
                  color: Colors.black,
                  fontSize: 12,
                ),
                readOnly: true,
                cursorColor: Colors.black,
                obscureText: false,
                controller: _meetingPointController,
                decoration: InputDecoration(
                  contentPadding:
                      EdgeInsets.symmetric(horizontal: 10, vertical:
10),
                  isDense: true,
                  hintText: "Pilih meeting pointmu...",
                  hintStyle: TextStyle(
                    color: Colors.grey,
                    fontSize: 12,
                  ),
                  filled: true,
                  fillColor: Colors.grey[300],
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(50),
                    borderSide: const BorderSide(
                      width: 0,
                      style: BorderStyle.none,
                    ),
                  ),
                  focusedBorder: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(50),
                    borderSide: const BorderSide(
                      width: 0,
                      style: BorderStyle.none,
                    ),
                  ),
                ),
```

```
              onTap: () async {
                if (lockMeetingPoint) {
                  _destinationController.text = '';
                  _selectedDestination = '';
                  lockMeetingPoint = false;
                }
                final removeUKP = _destinationController.text
                    .contains('Universitas Kristen Petra');
                MapLocation result = await Navigator.push(
                  context,
                  MaterialPageRoute(
                    builder: (context) =>
                        MapList(removeUKP: false, sendId: true),
                  ),
                );
                setState(() {
                  if (result.name != null) {
                    _selectedMeetingPoint = result.mapId!;
                    _meetingPointController.text = result.name!;

                    if (!_meetingPointController.text
                        .contains('Universitas Kristen Petra')) {
                      _destinationController.text =
                          'Universitas Kristen Petra';
                      _selectedDestination = config.mapIdPetra;
                      lockDestination = true;
                    } else {
                      lockDestination = false;
                    }

                    if (_meetingPointController.text ==
                        _destinationController.text) {
                      _destinationController.text = '';
                      _selectedDestination = '';
                    }
                  }
                });
              },
            ),
            SizedBox(height: 15),
            NunutText(
              title: "Tujuan Destinasi",
              fontWeight: FontWeight.bold,
              size: 12,
            ),
```

```
              SizedBox(height: 10),
              TextFormField(
                style: TextStyle(
                  color: Colors.black,
                  fontSize: 12,
                ),
                readOnly: true,
                cursorColor: Colors.black,
                obscureText: false,
                controller: _destinationController,
                decoration: InputDecoration(
                  contentPadding:
                      EdgeInsets.symmetric(horizontal: 10, vertical:
10),
                  isDense: true,
                  hintText: "Pilih tujuan destinasimu...",
                  hintStyle: TextStyle(
                    color: Colors.grey,
                    fontSize: 12,
                  ),
                  filled: true,
                  fillColor: Colors.grey[300],
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(50),
                    borderSide: const BorderSide(
                      width: 0,
                      style: BorderStyle.none,
                    ),
                  ),
                  focusedBorder: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(50),
                    borderSide: const BorderSide(
                      width: 0,
                      style: BorderStyle.none,
                    ),
                  ),
                ),
                onTap: () async {
                  if (lockDestination) {
                    _meetingPointController.text = '';
                    _selectedMeetingPoint = '';
                    lockDestination = false;
                  }
                  final removeUKP = _meetingPointController.text
                      .contains('Universitas Kristen Petra');
```

```
                MapLocation result = await Navigator.push(
                  context,
                  MaterialPageRoute(
                    builder: (context) => MapList(
                      removeUKP: false,
                      sendId: true,
                    ),
                  ),
                );
                setState(() {
                  if (result.name != null) {
                    _selectedDestination = result.mapId!;
                    _destinationController.text = result.name!;
                    if (!_destinationController.text
                        .contains('Universitas Kristen Petra')) {
                      _meetingPointController.text =
                          'Universitas Kristen Petra';
                      _selectedMeetingPoint = config.mapIdPetra;
                      lockMeetingPoint = true;
                    } else {
                      lockMeetingPoint = false;
                    }

                    if (_destinationController.text ==
                        _meetingPointController.text) {
                      _meetingPointController.text = '';
                      _selectedMeetingPoint = '';
                    }
                  }
                });
              },
            ),
            SizedBox(height: 40),
            Align(
              alignment: Alignment.center,
              child: NunutButton(
                title: "Minta",
                onPressed: () async {
                  if (_dateController.text.isEmpty ||
                      _timeController.text.isEmpty ||
                      _meetingPointController.text.isEmpty ||
                      _destinationController.text.isEmpty) {
                    Fluttertoast.showToast(
                        msg: "Semua field harus diisi",
                        toastLength: Toast.LENGTH_SHORT,
```

```
                          gravity: ToastGravity.BOTTOM,
                          timeInSecForIosWeb: 1,
                          backgroundColor: Colors.red,
                          textColor: Colors.white,
                          fontSize: 16.0);
                      return;
                    }
                    String namaMeetingPoint =
                        _meetingPointController.text.toString();
                    String namaDestination =
                        _destinationController.text.toString();
                    chatRoomId = await
chatApi.createRoom(config.user.id!,
                        "$namaMeetingPoint" + " - " +
"$namaDestination");

                    var postRideScheduleStatus =
                        await RideScheduleApi.PostRideScheduleUser(
                      _dateController.text.toString(),
                      _timeController.text.toString(),
                      _selectedMeetingPoint,
                      _selectedDestination,
                      config.user.id,
                      chatRoomId,
                    );
                    if (postRideScheduleStatus == true) {
                      Fluttertoast.showToast(
                          msg: "Berhasil membuat jadwal",
                          toastLength: Toast.LENGTH_SHORT,
                          gravity: ToastGravity.BOTTOM,
                          timeInSecForIosWeb: 1,
                          backgroundColor: Colors.green,
                          textColor: Colors.white,
                          fontSize: 16.0);
                      //empty all textfield
                      setState(() {
                        _dateController.text = "";
                        _timeController.text = "";
                        _meetingPointController.text = "";
                        _destinationController.text = "";
                      });
                      Navigator.pushNamedAndRemoveUntil(
                          context, '/main', (route) => false);
                      Navigator.of(context).pushNamed('/rideListUser');
                    } else {
```

```
                    Fluttertoast.showToast(
                        msg: "Gagal membuat jadwal",
                        toastLength: Toast.LENGTH_SHORT,
                        gravity: ToastGravity.BOTTOM,
                        timeInSecForIosWeb: 1,
                        backgroundColor: Colors.red,
                        textColor: Colors.white,
                        fontSize: 16.0);
                  }
                },
                heightButton: 35,
                widthButton: 110,
                fontWeight: FontWeight.w500,
                widthBorder: 0.0,
                borderColor: Colors.transparent,
              ),
            ),
          ],
        ),
      ),
    ),
  );
}
```

4.4.5. Halaman Request Tumpangan

Bagian ini memungkinkan user untuk melihat halaman utama untuk pengguna ketika ingin melakukan permintaan tumpangan yang terdiri dari 2 tombol yaitu buat permintaan dan permintaanku. Source code untuk halaman ini dapat dilihat pada Segmen Program 4.5.

Segmen Program 4. 5 Implementasi Halaman Request Tumpangan

```
@override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          Positioned(
            child: Container(
              height: 200,
              width: double.infinity,
              decoration: BoxDecoration(
                borderRadius: BorderRadius.only(
```

```
                  bottomRight: Radius.circular(50),
                ),
                color: nunutPrimaryColor,
              ),
            ),
          ),
          Container(
            margin: EdgeInsets.only(top: 90, left: 28, right: 28),
            child: SingleChildScrollView(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  NunutText(
                    title: "Hai, ${config.user.name}",
                    fontWeight: FontWeight.bold,
                  ),
                  SizedBox(height: 10),
                  NunutText(title: "Yuk cari\ntumpangan!", isTitle:
true, size: 32),
                  SizedBox(height: 20),
                  SizedBox(
                    height: MediaQuery.of(context).size.height - 200,
                    child: GridView.count(
                      crossAxisCount: 2,
                      mainAxisSpacing: 50,
                      children: [
                        GestureDetector(
                          onTap: () {

Navigator.of(context).pushNamed('/addRideScheduleUser', arguments:
true);
                          },
                          child: Column(
                            children: [
                              Container(
                                padding: EdgeInsets.all(40),
                                decoration: BoxDecoration(
                                  borderRadius:
BorderRadius.circular(20),
                                  color: Colors.black,
                                ),
                                child: Icon(
                                  Icons.add,
                                  size: 50,
                                  color: Colors.white,
```

```
                                ),
                              ),
                              SizedBox(
                                height: 10,
                              ),
                              NunutText(
                                title: "Minta Tumpangan",
                                fontWeight: FontWeight.bold,
                                size: 16,
                              ),
                            ],
                          ),
                        ),
                        InkWell(
                          onTap: () {

Navigator.of(context).pushNamed('/rideListUser');
                          },
                          child: Column(
                            children: [
                              Container(
                                padding: EdgeInsets.all(40),
                                decoration: BoxDecoration(
                                  borderRadius:
BorderRadius.circular(20),
                                  color: nunutPrimaryColor,
                                ),
                                child: Icon(
                                  Icons.group,
                                  size: 50,
                                  color: Colors.black,
                                ),
                              ),
                              SizedBox(
                                height: 10,
                              ),
                              NunutText(
                                title: "Permintaanku",
                                fontWeight: FontWeight.bold,
                                size: 16,
                              ),
                            ],
                          ),
                        ),
                      ],
```

```
                  ),
                )
              ],
            ),
          ),
        ),
      ],
    ),
  );
 }
}
```

4.4.6. List Permintaan Tumpangan User

Fitur ini memungkinkan pengguna untuk melihat daftar permintaan yang telah dibuat oleh pengguna. Source code untuk fitur ini dapat dilihat pada Segmen Program 4.6.

Segmen Program 4. 6 Implementasi List Permintaan Tumpangan User

```
Future<List<RideScheduleUser>> getRideScheduleListUser(String user_id)
async {
    FirebaseFirestore firestore = FirebaseFirestore.instance;
    QuerySnapshot result = await firestore
        .collection('ride_schedule_user')
        .where('user_id', isEqualTo: user_id)
        .get();

    List<RideScheduleUser> rideScheduleList = [];

    for (var item in result.docs) {
      var data = item.data() as Map<String, dynamic>;
      data['ride_schedule_user_id'] = item.id;
      rideScheduleList.add(RideScheduleUser.fromJson(data));
    }

    return rideScheduleList;
  }

Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          Positioned(
            //topright
```

```
            top: 0,
            right: 0,
            child: Image(
              image:

AssetImage('assets/backgroundCircle/backgroundCircle3.png'),
              fit: BoxFit.cover,
            ),
          ),
          Container(
            margin: EdgeInsets.only(top: 40, left: 20, right: 20),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                IconButton(
                  padding: EdgeInsets.zero,
                  icon: Icon(Icons.arrow_back, color: Colors.black),
                  onPressed: () {
                    Navigator.pop(context);
                  },
                ),
                SizedBox(height: 20),
                NunutText(title: "Permintaanku", isTitle: true, size:
32),
                Container(
                  margin:
                      EdgeInsets.only(top: 40, left: 8, right: 24,
bottom: 10),
                ),
                isLoading
                  ? Center(child: CircularProgressIndicator(color:
nunutPrimaryColor))
                  : Expanded(
                      child: Column(
                        children: [
                          Expanded(
                            child: ListView.separated(
                              padding: EdgeInsets.only(top: 0),
                              shrinkWrap: true,
                              physics: BouncingScrollPhysics(),
                              controller: _scrollController,
                              itemBuilder: (context, index) {
                                return NunutTripCard(
                                  images: images,
                                  date: rideScheduleList[index].date!,
```

```
                                    time: rideScheduleList[index].time!,
                                    pickupLocation:
rideScheduleList[index]
                                        .meetingPoint!
                                        .name!,
                                    destination: rideScheduleList[index]
                                        .destination!
                                        .name!,
                                    isActive:
rideScheduleList[index].isActive!,
                                    cancelTap: () {
                                      Navigator.pushNamed(
                                          context, '/pengaduanKendala',
                                          arguments: {
                                            'title': 'Pembatalan
Permintaan',
                                            'isiTitle': 'Pembatalan
Permintaan',
                                            'roomChatId':
rideScheduleList[index].chatRoomId,
                                            'ride_schedule_user_id':

rideScheduleList[index].id!,
                                          });
                                    },
                                    onPressed: () {
                                      try {
                                        Navigator.push(
                                          context,
                                          MaterialPageRoute(
                                            builder: (context) =>
GroupChat(
                                              chatRoomId:
rideScheduleList[index]

                                                  .chatRoomId!,
                                              time:
rideScheduleList[index].time!,
                                              date:
rideScheduleList[index].date!,
                                              rideScheduleId:
rideScheduleList[index].id!,
                                              driver: false,
                                            ),
                                          ),
                                        );
```

```
                                    } catch (e) {
                                      print(e);
                                    }
                                  },
                                );
                              },
                              separatorBuilder: (context, index) {
                                return SizedBox(height: 12);
                              },
                              itemCount: rideScheduleList.length,
                            ),
                          ),
                          isLoading
                              ? Container(
                                  margin: EdgeInsets.only(top: 20,
bottom: 20),
                                  child: Center(
                                    child: CircularProgressIndicator(
                                      color: nunutPrimaryColor,
                                    ),
                                  ),
                                )
                              : Container(),
                        ],
                      ),
                    ),
                  ],
                ),
              ),
            ],
          ),
          floatingActionButton: FloatingActionButton(
            onPressed: () {
              Navigator.of(context).pushNamed('/addRideScheduleUser');
            },
            backgroundColor: nunutPrimaryColor,
            child: Icon(Icons.add, color: Colors.black),
          ),
        );
      }
```

4.4.7. List Tawaran Tumpangan

Fitur ini memungkinkan user untuk membuat permintaan tumpangan yang mereka inginkan dari Universitas Kristen Petra ke tujuan yang diinginkan maupun sebaliknya. *Driver* dapat melihat semua permintaan pengguna yang telah dibuat pada halaman list permintaan. Source code untuk fitur ini dapat dilihat pada Segmen Program 4.7.

Segmen Program 4. 7 Implementasi Halaman Request Tumpangan

```
Future<void> _enterChatRoom(String userId, String roomChatId) async {
  await chatApi.enterChatRoom(userId, roomChatId);
}

Future<List<String>> checkWhoDriver(String driverId) async {
  return await chatApi.checkWhoDriver(driverId);

Future<void> initRideScheduleList() async {
    setState(() {
      destinationController.text = "Petra Christian University";
      rideScheduleListLoading = true;
      _page = 1;
    });

    List<RideSchedule> schedules = [];
    if (isSearch) {
      schedules = await rideScheduleApi.getRideScheduleListPetraSearch(
        parameter:
"user&driver&vehicle&ride_request&status_ride=active",
        page: _page,
        checkUrl: true,
        date: dateController.text,
        meetingPoint: pickUpController.text,
      );
    } else {
      schedules = await rideScheduleApi.getRideScheduleListPetra(
        parameter:
"user&driver&vehicle&ride_request&status_ride=active",
        page: _page,
        checkUrl: true,
      );
    }

    setState(() {
      rideScheduleList = schedules;
```

```
      rideScheduleListLoading = false;
      _page++;
    });
  }

  Future<void> loadMore() async {
    if (!isLoading) {
      setState(() {
        isLoading = true;
      });

      List<RideSchedule> schedules = [];
      if (isSearch) {
        schedules = await
rideScheduleApi.getRideScheduleListPetraSearch(
          parameter:
"user&driver&vehicle&ride_request&status_ride=active",
          page: _page,
          date: dateController.text,
          meetingPoint: pickUpController.text,
        );
      } else {
        schedules = await rideScheduleApi.getRideScheduleListPetra(
          parameter:
"user&driver&vehicle&ride_request&status_ride=active",
          page: _page,
        );
      }

      setState(() {
        rideScheduleList.addAll(schedules);
        isLoading = false;
        _page++;
        done = schedules.isEmpty;
      });
    }
  }

initRideScheduleList() async {
    setState(() {
      pickUpController.text = "Petra Christian University";
      rideScheduleListLoading = true;
      _page = 1;
    });
```

```
    if (isSearch) {
      setState(() {
        pickUpController.text = "Petra Christian University";
        rideScheduleListLoading = true;
        _page = 1;
      });
      rideScheduleList.clear();
      rideScheduleList = await
rideScheduleApi.getRideScheduleListPlacesSearch(
        parameter:
"user&driver&vehicle&ride_request&status_ride=active",
        page: _page,
        checkUrl: true,
        date: dateController.text,
        destination: destinationController.text,
      );
    } else {
      rideScheduleList.clear();
      rideScheduleList = await
rideScheduleApi.getRideScheduleListPlaces(
          parameter:
"user&driver&vehicle&ride_request&status_ride=active",
          page: _page,
          checkUrl: true);
    }

    setState(() {
      rideScheduleListLoading = false;
      _page++;
    });
  }

Future<List<RideSchedule>> getRideScheduleListPetra(
      {String parameter = "", bool checkUrl = false, int page = 0})
async {
    String _parameter = "";
    if (page > 0) _parameter = "/list/$page";
    if (parameter.isNotEmpty) _parameter += "?$parameter";
    var url = Uri.parse(config.baseUrl + '/ride-schedule' + _parameter);

    if (checkUrl) print(url);

    var response = await http.get(
      url,
      headers: {
```

```
        'Content-Type': 'application/json',
        'Authorization': 'Bearer ${config.user.token}',
      },
    );

    Result result;
    List<RideSchedule> rideScheduleList = [];

    result = Result.fromJson(json.decode(response.body));

    if (result.status == 200) {
      for (var item in result.data) {
        if (item['driver_id']['user_id'] != config.user.id &&
            item['destination']['map_id'] == config.mapIdPetra) {
          rideScheduleList.add(RideSchedule.fromJson(item));
        }
      }
    }
    return rideScheduleList;
  }

Future<List<RideSchedule>> getRideScheduleListPlaces(
      {String parameter = "", bool checkUrl = false, int page = 0})
async {
    String _parameter = "";
    if (page > 0) _parameter = "/list/$page";
    if (parameter.isNotEmpty) _parameter += "?$parameter";
    var url = Uri.parse(config.baseUrl + '/ride-schedule' + _parameter);

    if (checkUrl) print(url);

    var response = await http.get(
      url,
      headers: {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer ${config.user.token}',
      },
    );

    Result result;
    List<RideSchedule> rideScheduleList = [];

    result = Result.fromJson(json.decode(response.body));

    if (result.status == 200) {
```

```
      for (var item in result.data) {
        if (item['driver_id']['user_id'] != config.user.id &&
            item['meeting_point']['map_id'] == config.mapIdPetra) {
          rideScheduleList.add(RideSchedule.fromJson(item));
        }
      }
    }
    return rideScheduleList;
  }
```

4.4.8. Forum Chat

Fitur ini memungkinkan pengguna untuk saling berkomunikasi melalui pesan teks ke sesama *driver* maupun pengguna lainnya untuk saling berkoordinasi mengenai *ridesharing* yang akan dilakukan. Source code untuk fitur ini dapat dilihat pada Segmen Program 4.8.

Segmen Program 4. 8 Implementasi Forum Chat

```
Future<void> removeRequestAndChangeStatus() async {
    await chatApi.removeRequestAndChangeStatus(
        widget.rideScheduleId, widget.chatRoomId);
  }

  Future<void> _loadLocationPoint() async {
    List<String> locationPoints =
        await chatApi.getLocationPoint(widget.rideScheduleId);
    setState(() {
      meetingPoint = locationPoints[0];
      destination = locationPoints[1];
    });
  }

  Future<void> _loadVehicle() async {
    String vehicleId = await
chatApi.getVehicleRequest(config.user.driverId!);
    setState(() {
      vehicle = vehicleId;
    });
  }

  Future<void> _loadCheckDriver() async {
    bool check = await chatApi.checkDriver(widget.chatRoomId,
config.user.id!);
    setState(() {
```

```
      checkDriver = check;
    });
  }

  Future<void> _loadCheckRequest() async {
    bool check = await chatApi.checkRequest(widget.chatRoomId,
config.user.id!);
    setState(() {
      checkRequest = check;
    });
  }

  Future<void> _loadCheckSent() async {
    bool check = await chatApi.checkSent(widget.chatRoomId,
config.user.id!);
    setState(() {
      _hasSentMessage = check;
    });
  }

  Future<void> _loadCheckNunut() async {
    bool check = await chatApi.checkNunut(widget.chatRoomId,
config.user.id!);
    setState(() {
      checkNunut = check;
    });
  }

  Future<void> _loadChatRoomTitle() async {
    try {
      String title = await chatApi.getRoomTitle(widget.chatRoomId);
      setState(() {
        roomTitle = title;
        isLoading = false;
      });
    } catch (e) {
      setState(() {
        roomTitle = 'Error loading title';
        isLoading = false;
      });
    }
  }

void _sendNegotiatedPrice(String priceText) {
    chatApi.sendNegotiate(widget.chatRoomId, widget.rideScheduleId,
```

```
priceText, config.user.id!);
  }

Future<Widget> getMessages(String roomId, String rideScheduleId) async {
    try {
      Stream<QuerySnapshot> messageStream = FirebaseFirestore.instance
          .collection('chats')
          .doc(roomId)
          .collection('messages')
          .orderBy('timestamp')
          .snapshots();

      return StreamBuilder<QuerySnapshot>(
        stream: messageStream,
        builder: (BuildContext context, AsyncSnapshot<QuerySnapshot>
snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Container();
          } else if (snapshot.hasError) {
            return Text("Error: ${snapshot.error}");
          } else {
            return FutureBuilder<DocumentSnapshot>(
              future: FirebaseFirestore.instance
                  .collection('chats')
                  .doc(roomId)
                  .get(),
              builder: (BuildContext context,
                  AsyncSnapshot<DocumentSnapshot> roomSnapshot) {
                if (roomSnapshot.connectionState ==
ConnectionState.waiting) {
                  return Container();
                } else if (roomSnapshot.hasError) {
                  return Text("Error: ${roomSnapshot.error}");
                } else {
                  List<QueryDocumentSnapshot> messages =
snapshot.data!.docs;

                  return ListView.builder(
                    itemCount: messages.length,
                    itemBuilder: (context, index) {
                      if (messages[index]['isNegotiate'] != null &&
                          messages[index]['isNegotiate'] == true) {
                        return NegotiatePrice(
                          rideSchedule_id: rideScheduleId,
                          room_id: roomId,
```

```
                          messageId: messages[index].id,
                          nama: roomSnapshot.data!['members']
                              [messages[index]['senderId']]['name'],
                          negotiatePrice:
                              messages[index]['negoPrice'] as String,
                          time: messages[index]['timestamp'] != null
                              ? messages[index]['timestamp']
                                  .toDate()
                                  .toString()
                                  .substring(11, 16)
                              : '',
                          sender: messages[index]['senderId'] ==
config.user.id,
                          isDriver: roomSnapshot.data!['members']
                              [config.user.id]['isDriver'],
                          imageUrl: roomSnapshot.data!['members']
                              [messages[index]['senderId']]['imageUrl'],
                        );
                      } else if
(messages[index]['announcementNegotiate'] !=
                              null &&
                          messages[index]['announcementNegotiate'] ==
true &&
                          messages[index]['negoAccepted'] == true) {
                        return ChatAnnouncement(
                            nama: "",
                            join: 0,
                            offer: 1,
                            price:
int.tryParse(messages[index]['negoPrice']) ??
                                0);
                      } else if
(messages[index]['announcementNegotiate'] !=
                              null &&
                          messages[index]['announcementNegotiate'] ==
true) {
                        return ChatAnnouncement(
                            nama: "",
                            join: 0,
                            offer: 2,
                            price:
int.tryParse(messages[index]['negoPrice']) ??
                                0);
                      } else if (messages[index]['announcementRequest']
!=
```

```
                              null &&
                          messages[index]['announcementRequest'] ==
true) {
                        return ChatAnnouncement(
                            nama: roomSnapshot.data!['members']
                                [messages[index]['senderId']]['name'],
                            join: 2,
                            offer: 0,
                            price:
int.tryParse(messages[index]['negoPrice']) ??
                                0);
                      } else if (messages[index]['announcementRequest']
!=
                              null &&
                          messages[index]['announcementJoin'] == true) {
                        return ChatAnnouncement(
                            nama: roomSnapshot.data!['members']
                                [messages[index]['senderId']]['name'],
                            join: 1,
                            offer: 0,
                            price:
int.tryParse(messages[index]['negoPrice']) ??
                                0);
                      } else {
                        return ChatBubble(
                          nama: roomSnapshot.data!['members']
                              [messages[index]['senderId']]['name'],
                          sender: messages[index]['senderId'] ==
config.user.id,
                          message: messages[index]['message'],
                          time: messages[index]['timestamp'] != null
                              ? messages[index]['timestamp']
                                  .toDate()
                                  .toString()
                                  .substring(11, 16)
                              : '',
                          imageUrl: roomSnapshot.data!['members']
                              [messages[index]['senderId']]['imageUrl'],
                          isDriver: roomSnapshot.data!['members']
                              [messages[index]['senderId']]['isDriver'],
                        );
                      }
                    },
                  );
                }
```

```
            },
          );
        }
      },
    );
  } catch (e) {
    print("Error getting messages: $e");
    return Text("Error getting messages: $e");
  }
}
```

4.4.9. List Chat Nunut

Fitur ini memungkinkan pengguna untuk melihat daftar *chat* yang terhubung dan aktif saat ini dengan pengguna. Source code untuk fitur ini dapat dilihat pada Segmen Program 4.9.

Segmen Program 4. 9 Implementasi List Chat Nunut

```
Future<void> loadAllData() async {
    await loadRideSchedule();
    await loadTitleChat();
    await loadLastChat();
    await loadLastChatName();
    await loadCheckRequest();
    setState(() {
      isLoading = false;
    });
  }

  Future<void> loadRideSchedule() async {
    rideScheduleList =
        await rideScheduleApi.getRideScheduleListChat(config.user.id!);
  }

  Future<void> loadCheckRequest() async {
    List<bool> listCheckRequest =
        await chatApi.getCheckRequest(config.user.id!);
    if (mounted) {
      setState(() {
        checkRequestList = listCheckRequest;
      });
    }
  }
```

```
  Future<void> loadLastChat() async {
    List<String> listChat = await chatApi.getLastChat(config.user.id!);
    if (mounted) {
      setState(() {
        lastChatList = listChat;
      });
    }
  }

  Future<void> loadLastChatName() async {
    List<String> listChatName = await
chatApi.getLastChatName(config.user.id!);
    if (mounted) {
      setState(() {
        lastChatNameList = listChatName;
      });
    }
  }

  Future<void> loadTitleChat() async {
    List<String> titleChat = await
chatApi.getTitleChat(config.user.id!);
    if (mounted) {
      setState(() {
        titleChatList = titleChat;
      });
    }
  }
```

4.4.10. Negosiasi Harga

Fitur ini memungkinkan pengguna untuk melakukan negosiasi harga kepada *driver* sesuai kesepakatan bersama berdasarkan harga yang sudah ditetapkan. Source code untuk fitur ini dapat dilihat pada Segmen Program 4.10.

Segmen Program 4. 10 Implementasi Negosiasi Harga

```
@override
  Widget build(BuildContext context) {
    final bg = widget.sender ? nunutPrimaryColor : Colors.white;
    final align =
        widget.sender ? CrossAxisAlignment.start :
CrossAxisAlignment.end;
    final messageColor = widget.sender ? Colors.black : Colors.black;
```

```
    final negotiatePriceFormat = widget.negotiatePrice.replaceAllMapped(
        RegExp(r'(\d{1,3})(?=(\d{3})+(?!\d))'), (Match m) =>
'${m[1]},');

    return Column(
      crossAxisAlignment: align,
      children: [
        Row(
          mainAxisAlignment:
              !widget.sender ? MainAxisAlignment.start :
MainAxisAlignment.end,
          children: [
            if (!widget.sender)
              CircleAvatar(
                backgroundImage: NetworkImage(widget.imageUrl),
              ),
            if (widget.sender)
              Container(
                margin: EdgeInsets.only(right: 5),
                child: Text(
                  widget.time,
                  textAlign: TextAlign.end,
                  style: TextStyle(
                    color: Color(0xFF656565),
                    fontSize: 12,
                    textBaseline: TextBaseline.alphabetic,
                  ),
                ),
              ),
            Container(
                margin: EdgeInsets.all(5.0),
                padding: EdgeInsets.all(10.0),
                decoration: BoxDecoration(
                  color: bg,
                  borderRadius: BorderRadius.circular(10.0),
                ),
                constraints: BoxConstraints(maxWidth: 250),
                child: Column(
                  children: [
                    Row(
                        mainAxisAlignment: !widget.sender
                            ? MainAxisAlignment.start
                            : MainAxisAlignment.end,
                        children: [
                          Text(
```

```
                            widget.nama,
                            style: TextStyle(
                                color: messageColor,
                                fontWeight: FontWeight.bold),
                          ),
                        ]),
                    SizedBox(height: 10),
                    Text(
                      'Tawaran harga baru!',
                      style: TextStyle(color: messageColor),
                    ),
                    Text(
                      'Rp ' + negotiatePriceFormat,
                      style: TextStyle(
                          color: messageColor,
                          fontWeight: FontWeight.bold,
                          fontSize: 18),
                    ),
                    SizedBox(height: 10),
                    if (widget.isDriver && !widget.sender)
                      Row(
                        mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
                        children: [
                          TextButton(
                            onPressed: () {

chatApi.acceptNegotiation(widget.rideSchedule_id,
widget.negotiatePrice);
chatApi.acceptNegotiationAnnoucement(widget.negotiatePrice,
widget.room_id, widget.messageId);
                              setState(() {
                                offerAccepted = 1;
                              });
                            },
                            style: TextButton.styleFrom(
                              side: BorderSide(color: Colors.black),
                            ),
                            child: Text(
                              'TERIMA',
                              style: TextStyle(color: Colors.black),
                            ),
                          ),
                          TextButton(
                            onPressed: () {
```

```
chatApi.declineNegotiationAnnoucement(widget.negotiatePrice,
widget.room_id, widget.messageId);
                              setState(() {
                                offerAccepted = 2;
                              });
                            },
                            style: TextButton.styleFrom(
                              side: BorderSide(color: Colors.black),
                            ),
                            child: Text(
                              'TOLAK',
                              style: TextStyle(color: Colors.black),
                            ),
                          ),
                        ],
                      ),
                    if (!widget.isDriver && widget.sender)
                      TextButton(
                        onPressed: () {
                          chatApi.cancelNegotiation(widget.room_id,
widget.messageId);
                        },
                        style: TextButton.styleFrom(
                          side: BorderSide(color: Colors.black),
                        ),
                        child: Text(
                          'BATAL NEGO',
                          style: TextStyle(color: Colors.black),
                        ),
                      ),
                  ],
                )),
            if (!widget.sender)
              Container(
                margin: EdgeInsets.only(left: 5),
                child: Text(
                  widget.time,
                  textAlign: TextAlign.end,
                  style: TextStyle(
                    color: Color(0xFF656565),
                    fontSize: 12,
                    textBaseline: TextBaseline.alphabetic,
                  ),
                ),
```

```
              ),
          ],
        ),
        if (offerAccepted == 1)
          ChatAnnouncement(
            nama: 'Driver',
            join: 0,
            offer: 1,
          ),
        if (offerAccepted == 2)
          ChatAnnouncement(
            nama: 'Driver',
            join: 0,
            offer: 2,
          ),
      ],
    );
  }
}
```

4.4.11. Google Maps API

Fitur ini memungkinkan pengguna mencari titik tujuan maupun *meeting point* dengan cukup akurat dan lengkap dikarenakan fitur ini sudah menggunakan API dari *Google Maps*. Source code untuk fitur ini dapat dilihat pada Segmen Program 4.11.

Segmen Program 4. 11 Implementasi Google Maps API

```
Future<void> _searchPlaces(String query) async {
    final String apiKey = 'API KEY';
    final String country = 'ID'; // Country code for Indonesia
    final String url =

'https://maps.googleapis.com/maps/api/place/autocomplete/json?input=$que
ry&key=$apiKey&components=country:$country';
    final response = await http.get(Uri.parse(url));

    if (response.statusCode == 200) {
      setState(() {
        _searchResults = json.decode(response.body)['predictions'];
      });
    } else {
      throw Exception('Failed to load search results');
    }
```

```
  }

  Future<void> _getPlaceDetails(String placeId) async {
    final String apiKey = 'API KEY';
    final String url =

'https://maps.googleapis.com/maps/api/place/details/json?placeid=$placeI
d&key=$apiKey';
    final response = await http.get(Uri.parse(url));

    if (response.statusCode == 200) {
      final result = json.decode(response.body)['result'];
      final lat = result['geometry']['location']['lat'];
      final lng = result['geometry']['location']['lng'];
      final name = result['name']; // Get the name field

      MapLocation mapLocation = MapLocation(
        mapId: placeId,
        name: name,
        latitude: lat.toString(),
        longitude: lng.toString(),
      );

      Navigator.pop(context, mapLocation); // Return the map location

    } else {
      throw Exception('Failed to load place details');
    }

Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          Positioned(
            top: 0,
            left: 0,
            right: 0,
            child: Container(
              height: 200,
              decoration: BoxDecoration(
                borderRadius: BorderRadius.only(
                  bottomRight: Radius.circular(50),
                ),
                color: nunutPrimaryColor,
              ),
```

```
            ),
          ),
          Container(
            margin: EdgeInsets.only(top: 60, left: 28, right: 28),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                IconButton(
                  icon: Icon(Icons.arrow_back, color: Colors.black),
                  onPressed: () {
                    Navigator.pop(context);
                  },
                ),
                SizedBox(height: 35),
                NunutText(title: "Pilih Lokasi", isTitle: true, size:
32),
                SizedBox(height: 50),
                TextFormField(
                  controller: _searchController,
                  decoration: InputDecoration(
                    hintText: 'Cari Lokasi...',
                    hintStyle: TextStyle(
                      color: Colors.grey,
                    ),
                    suffixIcon: Container(
                      margin: EdgeInsets.only(right: 10, top: 10,
bottom: 10),
                      decoration: BoxDecoration(
                        color: Colors.black,
                        borderRadius: BorderRadius.circular(10),
                      ),
                      child: IconButton(
                        icon: Icon(Icons.search),
                        color: Colors.white,
                        onPressed: () =>
_searchPlaces(_searchController.text),
                      ),
                    ),
                    prefixIcon: Icon(
                      Icons.location_searching,
                      color: Colors.grey,
                    ),
                    border: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(10),
                      borderSide: BorderSide(
```

```
                      color: Colors.grey[400]!,
                    ),
                  ),
                  enabledBorder: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(10),
                    borderSide: BorderSide(
                      color: Colors.grey[400]!,
                    ),
                  ),
                  focusedBorder: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(10),
                    borderSide: BorderSide(
                      color: Colors.grey[400]!,
                    ),
                  ),
                ),
                autocorrect: false, // Disable autocorrect
                textCapitalization:
                    TextCapitalization.none, // Disable
auto-capitalization
              ),
              Expanded(
                child: ListView.builder(
                  itemCount: _searchResults.length,
                  padding: EdgeInsets.only(top: 30),
                  itemBuilder: (context, index) {
                    final place = _searchResults[index];
                    return GestureDetector(
                      onTap: () =>
_getPlaceDetails(place['place_id']),
                      child: Container(
                        margin: EdgeInsets.only(bottom: 10),
                        child: Column(
                          children: [
                            Row(
                              children: [
                                Icon(Icons.circle,
                                    color: nunutPrimaryColor, size:
18),
                                SizedBox(width: 15),
                                Expanded(
                                  child: NunutText(
                                    title: place['description'],
                                    fontWeight: FontWeight.w500,
                                    size: 16,
```

```
                                    maxLines: 2,
                                  ),
                                ),
                              ],
                            ),
                            SizedBox(height: 12),
                            Divider(color: Colors.grey, thickness:
0.5),
                          ],
                        ),
                      ),
                    );
                  },
                ),
              ),
            ],
          ),
        ),
      ],
    ),
  );
 }
```