BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Sistem Manajemen Keuangan Mahasiswa (Anak Kos) merupakan sistem yang dirancang untuk membantu mahasiswa mencatat pemasukan dan pengeluaran serta mengelola keuangan secara mandiri menggunakan metode alokasi 50/30/20. Sistem ini menampilkan laporan transaksi, grafik anggaran, rekap keuangan bulanan dan tahunan, serta fitur pengelolaan kategori, saldo pada masing-masing pos kebutuhan, keinginan, dan tabungan. Dikembangkan dengan Flutter dan terintegrasi Supabase untuk backend dan autentikasi, sistem ini telah diuji melalui User Acceptance Testing (UAT) oleh 28 mahasiswa. Hasilnya menunjukkan aplikasi ini memenuhi ekspektasi pengguna dengan tingkat kepuasan tinggi, terutama pada fitur laporan dan pos anggaran, serta dinilai mudah digunakan dan menarik. Sebanyak 82% responden menganggap aplikasi sudah siap digunakan, dengan saran perbaikan minor dari sisanya. Secara keseluruhan, aplikasi dinilai efektif dalam membantu pengelolaan keuangan dan mendorong literasi finansial di kalangan mahasiswa.

4.2 IMPLEMENTASI DESAIN ANTARMUKA

Implementasi desain antarmuka dilakukan dengan menerapkan rancangan visual sistem ke dalam bentuk aplikasi yang dapat digunakan oleh pengguna. Pada tahap ini, tampilan antarmuka dikembangkan berdasarkan prototipe yang telah disusun dengan mempertimbangkan aspek kenyamanan, kemudahan penggunaan, dan konsistensi elemen visual. Untuk implementasi, *framework* Flutter digunakan bersama dengan layanan *backend* Supabase untuk manajemen data. Setiap komponen diintegrasikan secara interaktif untuk membuat antarmuka pengguna yang responsif dan sesuai kebutuhan pengguna.

4.2.1 Implementasi Halaman *Login*

Gambar 4.1 merupakan halaman awal aplikasi yang digunakan sebagai akses masuk ke aplikasi, di mana pengguna harus memasukkan *email* dan *password*. Kemudian proses autentikasi data dilakukan menggunakan Supabase.

Gambar 4.1 Implementasi halaman *login*

Source code dibawah berfungsi untuk mengelola proses *login user* dengan memverifikasi kredensial (*email* dan *password*) melalui API autentikasi Supabase. Jika *login* berhasil, *user* diarahkan ke halaman *dashboard*. Jika gagal, baik karena kredensial atau alasan lainnya, kode akan memberikan pesan *error* yang sesuai.

```
@override
Future<UserModel> login(String email, String password) async {
  try {
    final response = await client.auth.signInWithPassword(
      email: email,
      password: password,
    );

    final user = response.user;
    if (user != null) {
      return UserModel(
        id: user.id,
        email: user.email!,
        emailConfirmedAt: user.emailConfirmedAt != null
            ? DateTime.parse(user.emailConfirmedAt!)
            : null,
      );
```

```
    } else {
      throw 'Gagal masuk';
    }
  } on AuthApiException catch (e) {
    throw translateError(e.message);
  }
}
```

Pada bagian *try* menunjukkan proses autentikasi *login* menggunakan *email* dan *password user*. Bagian kode *final user* merupakan proses pengambilan data pengguna dari respon yang diterima setelah proses *login.* Bagian *if (user != null)* artinya jika pengguna tidak *null* atau pengguna ditemukan, maka proses *login* berhasil dan sistem mengembalikan objek *UserModel* berisi detail akun. Pada bagian *else* berarti jika pengguna tidak ditemukan atau *null*, maka sistem akan menampilkan pesan kegagalan *login.* Dan pada bagian *on AuthApiException catch* merupakan proses mendeteksi kesalahan autentikasi dari Supabase dan mengubahnya menjadi pesan yang lebih informatif melalui fungsi *translateError*.

4.2.2 Implementasi Halaman *Register*

Gambar 4.2 merupakan halaman *register*, untuk daftar akun baru ke dalam aplikasi. Pengguna diminta mengisi data berupa nama, *email*, dan *password*. Sistem akan melakukan autentikasi data, setelah itu akun dapat digunakan untuk *login*.

Gambar 4.2 Implementasi halaman *register*

Source code dibawah berfungsi untuk mengelola proses *register user* dengan mengirimkan data nama, *email*, dan *password*, proses ini juga menghasilkan warna avatar acak. Setelah berhasil mendaftar, dua kategori pengeluaran *default* disimpan di *database* untuk pengguna tersebut. Jika proses pendaftaran sukses, Supabase akan mengirim *email* verifikasi agar akun dapat digunakan untuk *login*. Jika gagal, kode akan memberikan pesan *error* yang sesuai.

```
@override
Future<UserModel> register(String name, String email, String
password) async {
  try {
    final randomColor = _generateRandomColorHex();

    final response = await client.auth.signUp(
      email: email,
      password: password,
      data: {
        'display_name': name,
        'avatar_color': randomColor,
      },
    );

    final user = response.user;

    if (user != null) {
      await client.from('categories').insert([
        {
          'id': const Uuid().v4(),
          'user_id': user.id,
          'name': 'Bayar Kos',
          'type': 'expense',
        },
        {
          'id': const Uuid().v4(),
          'user_id': user.id,
          'name': 'Biaya Kuliah',
          'type': 'expense',
        },
      ]);
      return UserModel(
        id: user.id,
        email: user.email!,
        emailConfirmedAt: user.emailConfirmedAt != null
            ? DateTime.parse(user.emailConfirmedAt!)
            : null,
      );
    } else {
      throw 'Gagal daftar';
    }
  } on AuthApiException catch (e) {
    throw translateError(e.message);
```

```
  }
}

String _generateRandomColorHex() {
  final random = Random();
  final r = random.nextInt(256);
  final g = random.nextInt(256);
  final b = random.nextInt(256);
  final a = 255;

  return '#${(a << 24 | r << 16 | g << 8 |
b).toRadixString(16).padLeft(8, '0')}';
}
```

Pada bagian *try* menunjukkan rangkaian proses pendaftaran akun pengguna baru. Langkah pertama dimulai dengan pemanggilan fungsi *_generateRandomColorHex()* untuk membuat warna avatar secara acak dalam format HEX, yang nantinya disertakan sebagai bagian dari data tambahan pengguna. Metode *client.auth.signUp* mengirimkan informasi berupa email, password, serta metadata seperti display_name dan avatar_color ke Supabase guna membuat akun baru. Hasil dari proses ini disimpan dalam variabel user sebagai respons dari Supabase. Bagian kode *if (user != null)* artinya jika pengguna tidak *null*, maka pendaftaran berhasil. Pada bagian kode *client.from('categories').insert* sistem menambahkan dua kategori pengeluaran *default* ke dalam tabel *categories* milik pengguna tersebut, yaitu "Bayar Kos" dan "Biaya Kuliah", masing-masing dengan ID unik dan dikaitkan dengan *user_id.* Setelah itu, fungsi mengembalikan objek *UserModel* berisi informasi penting pengguna seperti ID, *email*, dan waktu konfirmasi jika ada. Jika *user* bernilai *null*, maka sistem akan menampilkan pesan kegagalan daftar. Dan pada bagian *on AuthApiException catch* merupakan proses mendeteksi kesalahan autentikasi dari Supabase dan mengubahnya menjadi pesan yang lebih informatif melalui fungsi *translateError*.

4.2.3 Implementasi Halaman *Dashboard*

Gambar 4.2 merupakan halaman *dashboard*, yaitu halaman yang menampilkan informasi transaksi harian, termasuk rincian serta total nominal transaksi yang dilakukan pada hari tersebut. Selain itu, pada halaman ini juga menampilkan persentase pengeluaran berdasarkan sumber anggaran.

Gambar 4.3 Implementasi halaman *dashboard*

Source code getTransactions berfungsi untuk mengelola proses pengambilan data transaksi harian berdasarkan tanggal, dan *user_id* yang sedang *login* ke sistem.

```
@override
Future<List<TransactionModel>> getTransactions(
    String userId, DateTime date) async {
  final response = await client
      .from('transactions')
      .select()
      .eq('user_id', userId)
      .eq('date', date.toIso8601String().split('T')[0])
      .order('created_at', ascending: false);

  return (response as List)
      .map((json) => TransactionModel.fromJson(json))
      .toList();
}
```

Source code getBudgetSummaryByDate berfungsi untuk mengelola proses pengambilan total data transaksi harian dari tabel *transactions*, yang menghitung total pemasukan dan pengeluaran berdasarkan tanggal, dan *user_id* yang sedang *login* ke sistem.

```
@override
Future<Map<String, double>> getBudgetSummaryByDate(
    String userId, DateTime date) async {
```

```
  final formattedDate = date.toIso8601String().split('T')[0];

  final response = await client
      .from('transactions')
      .select()
      .eq('user_id', userId)
      .eq('date', formattedDate);

  double totalIncome = 0;
  double totalExpense = 0;

  for (final item in response as List) {
    final amount = double.tryParse(item['amount'].toString()) ??
0;
    if (item['type'] == 'income') {
      totalIncome += amount;
    } else if (item['type'] == 'expense') {
      totalExpense += amount;
    }
  }

  return {
    'total_income': totalIncome,
    'total_expense': totalExpense,
  };
}
```

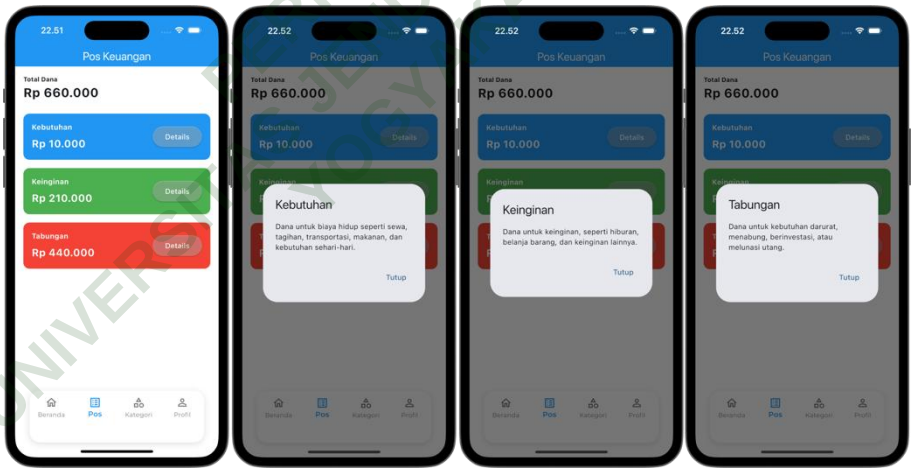
Source code getExpenseDistributionBySource merupakan proses pengambilan data pengeluaran berdasarkan *budget source*, dengan menghitung distribusi pengeluaran pada sumber (*needs*, *wants*, *savings*) untuk bulan tertentu.

```
@override
Future<Map<String, double>> getExpenseDistributionBySource(
    String userId, DateTime date) async {
  final month = date.month;
  final year = date.year;

  final startDate = DateTime(year, month, 1);
  final endDate = DateTime(year, month + 1, 0);

  final response = await client
      .from('transactions')
      .select('amount, source')
      .eq('user_id', userId)
      .eq('type', 'expense')
      .gte('date', startDate.toIso8601String().split('T').first)
      .lte('date', endDate.toIso8601String().split('T').first);

  double total = 0;
  final Map<String, double> sourceTotals = {
    'needs': 0,
    'wants': 0,
    'savings': 0,
  };
```

```
  for (final item in response as List) {
    final amount = double.tryParse(item['amount'].toString()) ??
0;
    final source = item['source'];
    if (sourceTotals.containsKey(source)) {
      sourceTotals[source] = sourceTotals[source]! + amount;
      total += amount;
    }
  }

  if (total > 0) {
    return sourceTotals.map((key, value) => MapEntry(
        key, double.parse(((value / total) *
100).toStringAsFixed(2))));
  } else {
    return sourceTotals.map((key, value) => MapEntry(key, 0.0));
  }
}
```

4.2.4 Implementasi Halaman Pos Anggaran

Gambar 4.4 merupakan halaman pos anggaran, halaman yang berisi informasi total dana yang dimiliki pengguna, termasuk informasi total dana di setiap pos anggaran pengguna.



Gambar 4.4 Implementasi halaman pos anggaran

Source code fetchPosData berfungsi untuk mengelola proses pengambilan data pos anggaran berdasarkan *userId* pengguna. Fungsi ini dimulai dalam blok *try*, pengambilan data dari Supabase oleh sistem. Bagian *incomeResponse* menjumlahkan seluruh nilai *amount* dari transaksi dengan tipe *income* untuk menghasilkan total pemasukan. Selanjutnya, *budgetResponse* mengambil data dari

tabel *budget*_allocation, khususnya kolom *needs*, *wants*, dan *savings*, yang difilter berdasarkan *userId*. Nilai *needsTotal*, *wantsTotal*, dan *savingsTotal* diinisialisasi dengan nol dan diisi melalui perulangan sambil memastikan nilai *null* dihitung sebagai nol. Setelah perhitungan selesai, data dikembalikan dalam bentuk objek PosModel. Jika terjadi kesalahan, maka akan ditangani oleh blok catch.

```
@override
Future<PosModel?> fetchPosData(String userId) async {
  try {
    final incomeResponse = await client
        .from('transactions')
        .select('amount')
        .eq('user_id', userId)
        .eq('type', 'income');

    final incomeTotal = incomeResponse.fold<double>(
        0, (sum, row) => sum + (row['amount'] as
num).toDouble());

    final budgetResponse = await client
        .from('budget_allocation')
        .select('needs, wants, savings')
        .eq('user_id', userId);

    double needsTotal = 0;
    double wantsTotal = 0;
    double savingsTotal = 0;

    for (final row in budgetResponse) {
      needsTotal += (row['needs'] ?? 0).toDouble();
      wantsTotal += (row['wants'] ?? 0).toDouble();
      savingsTotal += (row['savings'] ?? 0).toDouble();
    }

    return PosModel(
      totalIncome: incomeTotal,
      needs: needsTotal,
      wants: wantsTotal,
      savings: savingsTotal,
    );
  } catch (e) {
    throw Exception("Gagal mengambil data POS:
${e.toString()}");
  }
}
```

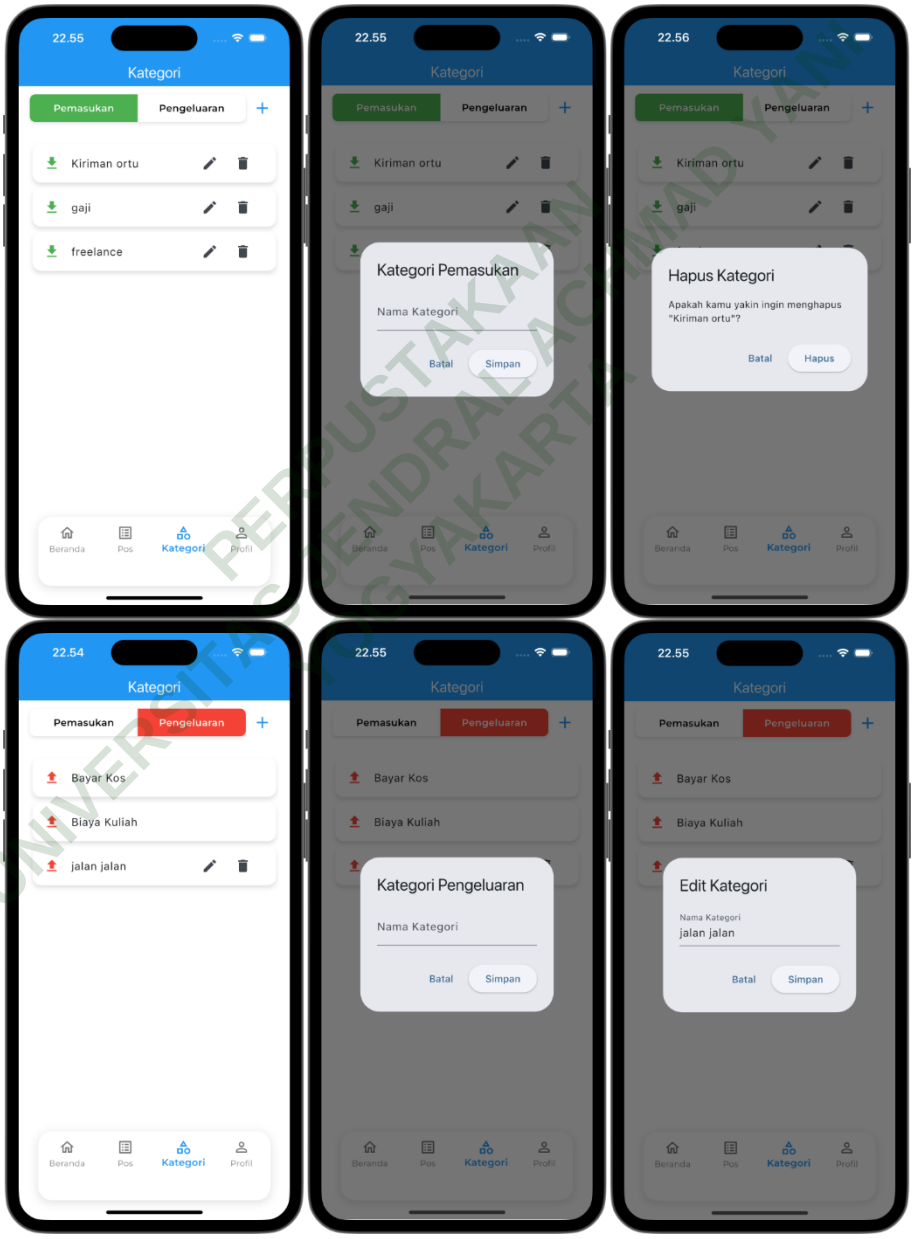
Source code *formatCurrency* merupakan proses penjumlahan dana kebutuhan, keinginan, dan tabungan untuk di tampilkan sebagai total dana.

```
Text(
  formatCurrency(data.needs + data.wants + data.savings),
```

```
  style: const TextStyle(
    fontSize: 24,
    fontWeight: FontWeight.bold,
  ),
),
```

4.2.5 Implementasi Halaman Kategori

Gambar 4.5 menampilkan halaman kategori dengan *pop-up* untuk menambah data, mengubah data, dan menghapus data.



Gambar 4.5 Implementasi halaman kategori

Source code getCategories berfungsi untuk mengelola proses pengambilan data kategori berdasarkan *userId* dan *type*, kemudian data tersebut dikembalikan dalam bentuk list.

```
@override
Future<List<CategoryModel>> getCategories(String userId, String
type) async {
  final response = await client
      .from('categories')
      .select()
      .eq('user_id', userId)
      .eq('type', type);

  return (response as List)
      .map((e) => CategoryModel.fromJson(e))
      .toList();
}
```

Source code dibawah berfungsi untuk mengelola proses tambah, ubah, dan hapus data kategori. Setiap aksi yang dilakukan berhasil, maka data kategori akan dimuat ulang. Bagian *AddCategoryEvent* digunakan untuk menambahkan kategori baru lalu memuat ulang daftar kategori melalui *LoadCategories*, dan jika gagal, akan memunculkan state *CategoryError*. Pada *DeleteCategoryEvent*, sistem menghapus kategori berdasarkan id kategori dan juga memuat ulang daftar kategori melalui *LoadCategories*, dan jika gagal, akan memunculkan state *CategoryError*. Pada *UpdateCategoryEvent*, fungsi memperbarui kategori berdasarkan id kategori dan juga memuat ulang daftar kategori melalui *LoadCategories*, dan jika gagal, akan memunculkan state *CategoryError*.

```
on<AddCategoryEvent>((event, emit) async {
    try {
      await addCategory(event.category);
      add(LoadCategories(event.category.userId,
event.category.type));
    } catch (e) {
      emit(CategoryError(e.toString()));
    }
  }
);

on<DeleteCategoryEvent>((event, emit) async {
    try {
      await deleteCategory(event.id);
      add(LoadCategories(event.userId, event.type));
    } catch (e) {
      emit(CategoryError(e.toString()));
```

```
    }
  }
);

on<UpdateCategoryEvent>((event, emit) async {
    try {
      await updateCategory(event.category);
      add(LoadCategories(event.category.userId,
event.category.type));
    } catch (e) {
      emit(CategoryError(e.toString()));
    }
  }
);
```

4.2.6 Implementasi Halaman Transaksi

Gambar 4.6 menampilkan halaman tambah transaksi, berisi *form* untuk isi data transaksi pemasukan atau pengeluaran.

Gambar 4.6 Implementasi halaman transaksi

Source code addTransaction digunakan untuk menyimpan data transaksi ke dalam basis data dengan mengambil informasi dari objek *TransactionEntity*. Data seperti *userId*, *amount*, *date*, *category*, *note*, *type*, *source*, dan *categoryId* diambil untuk mempermudah proses. Proses dimulai dalam blok *try*, diawali dengan mengambil bulan dan tahun dari tanggal transaksi untuk mencari data anggaran (*budget_allocation*) dengan metode *maybeSingle()*. Jika ditemukan, ID-nya

disimpan sebagai *allocationId*. Transaksi kemudian disimpan ke tabel *transactions*, beserta detail yang relevan seperti *source*, *allocation_id*, dan *category_id* jika tersedia. Jika transaksi bertipe *'income'*, sistem akan menghitung alokasi dana berdasarkan metode 50/30/20, dan akan menambahkan entri baru atau memperbarui data yang sudah ada di *budget_allocation*. Jika bertipe *'expense'*, sistem memastikan data anggaran sudah ada dan source terisi; jika tidak, akan dilempar error. Selanjutnya, sistem memverifikasi apakah dana di *source* cukup, dan jika cukup, nilai dana dikurangi serta *total_expense* ditambah. Jika terjadi kesalahan di salah satu langkah, maka akan ditangani oleh blok *catch* dan melempar *exception* dengan pesan "Gagal menyimpan transaksi" disertai detail error-nya..

```
@override
Future<void> addTransaction(TransactionEntity transaction) async
{
  final userId = transaction.userId;
  final amount = transaction.amount;
  final date = transaction.date;
  final category = transaction.category;
  final note = transaction.note;
  final type = transaction.type;
  final source = transaction.source;
  final categoryId = transaction.categoryId;

  try {
    final month = date.month;
    final year = date.year;

    final budgetData = await client
        .from('budget_allocation')
        .select()
        .eq('user_id', userId)
        .eq('month', month)
        .eq('year', year)
        .maybeSingle();

    final allocationId = budgetData != null ? budgetData['id'] :
null;

    await client.from('transactions').insert({
      'user_id': userId,
      'amount': amount,
      'type': type,
      'category': category,
      'note': note,
      'date': date.toIso8601String(),
      if (source != null) 'source': source,
      if (allocationId != null) 'allocation_id': allocationId,
      if (categoryId != null) 'category_id': categoryId,
```

```
    }).select();

    if (type == 'income') {
      final needs = amount * 0.5;
      final wants = amount * 0.3;
      final savings = amount * 0.2;

      if (budgetData == null) {
        await client.from('budget_allocation').insert({
          'user_id': userId,
          'month': month,
          'year': year,
          'needs': needs,
          'wants': wants,
          'savings': savings,
          'total_income': amount,
          'total_expense': 0,
        });
      } else {
        final updatedNeeds = (budgetData['needs'] ?? 0) + needs;
        final updatedWants = (budgetData['wants'] ?? 0) + wants;
        final updatedSavings = (budgetData['savings'] ?? 0) +
savings;
        final updatedIncome = (budgetData['total_income'] ?? 0)
+ amount;

        await client
            .from('budget_allocation')
            .update({
              'needs': updatedNeeds,
              'wants': updatedWants,
              'savings': updatedSavings,
              'total_income': updatedIncome,
            })
            .eq('user_id', userId)
            .eq('month', month)
            .eq('year', year);
      }
    } else if (type == 'expense') {
      if (budgetData == null) {
        throw Exception("Data alokasi tidak ditemukan untuk
bulan ini");
      }

      if (source == null) {
        throw Exception("Source harus diisi untuk pengeluaran");
      }

      final currentSourceValue = (budgetData[source] ?? 0) as
num;

      if (currentSourceValue < amount) {
        throw Exception("Dana ${source} tidak cukup");
      }

      final updatedSourceValue = currentSourceValue - amount;
```

```
      final updatedTotalExpense =
          (budgetData['total_expense'] ?? 0) + amount;

      await client
          .from('budget_allocation')
          .update(<String, dynamic>{
            source: updatedSourceValue,
            'total_expense': updatedTotalExpense,
          })
          .eq('user_id', userId)
          .eq('month', month)
          .eq('year', year);
    }
  } catch (e) {
    throw Exception('Gagal menyimpan transaksi:
${e.toString()}');
  }
}
```

4.2.7 Implementasi Halaman Profil

Gambar 4.7 menampilkan halaman profil dengan informasi data pengguna, terdapat juga navigasi untuk ke halaman laporan keuangan, serta tombol untuk keluar aplikasi.

Gambar 4.7 Implementasi halaman profil

Source code getInitials digunakan untuk mengambil inisial dari nama pengguna, dengan mengembalikan satu huruf jika hanya satu kata, atau dua huruf pertama dari dua kata awal jika lebih dari satu, semuanya dalam huruf kapital. Pada

bagian *initState*, fungsi *fetchUserData* mengambil data pengguna yang sedang login melalui Supabase. Di dalam *fetchUserData*, jika pengguna tidak *null*, maka metadata pengguna akan diakses. Fungsi *setState* kemudian digunakan untuk mengatur nilai variabel lokal *username* berdasarkan *display_name*, *email* serta *avatarColor* jika terdapat data warna avatar dalam format HEX yang kemudian dikonversi ke format *Color* menggunakan fungsi *_hexToColor*. Proses ini memastikan bahwa nama, email, dan warna avatar ditampilkan dengan benar di UI saat data pengguna tersedia.

```
String getInitials(String name) {
  if (name.trim().isEmpty) return "";

  List<String> words = name.trim().split(RegExp(r"\s+"));

  if (words.length == 1) {
    return words[0][0].toUpperCase();
  } else {
    return (words[0][0] + words[1][0]).toUpperCase();
  }
}

@override
void initState() {
  super.initState();
  fetchUserData();
}

void fetchUserData() async {
  final user = Supabase.instance.client.auth.currentUser;

  if (user != null) {
    final userMeta = user.userMetadata;

    setState(() {
      username = userMeta?['display_name'] ?? 'Unknown';
      email = user.email ?? 'No Email';
      final colorHex = userMeta?['avatar_color'];
      if (colorHex != null) {
        avatarColor = _hexToColor(colorHex);
      }
    });
  }
}
```

Source code logout berfungsi untuk mengelola proses keluar dari aplikasi, ketika berhasil keluar maka kode akan menampilkan pesan berhasil keluar.

```
Future<void> logout() async {
  try {
    await client.auth.signOut();
  } on AuthApiException catch (e) {
    throw translateError(e.message);
  }
}
```

4.2.8 Implementasi Halaman Laporan Keuangan

Gambar 4.8 menampilkan halaman laporan keuangan, terdapat informasi laporan keuangan bulanan dan tahunan, serta *form download* dengan *filter* tanggal.

Gambar 4.8 Implementasi halaman laporan keuangan

Source code dibawah berfungsi untuk mengelola proses pengambilan data dari tabel *transactions* yang akan diambil total transaksinya berdasarkan *user_id* yang sedang *login*. Pada bagian *getMonthlyReport*, parameter *month* dan *year* dikonversi menjadi bilangan bulat (*intMonth* dan *intYear*), lalu digunakan untuk membuat startDate di tanggal 1 dan endDate di awal bulan berikutnya. Fungsi kemudian mengambil *userId* dari pengguna yang sedang login. Jika userId tidak tersedia , maka fungsi langsung mengembalikan list kosong. Jika tersedia, sistem melakukan *query* ke tabel *transactions* untuk mengambil transaksi dengan *user_id* yang cocok, dan tanggal yang berada di antara *startDate* dan *endDate*, disusun berdasarkan tanggal secara menaik, dan dibatasi maksimal 100 data. Hasilnya

kemudian diubah menjadi *list* objek *ReportModel*. Pada bagian *getYearlyReport*, menerima parameter *year* lalu mengonversinya ke bilangan bulat (*intYear*). *startDate* diatur ke 1 Januari tahun tersebut dan *endDate* ke 1 Januari tahun berikutnya. Langkah-langkah selanjutnya identik: mengambil *userId*, melakukan validasi null, lalu melakukan *query* transaksi dengan *filter* tanggal dalam satu tahun penuh. Data dikembalikan dalam bentuk *list ReportModel*. Dan pada bagian *getReportsThisYearUntilNow* tidak menerima parameter karena otomatis mengambil tanggal saat ini. Fungsi ini membuat *startDate* di awal tahun berjalan, dan *endDate* di awal bulan berikutnya dari bulan saat ini, sehingga mencakup semua transaksi dari Januari hingga bulan sekarang. Setelah memverifikasi *userId*, fungsi menjalankan *query* yang hampir sama dengan sebelumnya, hanya saja batas data dinaikkan menjadi 1000. Semua data yang didapat dikonversi ke objek *ReportModel* dan dikembalikan dalam bentuk *list*.

```
Future<List<ReportModel>> getMonthlyReport(String month, String
year) async {
  final intMonth = int.parse(month);
  final intYear = int.parse(year);
  final startDate = DateTime(intYear, intMonth, 1);
  final endDate = DateTime(intYear, intMonth + 1, 1);

  final userId = Supabase.instance.client.auth.currentUser?.id;
  if (userId == null) return [];

  final response = await supabaseClient
      .from('transactions')
      .select('amount, type, category, date')
      .eq('user_id', userId)
      .gte('date', startDate.toIso8601String())
      .lt('date', endDate.toIso8601String())
      .order('date', ascending: true)
      .limit(100);

  final data = response as List<dynamic>?;

  if (data == null || data.isEmpty) {
    return [];
  }

  return data
      .map((item) => ReportModel.fromJson(item as Map<String,
dynamic>))
      .toList();
}

Future<List<ReportModel>> getYearlyReport(String year) async {
```

```
  final intYear = int.parse(year);
  final startDate = DateTime(intYear, 1, 1);
  final endDate = DateTime(intYear + 1, 1, 1);

  final userId = Supabase.instance.client.auth.currentUser?.id;
  if (userId == null) return [];

 final response = await supabaseClient
      .from('transactions')
      .select('amount, type, category, date')
      .eq('user_id', userId)
      .gte('date', startDate.toIso8601String())
      .lt('date', endDate.toIso8601String())
      .order('date', ascending: true)
      .limit(100);

  final data = response as List<dynamic>?;

  if (data == null || data.isEmpty) {
    return [];
  }

  return data
      .map((item) => ReportModel.fromJson(item as Map<String,
dynamic>))
      .toList();
}

Future<List<ReportModel>> getReportsThisYearUntilNow() async {
  final now = DateTime.now();
  final startDate = DateTime(now.year, 1, 1);
  final endDate = DateTime(now.year, now.month + 1, 1);

  final userId = Supabase.instance.client.auth.currentUser?.id;
  if (userId == null) return [];

  final response = await supabaseClient
      .from('transactions')
      .select('amount, type, category, date')
      .eq('user_id', userId)
      .gte('date', startDate.toIso8601String())
      .lt('date', endDate.toIso8601String())
      .order('date', ascending: true)
      .limit(1000);

  final data = response as List<dynamic>?;

  if (data == null || data.isEmpty) {
    return [];
  }

  return data
      .map((item) => ReportModel.fromJson(item as Map<String,
dynamic>))
      .toList();
}
```

Source code _loadTransactionTypes berfungsi untuk mengambil daftar transaksi berdasarkan tipe transaksi dari Supabase melalui *ReportRemoteDatasource*, data transaksi ini merupakan data laporan keuangan yang akan digunakan dalam proses *download*. Setelah daftar didapat, fungsi menyaring agar tidak mencantumkan item bertuliskan 'semua tipe' secara ganda dengan menggunakan metode *.where()* dan *.toSet()* untuk menghilangkan duplikat. Kemudian, melalui *setState()*, daftar *transactionTypes* diperbarui dengan menyisipkan 'Semua tipe' sebagai opsi pertama. Fungsi ini juga memastikan bahwa *selectedType* tetap valid; jika nilai yang dipilih sebelumnya tidak tersedia dalam daftar baru, maka akan di-set ulang ke 'Semua tipe'.

```
Future<void> _loadTransactionTypes() async {
  final types = await
ReportRemoteDatasource(Supabase.instance.client)
      .getTransactionTypes();

  final filteredTypes = types
      .where((type) => type.toLowerCase() != 'semua tipe')
      .toSet()
      .toList();

  setState(() {
    transactionTypes = ['Semua tipe', ...filteredTypes];

    if (!transactionTypes.contains(selectedType)) {
      selectedType = 'Semua tipe';
    }
  });
}
```

Source code _selectDate merupakan mekanisme yang digunakan untuk menampilkan dialog *date picker* kepada pengguna. Fungsi ini menerima parameter berupa konteks dan *controller* yang akan digunakan untuk menyimpan nilai tanggal yang dipilih. Tanggal yang dipilih ini merupakan bagian dari rentang filter antara tanggal mulai dan tanggal akhir untuk proses ekspor data laporan. Setelah pengguna memilih tanggal, nilai tersebut diformat menjadi *string* dengan format yyyy-MM-dd menggunakan *DateFormat* dari paket intl, kemudian hasil format tersebut dimasukkan ke dalam *TextEditingController* yang sesuai.

```
Future<void> _selectDate(
    BuildContext context, TextEditingController controller)
async {
```

```
  DateTime? selectedDate = await showDatePicker(
    context: context,
    initialDate: DateTime.now(),
    firstDate: DateTime(2025),
    lastDate: DateTime.now(),
  );

  if (selectedDate != null) {
    controller.text = DateFormat('yyyy-MM-
dd').format(selectedDate);
  }
}
```

Source code _exportPdf merupakan proses untuk menghasilkan file PDF berisi data transaksi yang difilter berdasarkan tanggal dan tipe transaksi yang dipilih pengguna. Fungsi ini mengambil data pengguna dari Supabase, kemudian membaca *input fromDate*, *toDate*, dan judul laporan dari *form*. Jika tipe transaksi dipilih selain "Semua tipe", sistem memetakan tipe tersebut untuk diterapkan sebagai *filter*. Selanjutnya, *query* dijalankan ke tabel *transactions* untuk mengambil data berdasarkan *user_id* dan rentang tanggal yang ditentukan. Data diurutkan berdasarkan tanggal dan waktu dibuat, lalu difilter ulang jika perlu. Terakhir, dokumen PDF baru dibuat menggunakan *pw.Document()* untuk menampung konten laporan.


```
Future<void> _exportPdf(BuildContext context) async {
  final client = Supabase.instance.client;
  final user = client.auth.currentUser;
  final fromDate = fromDateController.text;
  final toDate = toDateController.text;
  final title = titleController.text;

  final filterType = selectedType == 'Semua tipe'
      ? null
      : transactionTypeMap[selectedType ?? ''];

  final response = await client
      .from('transactions')
      .select()
      .eq('user_id', user!.id)
      .gte('date', fromDate)
      .lte('date', toDate)
      .order('date')
      .order('created_at');

  List data = response;

  if (filterType != null) {
```

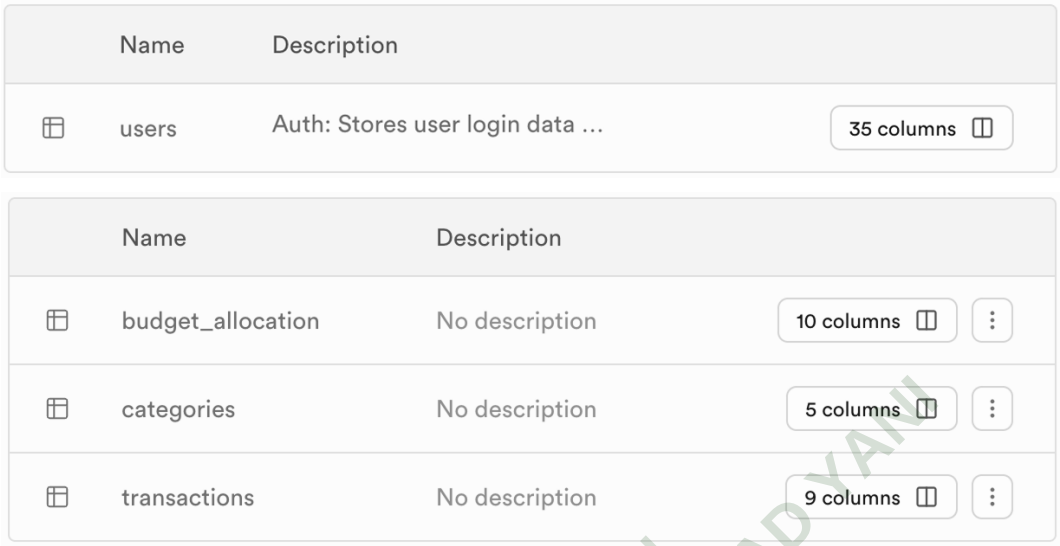
```
    data = data.where((item) => item['type'] ==
filterType).toList();
  }

  final pdf = pw.Document();
```

Source code Printing.layoutPdf memanggil fungsi dari package *printing* untuk menampilkan pratinjau dan mencetak dokumen PDF yang telah dibuat, dengan menyimpan isi dari dokumen PDF yang sebelumnya dikonstruksi. Setelah PDF selesai diproses, *Navigator.pop(context)* digunakan untuk menutup dialog dan kembali ke layar sebelumnya. Pada kode *showCustomSnackBar* menampilkan notifikasi di layar sebagai umpan balik kepada pengguna bahwa proses ekspor data telah berhasil dilakukan.

```
await Printing.layoutPdf(
      onLayout: (format) async => pdf.save(),
    );

    Navigator.pop(context);

    showCustomSnackBar(
        context, 'Data ${titleController.text} berhasil
diekspor');
```


4.3 IMPLEMENTASI DATABASE

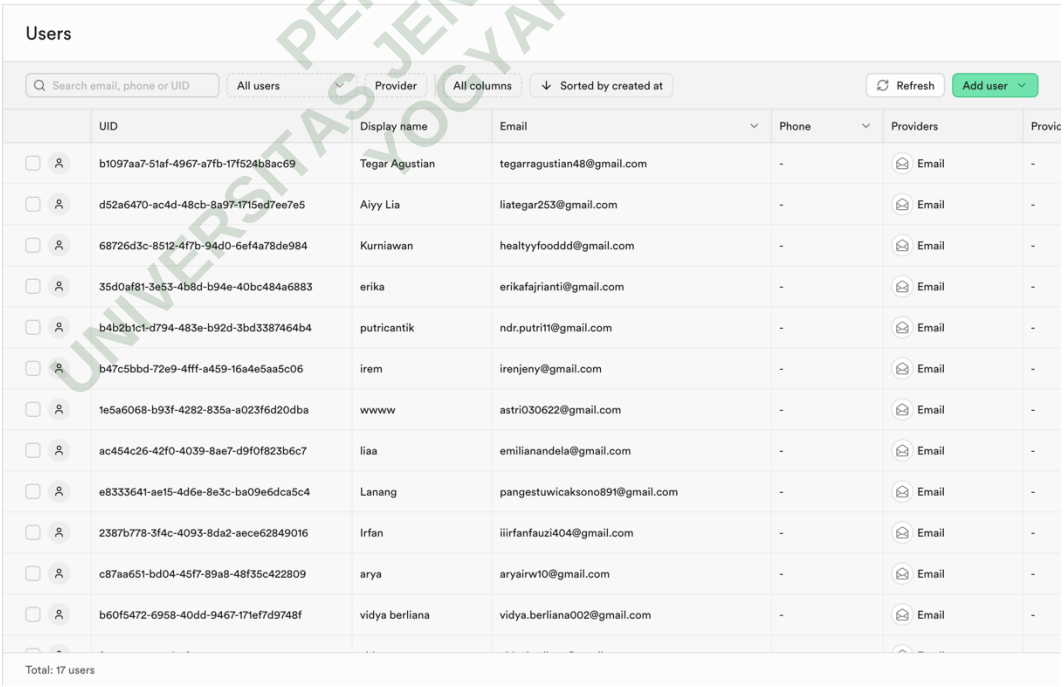
Database adalah kumpulan data yang terstruktur dan sistematis. Sistem Manajemen Keuangan Mahasiswa (Anak Kos) ini dirancang menggunakan layanan Supabase yang berbasis PostgreSQL sebagai basis datanya. Adapun nama *project* Supabase yang digunakan diberi nama Agustiann, yang berfungsi sebagai tempat penyimpanan seluruh data sistem. *Database* pada sistem ini memiliki 3 tabel utama dan juga memanfaatkan 1 tabel bawaan Supabase, yaitu tabel *auth.users*, untuk pengelolaan data autentikasi pengguna. Setiap tabel memiliki sebuah ide yang berfungsi sebagai *primary key* dan memiliki *foreign key* untuk membuat relasi antara tabel yang satu dengan tabel yang lainnya. Implementasi *database* dengan 3 tabel utama dan 1 tabel bawaan Supabase dapat dilihat pada Gambar 4.9.



Gambar 4.9 Implementasi *database* sistem

4.3.1 Implementasi Tabel Pengguna

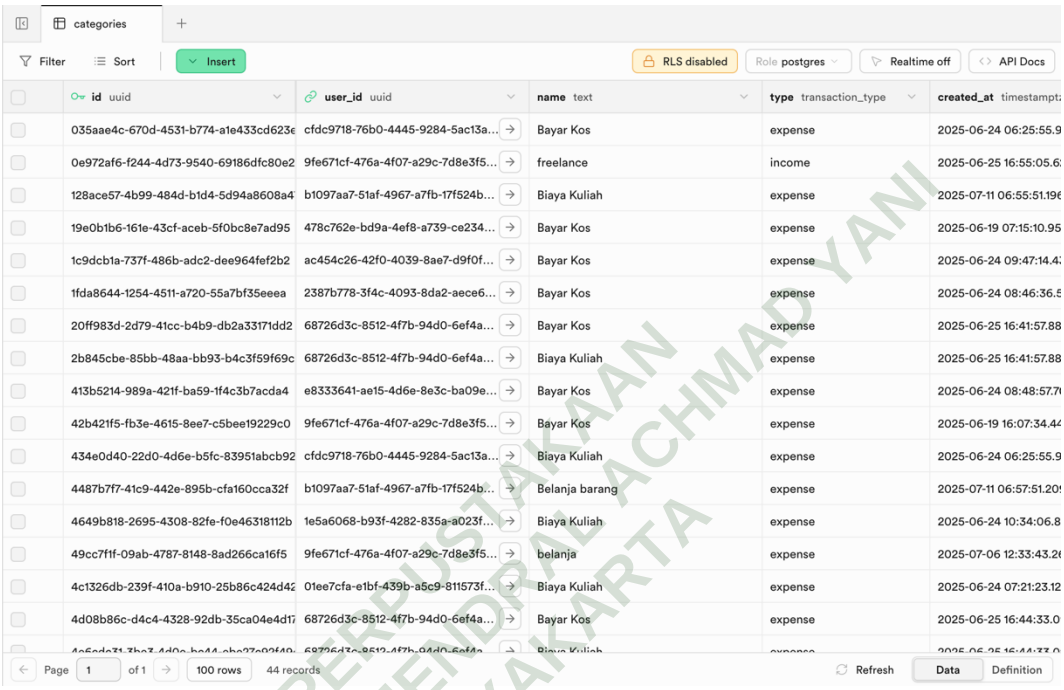
Gambar 4.10 menampilkan implementasi tabel pengguna yang diambil dari tabel bawaan Supabase yaitu tabel *auth.users*. Tabel ini memiliki id sebagai *primary key*.



Gambar 4.10 Implementasi tabel *auth.users*

4.3.2 Implementasi Tabel Kategori

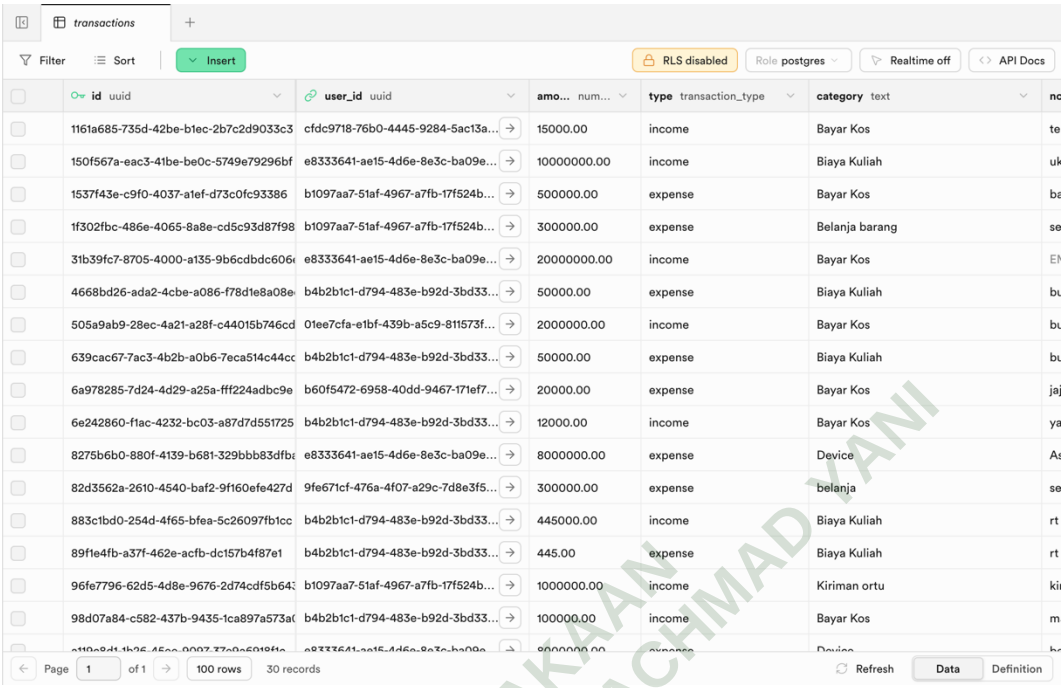
Gambar 4.11 menampilkan implementasi tabel kategori yang diberi nama *categories*. Tabel ini memiliki id sebagai *primary key* dan *user_id* sebagai *foreign key*.



Gambar 4.11 Implementasi tabel *categories*

4.3.3 Implementasi Tabel Transaksi

Gambar 4.12 menampilkan implementasi tabel transaksi yang diberi nama *transactions*. Tabel ini memiliki id sebagai *primary key* dan *user_id* sebagai *foreign key*.



transactions

Filter | Sort | Insert | RLS disabled | Role postgres | Realtime off | API Docs

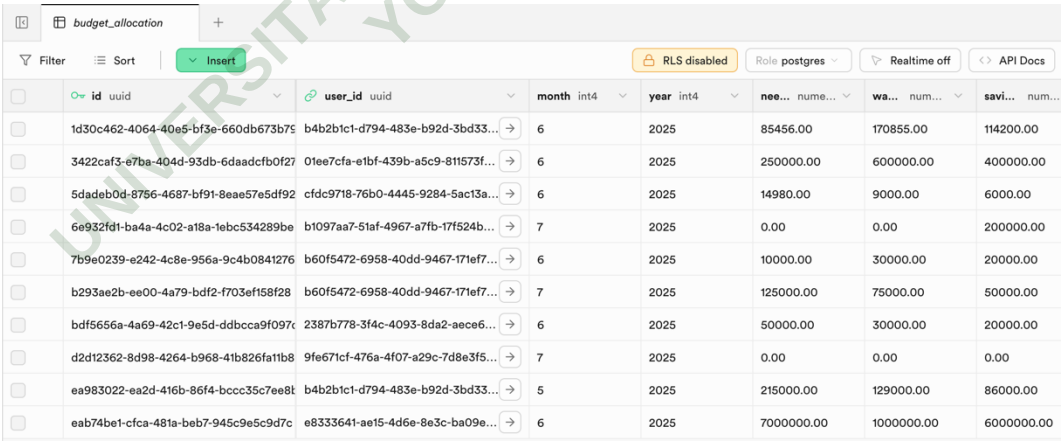
id uuid	user_id uuid	amo... num...	type transaction_type	category text	nc
1161a685-735d-42be-b1ec-2b7c2d9033c3	cfdc9718-76b0-4445-9284-5ac13a...	15000.00	income	Bayar Kos	te
150f567a-eac3-41be-be0c-5749e79296bf	e8333641-ae15-4d6e-8e3c-ba09e...	10000000.00	income	Biaya Kuliah	uk
1537f43e-c9f0-4037-a1ef-d73c0fc93386	b1097aa7-51af-4967-a7fb-17f524b...	500000.00	expense	Bayar Kos	ba
1f302fbc-486e-4065-8a8e-cd5c93d87f98	b1097aa7-51af-4967-a7fb-17f524b...	300000.00	expense	Belanja barang	se
31b39fc7-8705-4000-a135-9b6cdbdc606	e8333641-ae15-4d6e-8e3c-ba09e...	20000000.00	income	Bayar Kos	EM
4668bd26-ada2-4cbe-a086-f78d1e8a08e	b4b2b1c1-d794-483e-b92d-3bd33...	50000.00	expense	Biaya Kuliah	bu
505a9ab9-28ec-4a21-a28f-c44015b746cd	01ee7cfa-e1bf-439b-a5c9-811573f...	2000000.00	income	Bayar Kos	bu
639cac67-7ac3-4b2b-a0b6-7eca514c44cc	b4b2b1c1-d794-483e-b92d-3bd33...	50000.00	expense	Biaya Kuliah	bu
6a978285-7d24-4d29-a25a-fff224adbc9e	b60f5472-6958-40dd-9467-171ef7...	20000.00	expense	Bayar Kos	jaj
6e242860-f1ac-4232-bc03-a87d7d551725	b4b2b1c1-d794-483e-b92d-3bd33...	12000.00	income	Bayar Kos	ya
8275b6b0-880f-4139-b681-329bbb83dfba	e8333641-ae15-4d6e-8e3c-ba09e...	8000000.00	expense	Device	As
82d3562a-2610-4540-baf2-9f160efe427d	9fe671cf-476a-4f07-a29c-7d8e3f5...	300000.00	expense	belanja	se
883c1bd0-254d-4f65-bfea-5c26097fb1cc	b4b2b1c1-d794-483e-b92d-3bd33...	445000.00	income	Biaya Kuliah	rt
89f1e4fb-a37f-462e-acfb-dc157b4f87e1	b4b2b1c1-d794-483e-b92d-3bd33...	445.00	expense	Biaya Kuliah	rt
96fe7796-62d5-4d8e-9676-2d74cdf5b643	b1097aa7-51af-4967-a7fb-17f524b...	1000000.00	income	Kiriman ortu	ki
98d07a84-c582-437b-9435-1ca897a573a0	b4b2b1c1-d794-483e-b92d-3bd33...	100000.00	income	Bayar Kos	m

Page 1 of 1 | 100 rows | 30 records | Refresh | Data | Definition

Gambar 4.12 Implementasi tabel *transactions*

4.3.4 Implementasi Tabel Alokasi Anggaran

Gambar 4.13 menampilkan implementasi tabel alokasi anggaran yang diberi nama *budget_allocation.* Tabel ini memiliki id sebagai *primary key* dan *user_id* sebagai *foreign key.*



budget_allocation

Filter | Sort | Insert | RLS disabled | Role postgres | Realtime off | API Docs

id uuid	user_id uuid	month int4	year int4	nee... nume...	wa... num...	savi... num...
1d30c462-4064-40e5-bf3e-660db673b79	b4b2b1c1-d794-483e-b92d-3bd33...	6	2025	85456.00	170855.00	114200.00
3422caf3-e7ba-404d-93db-6daadcfb0f27	01ee7cfa-e1bf-439b-a5c9-811573f...	6	2025	250000.00	600000.00	400000.00
5dadeb0d-8756-4687-bf91-8eae57e5df92	cfdc9718-76b0-4445-9284-5ac13a...	6	2025	14980.00	9000.00	6000.00
6e932fd1-ba4a-4c02-a18a-1ebc534289be	b1097aa7-51af-4967-a7fb-17f524b...	7	2025	0.00	0.00	200000.00
7b9e0239-e242-4c8e-956a-9c4b0841276	b60f5472-6958-40dd-9467-171ef7...	6	2025	10000.00	30000.00	20000.00
b293ae2b-ee00-4a79-bdf2-f703ef158f28	b60f5472-6958-40dd-9467-171ef7...	7	2025	125000.00	75000.00	50000.00
bdf5656a-4a69-42c1-9e5d-ddbcca9f097c	2387b778-3f4c-4093-8da2-aece6...	6	2025	50000.00	30000.00	20000.00
d2d12362-8d98-4264-b968-41b826fa11b8	9fe671cf-476a-4f07-a29c-7d8e3f5...	7	2025	0.00	0.00	0.00
ea983022-ea2d-416b-86f4-bccc35c7ee8b	b4b2b1c1-d794-483e-b92d-3bd33...	5	2025	215000.00	129000.00	86000.00
eab74be1-cfca-481a-beb7-945c9e5c9d7c	e8333641-ae15-4d6e-8e3c-ba09e...	6	2025	7000000.00	1000000.00	6000000.00

Gambar 4.13 Implementasi tabel *budget_allocation*

4.4 IMPLEMENTASI METODE KANBAN

Dalam pendekatan Kanban, alur kerja dibagi menjadi bagian *To Do*, *Doing*, dan *Done*. Setiap langkah diawasi secara visual untuk meningkatkan efisiensi dan

mencegah penumpukan pekerjaan. Berikut implementasi pendekatan Kanban pada sistem.

1. Kanban *board Plan*

   Kanban *board Plan* menunjukkan proses pengumpulan dan analisis kebutuhan pengguna secara sistematis. Tugas diawali dari pengumpulan data literasi keuangan mahasiswa dan perancangan fitur berdasarkan kebutuhan (*To Do*), kemudian dilanjutkan dengan analisis masalah dan penentuan fitur utama (*Doing*), hingga seluruh kebutuhan pengguna teridentifikasi dan daftar fitur aplikasi selesai ditentukan (*Done*). Gambar 4.14 menampilkan Kanban *board plan.*

Plan		
To Do	**Doing**	**Done**
• Pengumpulan data literasi keuangan mahasiswa pada SNLIK 2024 dari OJK. • Rancang fitur berdasarkan kebutuhan.		

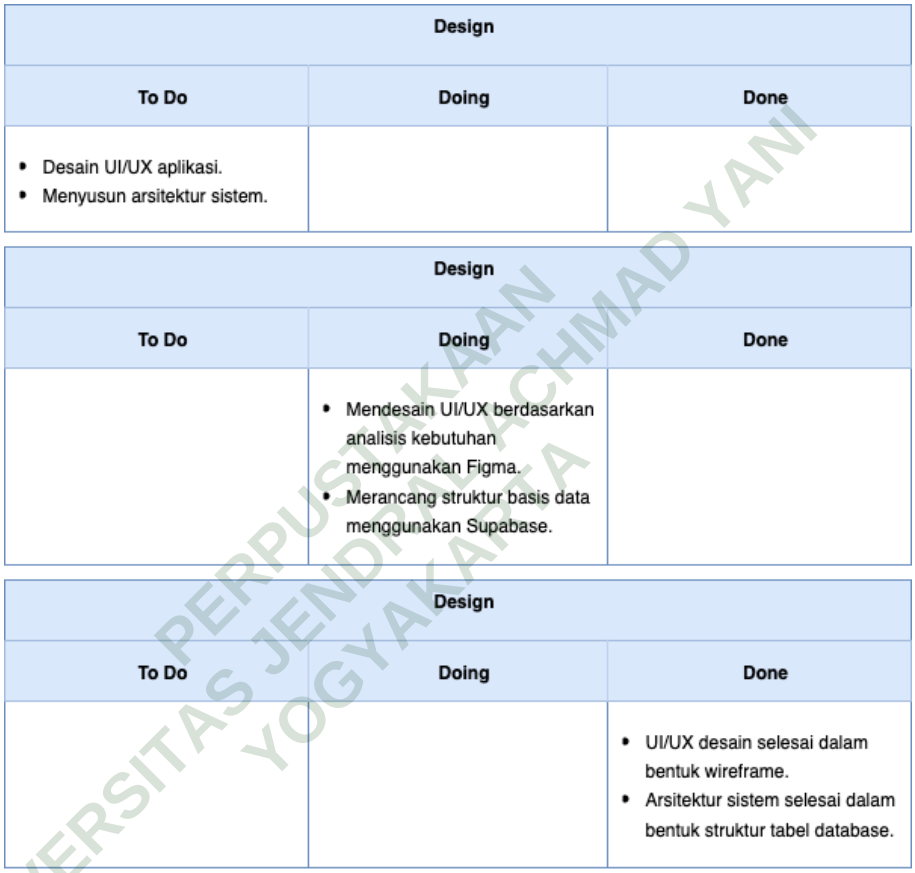
Plan		
To Do	**Doing**	**Done**
	• Analisis masalah keuangan mahasiswa. • Menentukan daftar fitur utama aplikasi.	

Plan		
To Do	**Doing**	**Done**
		• Identifikasi kebutuhan pengguna dengan hasil permasalahan mahasiswa dalam mengatur keuangan berdasarkan literasi keuangan mahasiswa. • Daftar fitur aplikasi sesuai kebutuhan pengguna.

Gambar 4.14 Kanban *board plan*

2. Kanban *board Design*

Kanban *board Design* menunjukkan proses desain aplikasi dengan menyusun desain UI/UX, dan arsitektur sistem. Setiap tugas dipindahkan dari *To Do* ke *Doing*, kemudian ke *Done* setelah selesai. Gambar 4.15 menampilkan Kanban *board develop.*



Gambar 4.15 Kanban *board design*

3. Kanban *board Develop*

Gambar 4.16 menampilkan Kanban *board Develop* yaitu proses pengembangan aplikasi secara bertahap dengan mengembangkan seluruh fitur utama aplikasi yang dilakukan dari *To Do* ke *Doing*, kemudian ke *Done.*

Develop		
To Do	**Doing**	**Done**
• Menyusun dan mengembangkan fitur login dan register. • Menyusun dan mengembangkan fitur kategori (add, update, delete). • Menyusun dan mengembangkan fitur tambah transaksi. • Menyusun dan mengembangkan fitur pos anggaran • Menyusun dan mengembangkan fitur laporan keuangan. • Menyusun dan mengembangkan fitur download.		

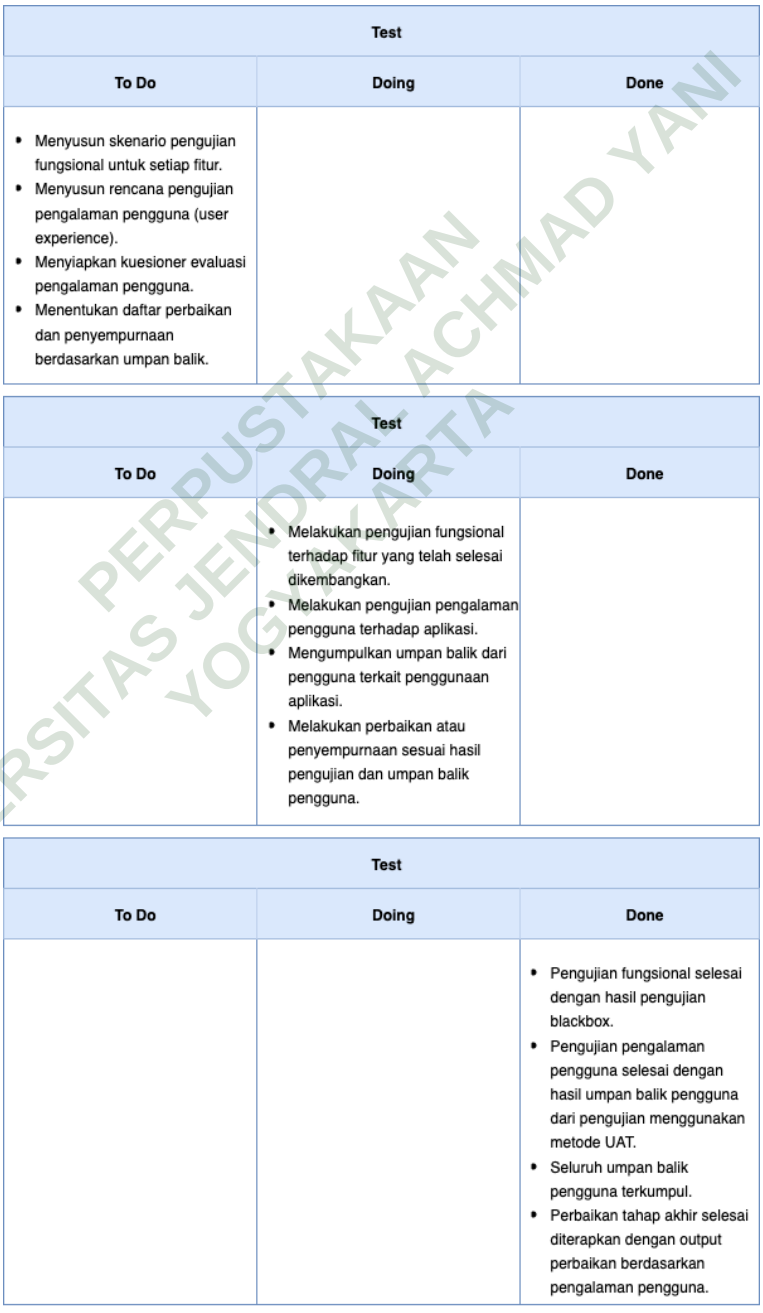
Develop		
To Do	**Doing**	**Done**
	• Mengembangkan fitur login dan register. • Mengembangkan fitur kategori transaksi (add, update, delete). • Mengembangkan fitur tambah transaksi. • Mengembangkan fitur pos anggaran. • Mengembangkan fitur laporan keuangan. • Mengembangkan fitur download.	

Develop		
To Do	**Doing**	**Done**
		• Implementasi fitur login dan register selesai. • Implementasi fitur kategori selesai. • Implementasi fitur tambah transaksi selesai. • Implementasi fitur pos anggaran selesai. • Implementasi fitur laporan keuangan selesai. • Implementasi fitur download selesai.

Gambar 4.16 Kanban *board develop*

4. Kanban *board Test*

Gambar 4.17 menunjukkan Kanban *board Test* dengan proses pengujian aplikasi dengan setiap tugasnya dimulai dari *To Do* ke *Doing*, kemudian ke *Done*. Pengujian awal dilakukan secara internal kemudian, uji fungsional dan pengalaman pengguna dilakukan. Sampai semua pengujian selesai, umpan balik dikumpulkan, dan perbaikan akhir diterapkan.



Test		
To Do	**Doing**	**Done**
• Menyusun skenario pengujian fungsional untuk setiap fitur. • Menyusun rencana pengujian pengalaman pengguna (user experience). • Menyiapkan kuesioner evaluasi pengalaman pengguna. • Menentukan daftar perbaikan dan penyempurnaan berdasarkan umpan balik.		

Test		
To Do	**Doing**	**Done**
	• Melakukan pengujian fungsional terhadap fitur yang telah selesai dikembangkan. • Melakukan pengujian pengalaman pengguna terhadap aplikasi. • Mengumpulkan umpan balik dari pengguna terkait penggunaan aplikasi. • Melakukan perbaikan atau penyempurnaan sesuai hasil pengujian dan umpan balik pengguna.	

Test		
To Do	**Doing**	**Done**
		• Pengujian fungsional selesai dengan hasil pengujian blackbox. • Pengujian pengalaman pengguna selesai dengan hasil umpan balik pengguna dari pengujian menggunakan metode UAT. • Seluruh umpan balik pengguna terkumpul. • Perbaikan tahap akhir selesai diterapkan dengan output perbaikan berdasarkan pengalaman pengguna.

Gambar 4.17 Kanban *board test*

5. Kanban *board Deploy*

Gambar 4.18 menampilkan Kanban *board Deploy*, yaitu proses implementasi sistem yang dilakukan secara bertahap. Aktivitas dimulai dari tahap *To Do*, dilanjutkan ke *Doing*, kemudian ke *Done*.

Deploy		
To Do	**Doing**	**Done**
• Mempersiapkan sistem untuk implementasi.		

Deploy		
To Do	**Doing**	**Done**
	• Implementasi sistem ke lingkungan pengguna.	

Deploy		
To Do	**Doing**	**Done**
		• Sistem berhasil diterapkan ke lingkungan pengguna.

Gambar 4.18 Kanban *board deploy*

6. Kanban *board Review*

Gambar 4.19 menampilkan Kanban *board Review*, yaitu proses evaluasi hasil implementasi sistem yang tugasnya dimulai dari *To Do* ke *Doing*, kemudian ke *Done*.

Review		
To Do	**Doing**	**Done**
• Menganalisis hasil implementasi awal pada penerapan ke lingkungan pengguna. • Melakukan perbaikan sistem dari saran dan umpan balik.		

Review		
To Do	**Doing**	**Done**
	• Mengevaluasi kekurangan dan kelebihan aplikasi. • Implementasi perbaikan sistem.	

Review		
To Do	**Doing**	**Done**
		• Daftar masukan perbaikan dari saran dan umpan balik pengguna. • Sistem telah diperbaiki sesuai saran dan umpan balik pengguna.

Gambar 4.19 Kanban *board review*

7. Kanban *board Launch*

Gambar 4.20 menampilkan *Kanban board Launch*, yaitu proses peluncuran akhir aplikasi setelah semua tahapan pengembangan selesai. Proses dilakukan mulai dari tahap *To Do*, dilanjutkan ke *Doing*, kemudian ke *Done.*

Launch		
To Do	**Doing**	**Done**
• Persiapan akhir distribusi aplikasi.		

Launch		
To Do	**Doing**	**Done**
	• Perbaikan dan finalisasi aplikasi berdasarkan hasil review.	

Launch		
To Do	**Doing**	**Done**
		• Sistem diluncurkan ke pengguna akhir dalam bentuk aplikasi.

Gambar 4.20 Kanban *board launch*

4.5 PENGUJIAN SISTEM

Pengujian sistem dilakukan melalui dua tahap untuk memastikan setiap fitur dalam aplikasi berjalan sebagaimana mestinya serta bebas dari kesalahan saat digunakan. Tahapan tersebut meliputi pengujian internal menggunakan metode *black box testing* untuk mengidentifikasi fungsionalitas masing-masing fitur, dan pengujian *User Acceptance Test* (UAT) yang bertujuan memastikan aplikasi telah sesuai dengan kebutuhan dan harapan mahasiswa sebagai pengguna akhir.

4.5.1 Pengujian *Black Box Testing*

Pengujian black box testing dilakukan dengan pengujian perangkat lunak yang berfokus pada fungsionalitas perangkat lunak tanpa memperhatikan struktur internal atau kode programnya. Tujuan pengujian ini adalah untuk memastikan bahwa setiap fitur atau fungsi yang tersedia dalam sistem dapat berjalan sesuai dengan kebutuhan dan menghasilkan *output* yang sesuai dengan harapan.

1. Pengujian Fitur *Login*

   Pengujian fitur *login* dapat dilihat pada Tabel 4.1

Tabel 4.1 *Black box testing login*

Fungsi yang diuji	Skenario pengujian	Hasil yang diharapkan	Kesimpulan
Login (Berhasil)	*User* memasukkan data *email* dan *password* yang benar	*User* berhasil login, diarahkan ke halaman *dashboard*, serta session *user* aktif	Berhasil
Login (*Email* salah)	*User* memasukkan data *email* salah, dan password benar	Pesan *eror* "*Email* atau kata sandi salah"	Berhasil
Login (*Password* salah)	*User* memasukkan data *email* benar, dan password salah	Pesan *eror* "*Email* atau kata sandi salah"	Berhasil
Login (*Input* kosong)	*User* tidak memasukkan data *email*, dan password	Pesan *eror* "*Email* dan kata harus diisi"	Berhasil

2. Pengujian Fitur *Register*

Pengujian fitur register dapat dilihat pada Tabel 4.2.

Tabel 4.2 *Black box testing register*

Fungsi yang diuji	Skenario pengujian	Hasil yang diharapkan	Kesimpulan
Register (Berhasil)	*User* memasukkan data nama, *email* dan *password* yang benar	*User* berhasil daftar, diarahkan ke halaman *login*, dan mendapatkan email dari Supabase	Berhasil
Register (*Email* salah)	*User* memasukkan data *email* tanpa format	Pesan *eror* "Format *email* salah"	Berhasil

Register (*Email* salah)	*User* memasukkan data *email* yang tidak tersedia	Pesan *eror* "*email* tidak valid atau tidak aktif"	Berhasil
Register (*Password* salah)	*User* memasukkan password kurang dari 6 karakter	Pesan *eror* "*Password harus 6 karakter atau lebih*"	Berhasil
Register (*Input* kosong)	*User* tidak memasukkan data nama, *email*, dan password	Pesan *eror* "*Isi semua data*"	Berhasil

3. Pengujian Fitur *Dashboard*

Pengujian fitur *dashboard* dapat dilihat pada Tabel 4.3.

Tabel 4.3 *Black box testing dashboard*

Fungsi yang diuji	Skenario pengujian	Hasil yang diharapkan	Kesimpulan
Data transaksi pemasukan (Berhasil)	*User* melakukan transaksi pemasukan	*Dashboard* menampilkan list transaksi pemasukan, dan menambahkan data total transaksi pemasukan	Berhasil
Data transaksi pengeluaran (Berhasil)	*User* melakukan transaksi pengeluaran	*Dashboard* menampilkan list transaksi pengeluaran, dan menambahkan data total transaksi pengeluaran dan menambah persentase grafik pengeluaran bulanan sesuai sumber pengeluaran	Berhasil

4. Pengujian Fitur Pos Anggaran

Pengujian fitur pos anggaran dapat dilihat pada Tabel 4.4.

Tabel 4.4 *Black box testing* pos anggaran

Fungsi yang diuji	Skenario pengujian	Hasil yang diharapkan	Kesimpulan
Data total dana, dan dana pos anggaran (Berhasil)	*User* melakukan transaksi pemasukan	Data pos anggaran yang di tampilkan sesuai dengan pembagian 50/30/20 dan total dana sesuai dengan penjumlahan seluruh dana	Berhasil

5. Pengujian Fitur Kategori

Pengujian fitur kategori dapat dilihat pada Tabel 4.5.

Tabel 4.5 *Black box testing* kategori

Fungsi yang diuji	Skenario pengujian	Hasil yang diharapkan	Kesimpulan
Tambah kategori (Berhasil)	*User* memasukkan data kategori pemasukan atau pengeluaran	Data berhasil ditambahkan dan keluar di list kategori	Berhasil
Ubah kategori (Berhasil)	*User* mengubah data kategori pemasukan atau pengeluaran	Data berhasil diubah dan di list kategori juga berubah	Berhasil
Hapus kategori (Berhasil)	*User* menghapus data kategori pemasukan atau pengeluaran	Data berhasil dihapus dan di list kategori juga datanya hilang	Berhasil

6. Pengujian Fitur Transaksi

Pengujian fitur transaksi dapat dilihat pada Tabel 4.6.

Tabel 4.6 *Black box testing* kategori

Fungsi yang diuji	Skenario pengujian	Hasil yang diharapkan	Kesimpulan
Tambah transaksi (Berhasil)	*User* memasukkan data transaksi pemasukan dan pengeluaran	*User* diarahkan ke halaman *dashboard* dan list transaksi serta total transaksi ditambahkan	Berhasil
Tambah transaksi (*Input* kosong)	*User* tidak memasukkan data transaksi	Pesan *eror* "Mohon isi semua data"	Berhasil
Tambah transaksi pengeluaran (Sumber dana tidak cukup)	*User* memasukkan data dana melebihi saldo pada sumber anggaran	Pesan *eror "Dana sumber anggaran tidak cukup*"	Berhasil

7. Pengujian Fitur Laporan Keuangan

Pengujian fitur laporan keuangan dapat dilihat pada Tabel 4.7.

Tabel 4.7 *Black box testing* laporan keuangan

Fungsi yang diuji	Skenario pengujian	Hasil yang diharapkan	Kesimpulan
Data laporan keuangan (Berhasil)	*User* melakukan transaksi	Data laporan keuangan, yaitu total pemasukan dan pengeluaran bulanan dan tahunan tampil	Berhasil

8. Pengujian Fitur Download

Pengujian fitur *download* dapat dilihat pada Tabel 4.8.

Tabel 4.8 *Black box testing download*

Fungsi yang diuji	Skenario pengujian	Hasil yang diharapkan	Kesimpulan
Download (Berhasil)	*User* memasukkan data judul, tanggal mulai, tanggal selesai, serta tipe transaksi	Menampilkan preview *download* dan *user* akan *download*, kemudian pesan berhasil "Data berhasil didownload"	Berhasil

4.5.2 Pengujian *User Acceptance Testing* (UAT)

Pengujian User Acceptance Testing dilakukan untuk memastikan bahwa aplikasi telah sesuai dengan kebutuhan, harapan, dan standar yang diinginkan oleh pengguna akhir, yaitu para mahasiswa. Proses pengujian ini melibatkan pengguna secara langsung untuk mengevaluasi apakah aplikasi dapat berfungsi dengan baik dalam situasi yang sebenarnya.

1. Skor Skala *Likert*

Pengujian ini dilakukan dengan menggunakan kuesioner untuk mendapatkan evaluasi mahasiswa sebagai pengguna akhir. Kuesioner disebarkan dan melibatkan 28 responden mahasiswa dari berbagai instansi. Kuesioner terdiri dari beberapa pernyataan pengguna dengan menggunakan skala pengukuran teknik *Likert* untuk menghitung skor. Tabel 4.9 menampilkan informasi skala *Likert.*

Tabel 4.9 Tabel skala *Likert*

Skala Jawaban	Keterangan	Skor	Persentase
SS	Sangat Setuju	5	100% - 80%
S	Setuju	4	79% - 60%
MS	Mungkin Setuju	3	59% - 40%
TS	Tidak Setuju	2	39% - 20%
ST	Sangat Tidak Setuju	1	19% - 0%

Dengan menggunakan rumus berikut, data yang diperoleh dari kuesioner dapat dianalisis dengan menghitung rata-rata jawaban berdasarkan skor setiap jawaban responden.

$$P = \frac{S}{Skor\ Ideal} \times 100\% \quad (1)$$

Keterangan:

P = Persentase

S = Jumlah frekuensi jawaban dikalikan dengan skor tiap jawaban

Skor Ideal = Skor tertinggi dikalikan dengan jumlah responden

Skor Ideal $= 5 \times$ jumlah responden

$= 5 \times 28$

$= 140$

Berikut pertanyaan hasil persentase setiap jawaban yang didapatkan dari kuesioner :

a. Aplikasi ini mudah digunakan meskipun saya baru pertama kali mencobanya.

Tabel 4.10 Skor pertanyaan 1

Jawaban	Skor	Frekuensi	S
SS	5	12	$5 \times 12 = 60$
S	4	11	$4 \times 11 = 44$
MS	3	4	$3 \times 4 = 12$
TS	2	0	$2 \times 0 = 0$
ST	1	1	$1 \times 1 = 1$
Jumlah			117

$$P = \frac{117}{140} \times 100\%$$

$= 83{,}57\%$

$= 84\%$

Berdasarkan persentase nilai pada pernyataan 1, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 84% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

b. Saya dapat dengan cepat memahami cara kerja aplikasi ini.

Tabel 4.11 Skor pertanyaan 2

Jawaban	Skor	Frekuensi	S
SS	5	14	$5 \times 14 = 70$
S	4	10	$4 \times 10 = 40$
MS	3	3	$3 \times 3 = 9$
TS	2	0	$2 \times 0 = 0$
ST	1	1	$1 \times 1 = 1$
Jumlah			120

$$P = \frac{120}{140} \times 100\%$$
$$= 85{,}71\%$$
$$= 86\%$$

Berdasarkan persentase nilai pada pernyataan 2, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 86% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

c. Navigasi antar fitur sangat mudah dan intuitif.

Tabel 4.12 Skor pertanyaan 3

Jawaban	Skor	Frekuensi	S
SS	5	13	$5 \times 13 = 65$
S	4	10	$4 \times 10 = 40$
MS	3	3	$3 \times 3 = 9$
TS	2	1	$2 \times 1 = 2$
ST	1	1	$1 \times 1 = 1$
Jumlah			117

$$P = \frac{117}{140} \times 100\%$$
$$= 83{,}57\%$$
$$= 84\%$$

Berdasarkan persentase nilai pada pernyataan 3, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 84% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

d. Tampilan antarmuka aplikasi ini menarik dan nyaman dilihat.

Tabel 4.13 Skor pertanyaan 4

Jawaban	Skor	Frekuensi	S
SS	5	8	$5 \times 8 = 40$
S	4	13	$4 \times 13 = 52$
MS	3	5	$3 \times 5 = 15$
TS	2	1	$2 \times 1 = 2$
ST	1	1	$1 \times 1 = 1$
Jumlah			110

$$P = \frac{110}{140} \times 100\%$$
$$= 78{,}57\%$$
$$= 79\%$$

Berdasarkan persentase nilai pada pernyataan 4, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 79% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai setuju.

e. Ukuran teks, ikon, dan tombol dalam aplikasi cukup jelas dan mudah dibaca.

Tabel 4.14 Skor pertanyaan 5

Jawaban	Skor	Frekuensi	S
SS	5	12	$5 \times 12 = 60$
S	4	13	$4 \times 13 = 52$
MS	3	1	$3 \times 1 = 3$
TS	2	1	$2 \times 1 = 2$
ST	1	2	$1 \times 2 = 2$
Jumlah			119

$$P = \frac{119}{140} \times 100\%$$

$= 85\%$

Berdasarkan persentase nilai pada pernyataan 5, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 85% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

f. Warna dan elemen desain memudahkan saya dalam menggunakan aplikasi.

Tabel 4.15 Skor pertanyaan 6

Jawaban	Skor	Frekuensi	S
SS	5	12	$5 \times 12 = 60$
S	4	11	$4 \times 11 = 44$
MS	3	2	$3 \times 2 = 6$
TS	2	2	$2 \times 2 = 4$
ST	1	1	$1 \times 1 = 1$
Jumlah			115

$$P = \frac{115}{140} \times 100\%$$
$$= 82{,}14\%$$
$$= 82\%$$

Berdasarkan persentase nilai pada pernyataan 6, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 82% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

g. Semua fitur dalam aplikasi berjalan dengan baik.

Tabel 4.16 Skor pertanyaan 7

Jawaban	Skor	Frekuensi	S
SS	5	13	$5 \times 13 = 65$
S	4	10	$4 \times 10 = 40$
MS	3	2	$3 \times 2 = 6$
TS	2	2	$2 \times 2 = 4$
ST	1	1	$1 \times 1 = 1$
Jumlah			116

$$P = \frac{116}{140} \times 100\%$$
$$= 82{,}86\%$$
$$= 83\%$$

Berdasarkan persentase nilai pada pernyataan 7, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 83% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

h. Laporan bulanan yang dihasilkan aplikasi membantu saya memahami kondisi keuangan saya.

Tabel 4.17 Skor pertanyaan 8

Jawaban	Skor	Frekuensi	S
SS	5	19	$5 \times 19 = 95$
S	4	7	$4 \times 7 = 28$
MS	3	0	$3 \times 0 = 0$
TS	2	1	$2 \times 1 = 2$
ST	1	1	$1 \times 1 = 1$
Jumlah			126

$$P = \frac{126}{140} \times 100\%$$
$$= 90\%$$

Berdasarkan persentase nilai pada pernyataan 8, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 90% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

i. Fitur pos anggaran (kebutuhan, keinginan, tabungan) membantu saya mengalokasikan dana dengan lebih bijak.

Tabel 4.18 Skor pertanyaan 9

Jawaban	Skor	Frekuensi	S
SS	5	17	$5 \times 17 = 85$
S	4	9	$4 \times 9 = 36$
MS	3	0	$3 \times 0 = 0$
TS	2	1	$2 \times 1 = 2$
ST	1	1	$1 \times 1 = 1$
Jumlah			124

$$P = \frac{124}{140} \times 100\%$$
$$= 88{,}57\%$$
$$= 89\%$$

Berdasarkan persentase nilai pada pernyataan 9, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 89% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

j. Adanya kategori tetap seperti biaya kuliah dan kos sangat membantu mencatat pengeluaran penting sebagai Mahasiswa (Anak kos).

Tabel 4.19 Skor pertanyaan 10

Jawaban	Skor	Frekuensi	S
SS	5	19	$5 \times 19 = 95$
S	4	7	$4 \times 7 = 28$
MS	3	0	$3 \times 0 = 0$
TS	2	1	$2 \times 1 = 2$
ST	1	1	$1 \times 1 = 1$
Jumlah			126

$$P = \frac{126}{140} \times 100\%$$
$$= 90\%$$

Berdasarkan persentase nilai pada pernyataan 10, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 90% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

k. Saya merasa fitur-fitur yang tersedia sudah sesuai dengan kebutuhan saya.

Tabel 4.20 Skor pertanyaan 11

Jawaban	Skor	Frekuensi	S
SS	5	13	$5 \times 13 = 65$
S	4	8	$4 \times 8 = 32$
MS	3	6	$3 \times 6 = 18$
TS	2	0	$2 \times 0 = 0$
ST	1	1	$1 \times 1 = 1$
Jumlah			116

$$P = \frac{116}{140} \times 100\%$$
$$= 82{,}86\%$$
$$= 83\%$$

Berdasarkan persentase nilai pada pernyataan 11, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 83% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

l. Aplikasi ini membantu saya lebih sadar terhadap pengeluaran saya.

Tabel 4.21 Skor pertanyaan 12

Jawaban	Skor	Frekuensi	S
SS	5	16	$5 \times 16 = 80$
S	4	8	$4 \times 8 = 32$
MS	3	2	$3 \times 2 = 6$
TS	2	1	$2 \times 1 = 2$
ST	1	1	$1 \times 1 = 1$
Jumlah			121

$$P = \frac{121}{140} \times 100\%$$
$$= 86{,}43\%$$
$$= 86\%$$

Berdasarkan persentase nilai pada pernyataan 12, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 86% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

m. Dengan menggunakan aplikasi ini, saya merasa mengelola keuangan akan lebih teratur.

Tabel 4.22 Skor pertanyaan 13

Jawaban	Skor	Frekuensi	S
SS	5	17	$5 \times 17 = 85$
S	4	7	$4 \times 7 = 28$
MS	3	2	$3 \times 2 = 6$
TS	2	1	$2 \times 1 = 2$
ST	1	1	$1 \times 1 = 1$
Jumlah			122

$$P = \frac{122}{140} \times 100\%$$
$$= 87{,}14\%$$
$$= 87\%$$

Berdasarkan persentase nilai pada pernyataan 13, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 87% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

n. Aplikasi ini membantu saya menghindari pengeluaran yang tidak perlu.

Tabel 4.23 Skor pertanyaan 14

Jawaban	Skor	Frekuensi	S
SS	5	14	$5 \times 14 = 70$
S	4	8	$4 \times 8 = 32$
MS	3	2	$3 \times 2 = 6$
TS	2	3	$2 \times 3 = 6$
ST	1	1	$1 \times 1 = 1$
Jumlah			115

$$P = \frac{115}{140} \times 100\%$$

$= 82{,}14\%$

$= 82\%$

Berdasarkan persentase nilai pada pernyataan 14, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 82% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

o. Aplikasi ini berkontribusi dalam meningkatkan literasi keuangan saya.

Tabel 4.24 Skor pertanyaan 15

Jawaban	Skor	Frekuensi	S
SS	5	13	$5 \times 13 = 65$
S	4	9	$4 \times 9 = 36$
MS	3	4	$3 \times 4 = 12$
TS	2	1	$2 \times 1 = 2$
ST	1	1	$1 \times 1 = 1$
Jumlah			116

$$P = \frac{116}{140} \times 100\%$$

$= 82{,}86\%$

$= 83\%$

Berdasarkan persentase nilai pada pernyataan 5, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 83% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

p. Saya puas dengan pengalaman menggunakan aplikasi ini secara keseluruhan.

Tabel 4.25 Skor pertanyaan 16

Jawaban	Skor	Frekuensi	S
SS	5	17	$5 \times 17 = 85$
S	4	8	$4 \times 8 = 32$
MS	3	2	$3 \times 2 = 6$
TS	2	0	$2 \times 0 = 0$
ST	1	1	$1 \times 1 = 1$
Jumlah			124

$$P = \frac{124}{140} \times 100\%$$
$$= 88{,}57\%$$
$$= 89\%$$

Berdasarkan persentase nilai pada pernyataan 16, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 89% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

q. Aplikasi ini memenuhi kebutuhan saya sebagai mahasiswa dalam mengatur keuangan.

Tabel 4.26 Skor pertanyaan 17

Jawaban	Skor	Frekuensi	S
SS	5	15	$5 \times 15 = 75$
S	4	8	$4 \times 8 = 32$
MS	3	3	$3 \times 3 = 9$
TS	2	1	$2 \times 1 = 2$
ST	1	1	$1 \times 1 = 1$
Jumlah			119

$$P = \frac{119}{140} \times 100\%$$
$$= 85\%$$

Berdasarkan persentase nilai pada pernyataan 17, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 85% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

r. Saya akan terus menggunakan aplikasi ini ke depannya.

Tabel 4.27 Skor pertanyaan 18

Jawaban	Skor	Frekuensi	S
SS	5	10	$5 \times 10 = 50$
S	4	8	$4 \times 8 = 32$
MS	3	9	$3 \times 9 = 27$
TS	2	0	$2 \times 0 = 0$
ST	1	1	$1 \times 1 = 1$
Jumlah			110

$$P = \frac{110}{140} \times 100\%$$
$$= 78{,}57\%$$
$$= 79\%$$

Berdasarkan persentase nilai pada pernyataan 18, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 79% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai setuju.

s. Saya akan merekomendasikan aplikasi ini kepada teman-teman saya.

Tabel 4.28 Skor pertanyaan 19

Jawaban	Skor	Frekuensi	S
SS	5	11	$5 \times 11 = 55$
S	4	8	$4 \times 8 = 32$
MS	3	8	$3 \times 8 = 24$
TS	2	0	$2 \times 12 = 0$
ST	1	1	$1 \times 1 = 1$
Jumlah			112

$$P = \frac{112}{140} \times 100\%$$
$$= 80\%$$

Berdasarkan persentase nilai pada pernyataan 19, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 80% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

t. Saya tertarik untuk melihat pengembangan lebih lanjut dari aplikasi ini.

Tabel 4.29 Skor pertanyaan 20

Jawaban	Skor	Frekuensi	S
SS	5	17	$5 \times 17 = 85$
S	4	7	$4 \times 7 = 28$
MS	3	2	$3 \times 2 = 6$
TS	2	1	$2 \times 1 = 2$
ST	1	1	$1 \times 1 = 1$
Jumlah			122

$$P = \frac{122}{140} \times 100\%$$
$$= 87{,}14\%$$
$$= 87\%$$

Berdasarkan persentase nilai pada pernyataan 20, dapat disimpulkan bahwa penilaian terhadap pernyataan tersebut adalah 87% dari 100% yang diharapkan, sehingga dapat dikategorikan sebagai sangat setuju.

2. Umpan Balik Pengguna

Selain hasil kuantitatif, pengguna juga diberikan dua pertanyaan terbuka agar bisa memberikan masukan langsung mengenai aplikasi yang mereka gunakan. Tujuannya adalah untuk memperoleh kesan, pendapat, dan saran dari pengguna akhir yang tidak bisa diukur melalui skala Likert. Dari total 28 responden, semua memberikan jawaban, dan berikut adalah hasil analisis tema dari kedua pertanyaan terbuka tersebut.

a. Apa hal yang paling Anda sukai dari aplikasi ini?

Berdasarkan jawaban dari 28 responden, berikut adalah penjelasan singkat mengenai hal-hal yang paling disenangi dapat dilihat pada Tabel 4.30.

Tabel 4.30 Hal yang disukai pengguna

No	Hal Yang Disukai	Kutipan Jawaban	Kesimpulan
1	Desain dan Tampilan (UI/UX)	a. UI/UX yang modern dan minimalis. b. Tampilannya bagus dan rapi. c. Desain bagus, tidak jadul dan sesuai kebutuhan mahasiswa.	Banyak responden menyukai tampilan aplikasi yang modern, simpel, dan rapi, serta antarmuka yang user-friendly dan mudah dipahami.
2	Kemudahan Penggunaan	a. Sangat mudah dipahami bagi orang yang baru pakai. b. Mudah digunakan dan sederhana. c. Simple jadi mudah.	Aplikasi mudah digunakan dan fiturnya mudah dipahami, bahkan oleh pengguna baru.
3	Fitur Pos Anggaran & Kategorisasi	• Fitur pos anggaran dan adanya diagram lingkaran. • Bisa menambah kategori kebutuhan. • Adanya kategorisasi.	Responden menyukai fitur pos anggaran, kategori, dan fleksibilitas dalam menyesuaikan nama kategori.
4	Manfaat Finansial	• Sangat berguna untuk para mahasiswa dalam mengatur keuangannya. • Pencatatan uang masuk dan keluar memudahkan	Aplikasi membantu mahasiswa mencatat keuangan dan menyusun laporan bulanan untuk evaluasi pengeluaran.

		manajemen keuangan. • Terdapat laporan bulanan.	
5	Fitur Tambahan yang Membantu	• Fitur ekspor dokumen lengkap dengan tanggal, waktu, kebutuhan. • Bisa menjumlahkan semuanya secara otomatis.	Fitur ekspor atau *download* laporan, grafik lingkaran, dan penjumlahan otomatis dianggap sangat membantu.

b. Berikan saran atau umpan balik untuk aplikasi ini.

Berdasarkan jawaban dari 28 responden, berikut adalah penjelasan singkat mengenai saran atau umpan balik pengguna dapat dilihat pada Tabel 4.31.

Tabel 4.31 Umpan balik dan saran pengguna

No	Hal Yang Disukai	Kutipan Jawaban	Kesimpulan
1	Kepuasan Umum	• Sudah sesuai. • Aplikasinya bagus dan keren. • Terima kasih, semua fitur juga oke sangat membantu untuk mengatur keuangan.	Sebagian besar responden menyampaikan bahwa aplikasi sudah sesuai dan tidak ada kendala berarti.
2	Desain dan Tampilan (UI/UX)	• Tombol '+' di menu beranda sebaiknya dibuat lebih besar dan lebih ke tengah. • Tombol sign out terlalu di tengah dan besar.	Beberapa masukan terkait tombol dan tata letak.
3	Warna dan Tampilan Visual	• Warna pada diagram bisa disesuaikan agar lebih terlihat hidup. • Tampilan bisa ditambahkan gambar atau animasi agar tidak monoton.	Beberapa responden menyarankan peningkatan warna atau elemen visual agar lebih menarik dan hidup.
4	Fitur Teknis dan Kinerja	• Dropdown kategori pada pemasukan tidak bisa diklik di HP saya. • Diberikan pesan seperti ‘berhasil’ saat	Masukan teknis terkait performa dan fitur *dropdown*, *snackbar*, dan infor tanggal dan waktu.

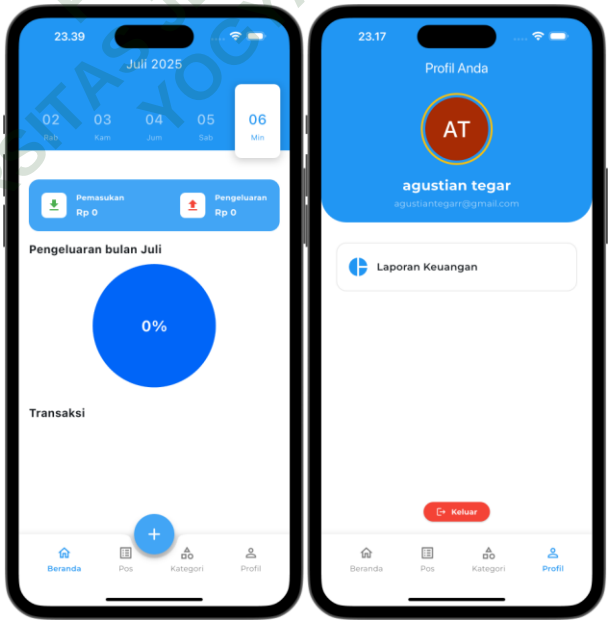
		menambah data. • Diberikan info tanggal dan waktu pada laporan keuangan.	
5	Fitur Pengembangan	• Perluas kalender dan berikan saran untuk berhemat.	Saran untuk pengembangan aplikasi di masa depan.

3. Hasil Umpan Balik

Berdasarkan umpan balik yang diberikan oleh pengguna pada tahap pengujian dan evaluasi, beberapa perbaikan dan penyempurnaan sistem dilakukan untuk memenuhi kebutuhan dan harapan pengguna. Berikut ini adalah hasil implementasi umpan balik:

a. Perbaikan desain dan tampilan

Gambar 4.21 menampilkan hasil perbaikan dilakukan pada tombol "+" dan tombol *signout*.



Gambar 4.21 Hasil umpan balik desain dan tampilan

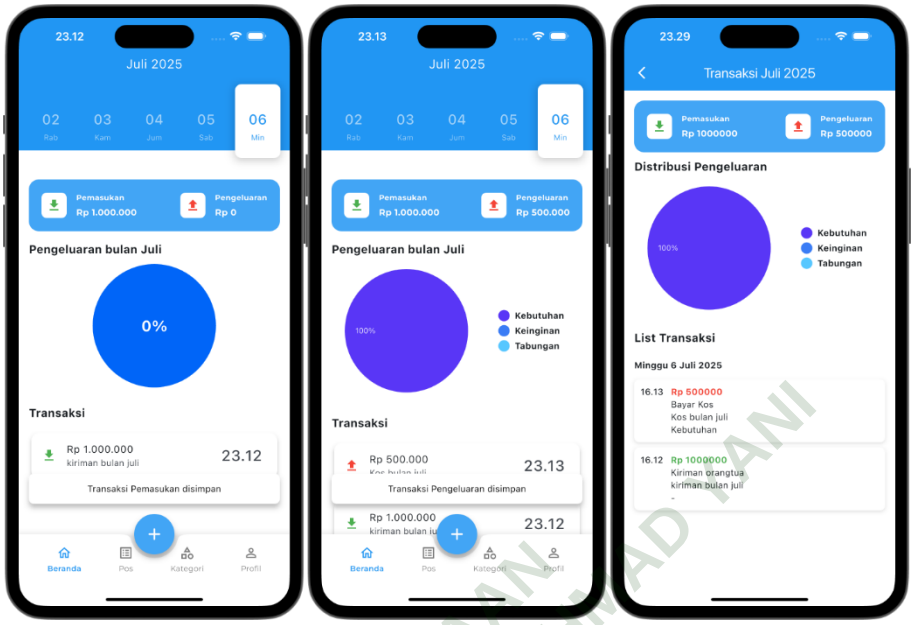
b. Penyempurnaan visual

Gambar 4.1 menampilkan hasil perbaikan tampilan grafik dengan warna yang lebih hidup, yang sekaligus berfungsi sebagai elemen ilustratif dalam antarmuka aplikasi.

Gambar 4.22 Hasil umpan balik penyempurnaan visual

c. Perbaikan fitur teknis

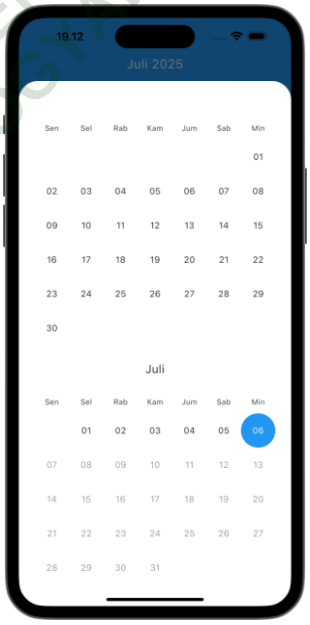
Perbaikan fitur teknis, yaitu dropdown kategori yang kini kompatibel di berbagai perangkat, penambahan notifikasi berhasil saat data transaksi disimpan, serta penampilan informasi tanggal dan waktu pada laporan keuangan untuk memperjelas detail data laporan. Perbaikan fitur teknis dapat dilihat pada Gambar 4.1.



Gambar 4.23 Hasil umpan balik fitur teknis

d. Pengembangan fitur tambahan

Gambar 4.2 menampilkan fitur kalender yang mencakup pilihan tanggal secara luas melalui klik bulan di tampilan beranda.



Gambar 4.24 Hasil umpan balik fitur tambahan

4.5.3 Kesimpulan Pengujian

Berdasarkan hasil evaluasi *User Acceptance Testing* (UAT) yang dilakukan pada 28 mahasiswa responden, dapat disimpulkan bahwa aplikasi ini berhasil memenuhi ekspektasi dan kebutuhan pengguna. Ini terlihat dari rata-rata skor persentase untuk setiap pernyataan yang jatuh dalam kategori "Sangat Setuju", dengan nilai tertinggi mencapai 90% di beberapa elemen penting seperti laporan bulanan, pos anggaran, dan kategori pengeluaran tetap.

Secara umum, responden mengungkapkan bahwa aplikasi ini mudah untuk digunakan, tampilannya menarik dan modern, serta fiturnya sesuai dengan kebutuhan mahasiswa dalam pengelolaan keuangan. Fitur-fitur yang paling disukai meliputi: desain UI/UX yang sederhana, fitur pos anggaran dan kategorisasi, laporan keuangan bulanan, serta kemampuan untuk mengekspor data.

Di sisi lain, 82% dari responden mengungkapkan bahwa aplikasi ini sudah memenuhi standar tanpa memerlukan perbaikan lebih lanjut, sementara yang lainnya memberikan masukan minor, seperti penyesuaian ukuran tombol, perubahan warna pada diagram, penambahan animasi, dan perbaikan teknis pada *dropdown* serta notifikasi. Ini menunjukkan bahwa aplikasi ini sudah siap untuk digunakan dengan hanya beberapa pembaruan kecil untuk meningkatkan pengalaman pengguna.

Dengan demikian, dapat disimpulkan bahwa aplikasi ini diterima dengan sangat baik oleh pengguna, dan telah memenuhi kriteria kelayakan berdasarkan pengujian UAT. Aplikasi ini dinilai berkontribusi dalam membantu mahasiswa mengelola keuangan secara lebih teratur, meningkatkan kesadaran akan pengeluaran, serta mendukung literasi keuangan di kalangan generasi muda.

4.6 PEMBAHASAN HASIL PENGEMBANGAN SISTEM

Bab ini membahas hasil pengembangan dan pengujian sistem manajemen keuangan mahasiswa (anak kos) berbasis mobile yang dikembangkan menggunakan Flutter dan Supabase. Fokus pembahasan ini adalah bagaimana sistem ini mampu memenuhi kebutuhan pengguna, meningkatkan efisiensi, akurasi, serta kemudahan dalam pengelolaan keuangan mahasiswa. Pengembangan sistem

dilakukan menggunakan metode Agile dan pendekatan Kanban, serta disusun dengan strategi pengembangan yang terencana dan terukur.

4.6.1 Kelebihan Sistem

Sistem ini memiliki sejumlah keunggulan yang menunjang keberhasilannya, antara lain:.

1. Fitur pos anggaran dengan metode 50/30/20, yang otomatis membagi pendapatan menjadi 50% untuk kebutuhan pokok, 30% untuk keinginan, dan 20% untuk tabungan serta dana darurat.
2. Kemudahan akses dan pengoperasian aplikasi didukung oleh antarmuka mobile yang responsif dan mudah dipahami, sehingga mahasiswa dapat mengelola keuangan mereka dengan fleksibilitas penuh kapan saja dan di mana saja.
3. Stabilitas dalam integrasi database didukung oleh Supabase yang menyediakan platform pengelolaan data konsisten dan handal untuk mempermudah manajemen transaksi, kategori, serta laporan keuangan.
4. Pencatatan yang komprehensif dan sistem otomatisasi terintegrasi mempermudah pembuatan laporan keuangan, perhitungan anggaran, dan klasifikasi transaksi secara otomatis, serta meminimalkan kesalahan kesalahan operasional.
5. Keamanan data dijaga melalui penerapan autentikasi, otorisasi, serta mekanisme enkripsi yang memadai guna melindungi kerahasiaan dan keutuhan informasi keuangan pengguna.
6. Pengembangan sistem dilakukan secara fleksibel menggunakan metode Agile dan pendekatan Kanban yang memungkinkan pembaruan dan penyesuaian berkelanjutan sesuai kebutuhan pengguna agar sistem tetap relevan dan adaptif.

4.6.2 Kekurangan Sistem

Sistem ini memiliki memiliki kekurangan seperti berikut:

1. Fitur analisis keuangan lanjutan masih memiliki keterbatasan karena sistem belum menyediakan fungsi seperti prediksi pengeluaran dan saran keuangan berbasis sistem.
2. Sistem yang mengandalkan platform *cloud* dan *mobile* membutuhkan koneksi internet yang baik untuk akses dan sinkronisasi data sehingga pengguna di area dengan jaringan yang buruk berpotensi mengalami kesulitan dalam penggunaan aplikasi.
3. Sistem belum mendukung integrasi dengan aplikasi keuangan eksternal atau fitur impor dan ekspor data, sehingga kemampuan untuk bertukar informasi antarplatform masih terbatas.
4. Sistem belum menyediakan integrasi otomatis dengan rekening bank atau layanan pembayaran digital, sehingga proses pencatatan transaksi harus dilakukan secara manual, yang berisiko meningkatkan kesalahan input dan kurang efisien.