DAFTAR LAMPIRAN

Lampiran – 1 List Program Main.dart

```
      Future<void>    main()
async {
      WidgetsFlutterBinding
.ensureInitialized();
      setPathUrlStrategy();
        await
setupLocator(stackedRouter:
stackedRouter);
      setupDialogUi();
      setupBottomSheetUi();
        await
ThemeManager.initialise();
        await
Supabase.initialize(
          url:         const
String.fromEnvironment('SUP
ABASE_URL'),
      anonKey:         const
String.fromEnvironment('SUP
ABASE_ANON_KEY'),
        );
      runApp(const
MainApp());
      }
      class MainApp extends
StatelessWidget {
        const
MainApp({super.key});
        @override
        Widget
build(BuildContext  context)
{
          return
ResponsiveApp(
            builder:     (_)
=>ThemeBuilder(
      defaultThemeMode:
ThemeMode.light,
      darkTheme:
AppThemes.lightTheme,
              //  darkTheme:
AppThemes.darkTheme,
      lightTheme:
AppThemes.lightTheme,
              builder:
(context,      regularTheme,
darkTheme,   themeMode)   =>
Portal(
                child:
MaterialApp.router(
                  theme:
regularTheme,
      darkTheme: darkTheme,
      themeMode: themeMode,
      scrollBehavior:  const
MaterialScrollBehavior().co
pyWith(
      dragDevices: {
      PointerDeviceKind.mou
se,
      PointerDeviceKind.tou
ch,
      PointerDeviceKind.sty
lus,
      PointerDeviceKind.unk
nown
                    },
                  ),
      routerDelegate:
stackedRouter.delegate(),
      routeInformationParse
r:
stackedRouter.defaultRouteP
arser(),
                ),
              ),
            ),

).animate().fadeIn(
                delay:
const
Duration(milliseconds: 50),
                duration:
const
Duration(milliseconds:
400),
              );
        }
      }
```

Lampiran – 2 List Program Ant colony service.dart

```
      class  AntColonyService
{
        final      log      =
getLogger('AntColonyService'
);

List<PengampuJadwalModel>pen
gampuJadwalList;

List<AntSlotModel>antSlotLis
t;
        late   List<AntModel>
ants;
        late
List<List<double>>pheromoneM
atrix;
        double alpha;
        double beta;
        double
evaporationRate;
        int maxIterations;

        final Set<DosenModel>
_dosenSet = {};
        final
Set<MatakuliahModel>
_matakuliahSet = {};
        final
Set<PengampuKelasModel>
_kelasSet = {};
        final
Set<RuanganModel> _ruanganSet
= {};
        final  Set<HariModel>
_hariSet = {};
        final   Set<JamModel>
_jamSet = {};
        Random random;
      AntColonyService(
      this.pengampuJadwalLis
t,
      this.antSlotList, {
      this.alpha = 1,
      this.beta = 1,
      this.evaporationRate  =
.5,
      this.maxIterations    =
100,
        })    :    random    =
Random() {
          ants = [];
      pheromoneMatrix       =
List.generate(
      pengampuJadwalList.len
gth,
            (_)
=>List.filled(antSlotList.le
ngth, 1.0),
          );
        }
        void  initializeAnt()
{
          for            (var
pengampuJadwal              in
pengampuJadwalList) {

_dosenSet.add(pengampuJadwal
.dosen);

_matakuliahSet.add(pengampuJ
adwal.matakuliah);

_kelasSet.add(pengampuJadwal
.kelas);
          }
          ants              =
List.generate(
      pengampuJadwalList.len
gth,
            (index) {
              final
pengampuJadwal              =
pengampuJadwalList.elementAt
(index);
              return
AntModel(
      pengampuJadwal,
      totalAnt:
pengampuJadwalList.length,
      titikDosen:
_dosenSet.toList().indexOf(p
engampuJadwal.dosen),
      titikMatakuliah:
      _matakuliahSet.toList(
).indexOf(pengampuJadwal.mat
akuliah),
```

```
      titikKelas:
_kelasSet.toList().indexOf(p
engampuJadwal.kelas),
              );
            },
          );
          for (var antSlot in
antSlotList) {

_ruanganSet.add(antSlot.ruan
gan);

_hariSet.add(antSlot.hari);

_jamSet.add(antSlot.jam);
          }
        }
        void
updatePheromoneMatrix() {
          for (var i = 0;
i<pengampuJadwalList.length;
i++) {
            for (var j = 0; j
<antSlotList.length; j++) {
      pheromoneMatrix[i][j]
*= evaporationRate;
            }
          }
          // for (var ant in
ants) {
          //              var
deltaPheromone    =    1    /
ant.jadwal.calculateFitness(
);
          //   for (var entry
in
ant.jadwal.assignments.entri
es) {
          //              var
pengampuJadwal = entry.key;
          //      var antSlot
= entry.value;

          //        var i =
pengampuJadwalList.indexOf(p
engampuJadwal);
      //           var j =
antSlotList.indexOf(antSlot)
;
      //
pheromoneMatrix[i][j]      +=
deltaPheromone;
          //   }
          // }
        }

        //
Future<List<JadwalModel>>run
ACO() async {
        void
runACO(SendPortsendPort) {
      initializeAnt();
          List<AntModel>?
bestSchedule;
          var bestFitness =
double.infinity;
          for (var iteration
=        0;        iteration
<maxIterations;  iteration++)
{
            if  (bestSchedule
!= null) {
              ants          =
List.from(bestSchedule);
            }
            for (var ant in
ants) {
              final antSlot =
selectAntSlot(ant);
      ant.setAntSlot(antSlot
);
              //
ant.setTitik(
              //
ruanganSet: _ruanganSet,
              //     hariSet:
_hariSet,
              //      jamSet:
_jamSet,
              // );
            }
            for (var ant in
ants) {
              //
ant.hitungJarakAntaraAnts(an
ts);
      ant.hitungBentrok(ants
, antSlotList);
            }
            var   fitness   =
calculateFitness();
            if       (fitness
<bestFitness) {
      bestFitness = fitness;
      bestSchedule          =
List.from(ants);
            }
          //
updatePheromoneMatrix();
      sendPort.send(
                "Iterasi:
${iteration  +  1}  ||  Best
```

```
Fitness:    $bestFitness    ||
Fitness: $fitness ");

            if (fitness == 0)
break;
          }
      log.d("Best    Fitness:
$bestFitness");
      bestSchedule!.sort((a,
b)
=>antSlotList.indexOf(a.antS
lot!).compareTo(
      antSlotList.indexOf(b.
antSlot!),
              ));
          final  jadwalList  =
bestSchedule.map((ant) {
      var  pengampuJadwal  =
ant.pengampuJadwal;
          var    antSlot    =
ant.antSlot!;
            var    hari    =
antSlot.hari;
       var     ruangan     =
antSlot.ruangan;
            var startJamIndex
=
antSlotList.indexOf(antSlot)
;
            var     jam     =
<JamModel>[];
            var  jamIndex  =
startJamIndex + jam.length;
      while
(jam.length<pengampuJadwal.m
atakuliah.sks) {
              if  (jamIndex>=
antSlotList.length) {
                //      throw
'Tidak    ada    slot    yang
tersedia';
                break;
              }
              var  antSlot  =
antSlotList[jamIndex];
              //           if
(antSlot.hari   !=   hari   ||
antSlot.ruangan != ruangan) {
              //         throw
'Tidak    ada    slot    yang
tersedia';
              // }
              // cekjika jam
sudahada di list jam
              if
(jam.contains(antSlot.jam)) {
      // lanjut ke slot
berikutnya
                jamIndex++;
      continue;
              }
      jam.add(antSlot.jam);
      jamIndex++;
            }
            return
JadwalModel(
      pengampuJadwal:
pengampuJadwal,
      hari: hari,
              jam: jam,
      ruangan: ruangan,
            );
          }).toList();
      sendPort.send(jadwalLi
st);
        }
      AntSlotModelselectAntS
lot(AntModel ant) {
          final
pengampuJadwal              =
ant.pengampuJadwal;
          if (ant.bentrok ==
0 &&ant.antSlot != null) {
            if
(random.nextBool())    return
ant.antSlot!;
          }
          var
availableAntSlotList        =
<AntSlotModel>[];
          for (var antSlot in
antSlotList) {
            if   (antSlot   ==
ant.antSlot) continue;
            if
(!antSlot.hari.kelas.contain
s(pengampuJadwal.kelasType))
{
              continue;
            }
            //
cekjikakemungkinan        jam
selesaimelebihi           jam
selesaihari
            var indexOfJam =
_jamSet.toList().indexOf(ant
Slot.jam);
            if (indexOfJam +
pengampuJadwal.matakuliah.sk
s> _jamSet.length) {
              continue;
        }
```

```
      availableAntSlotList.add(antSlot);
          }
          if (availableAntSlotList.isEmpty) {
            throw 'Tidak ada slot yang tersedia';
          }
          return availableAntSlotList[random.nextInt(availableAntSlotList.length)];
        }
        // calculate fitness
        double calculateFitness() {
          double fitness = 0;
          // cekbentrok
          for (var i = 0; i<ants.length; i++) {
            var ant = ants[i];
            var pengampuJadwal = ant.pengampuJadwal;
            var startAntSlotIndex = antSlotList.indexOf(ant.antSlot!);
            if (startAntSlotIndex == -1) {
              continue;
            }
            for (var j = i + 1; j <ants.length; j++) {
              var otherAnt = ants[j];
              var otherPengampuJadwal = otherAnt.pengampuJadwal;
              var otherAntSlot = otherAnt.antSlot;
              for (var i = 0; i<pengampuJadwal.matakuliah.sks; i++) {
                var antSlotIndex = startAntSlotIndex + i;
                if (antSlotIndex>= antSlotList.length) {
                  break;
                }
                var antSlot = antSlotList[antSlotIndex];
                if (otherAntSlot == null) {
                  continue;
                }

                if (antSlot == otherAntSlot) {
                  fitness += 1;
                }
                if (antSlot.hari.id == otherAntSlot.hari.id &&

                antSlot.jam.id == otherAntSlot.jam.id) {
                  if (antSlot.ruangan.id == otherAntSlot.ruangan.id &&

                antSlot.jam.id == otherAntSlot.jam.id) {
                    fitness += 1;
                  }
                  if (pengampuJadwal.dosen.id == otherPengampuJadwal.dosen.id &&

                antSlot.jam.id == otherAntSlot.jam.id) {
                    fitness += 1;
                  }
                  if (pengampuJadwal.kelas.id == otherPengampuJadwal.kelas.id &&

                antSlot.jam.id == otherAntSlot.jam.id) {
                    fitness += 1;
                  }
                }
              }
            }
          }
          return fitness;
        }
      }
```

Lampiran – 3 List Program Pengampu_service.dart

```
class PengampuService with ListenableServiceMixin {
  PengampuService() {
    listenToReactiveValues([items]);
  }
  final log = getLogger('PengampuService');
  static const String tableName = 'pengampu';
  final _supabase = Supabase.instance.client;
  final _items = ReactiveList<PengampuModel>();
  /// List of all data
  List<PengampuModel> get items => _items.toSet().toList();

  bool _isSync = false;
  Future<void>syncData() async {
    if (_isSync) return;
    await gets();
    _isSync = true;
  }
  // generate dummy
  Future<void>generateDummy() async {
    final pengampuList = <PengampuModel>[];
    const idProgramStudi = '15deb00c-d7ee-4a96-8108-a7c44e3d6765';
    final listMatakuliah = locator<MatakuliahService>()
        .items
        .where((e) =>e.idProgramStudi == idProgramStudi&& !e.semester.isEven);
    final listDosen = locator<DosenService>().items;
    final listKelas = locator<KelasService>()
        .items
        .where((e) =>e.idProgramStudi == idProgramStudi);
    for (var matakuliah in listMatakuliah) {
      final randomDosen = listDosen[Random().nextInt(listDosen.length)];
      final idPengampu = const Uuid().v4();
      final kelas = <PengampuKelasModel>[];
      final listKelasSemester = listKelas.where((e) =>e.semester == matakuliah.semester).toList();
      if (listKelasSemester.isEmpty) continue;

      for (var kelasSemester in listKelasSemester) {
        for (var i = 0; i<kelasSemester.nama.length; i++) {
          final nama = kelasSemester.nama[i];
          kelas.add(
            PengampuKelasModel.create(
              idKelas: kelasSemester.id,
              idPengampu: idPengampu,
              kelas: nama,
            ),
          );
        }
      }
      final model = PengampuModel(
        id: idPengampu,
        tahunAkademik: '2023 - 2024',
        idMatakuliah: matakuliah.id,
```

```
                    idDosen:
randomDosen.id,
              kelas: kelas,
            );
             pengampuList.add
(model);
          }
          await _supabase
               .from('pengamp
u')
               .insert(pengam
puList.map((e)
=>e.toJson()).toList());
                         await
_supabase.from('pengampu_kel
as').insert(pengampuList
               .expand((eleme
nt) =>element.kelas)
                    .map((e)
=>e.toJson())
              .toList());
        }
        /// Get all data
         Future<List<Pengampu
Model>> gets() async {
          try {
             final response =
await _supabase
                .from(tableN
ame)
                .select(
                           '*,
kelas:pengampu_kelas(*)',
                 )
                 .order('tahu
n_akademik', ascending: true)
                .order('id_m
atakuliah', ascending: true)
                .order('id_d
osen', ascending: true);
             log.d("response:
$response");
                          if
(response.isEmpty) {
              return [];
            }
            final list =
                response.map
((json)
=>PengampuModel.fromJson(jso
n)).toList();
            _items.clear();
             _items.addAll(li
st);
            return list;
          } catch (e) {
              log.e(e);
            }
            return [];
          }
           /// Get data by
semester and tahunakademik
          Future<List<Pengampu
JadwalModel>>getJadwal({
                       required
ProgramStudiModelprogramStud
i,
                       required
SemesterType semester,
             required String
tahunAkademik,
          }) async {
            try {
               log.wtf(programS
tudi);
               log.wtf("semeste
r: ${semester.gets()}");
               log.wtf("tahunAk
ademik: $tahunAkademik");
              // final response
= await _supabase
              //      .from('pe
ngampu_kelas')
              //      .select(
                //          '*,
kelas_jenis:id_kelas(jenis),
pengampu:id_pengampu(id,
tahun_akademik,
matakuliah:id_matakuliah(*),
dosen:id_dosen(*))',
              //      )
               //      .not('pen
gampu', 'is', null)
               //      .not('pen
gampu.matakuliah',      'is',
null)
               //      .in_('pen
gampu.matakuliah.semester',
semester.gets())
               //      .eq('peng
ampu.tahun_akademik',
tahunAkademik)
               //      .eq('peng
ampu.matakuliah.id_program_s
tudi', programStudi.id);
               final response =
await
_supabase.rpc('get_pengampu_
kelas', params: {
                  'program_studi
_id_value': programStudi.id,
```

```
              'semester_values': semester.gets(),
              'tahun_akademik_value': tahunAkademik,
            });
            if (response == null) return [];

            log.d("response: ${response}");
                        if (response.isEmpty) return [];
              final list = <PengampuJadwalModel>[];
            for (final json in response) {
              try {
                            // log.d('json: $json');
                final kelas = PengampuKelasModel.fromJson(json);
                final newJson = json['pengampu'] as Map<String, dynamic>;
                newJson['kelas'] = kelas.toJson();
                newJson['kelas_type'] = json['kelas_jenis']['jenis'] as int;

                        final pengampuJadwal = PengampuJadwalModel.fromJson(newJson);
                list.add(pengampuJadwal);
              } catch (e) {
                log.wtf('json: $json');
                log.e(e);
              }
            }
            return list;
          } catch (e) {
            log.e(e);
          }
          return [];
        }
        Future<List<String>> getTahunAkademik() async {
          try {
            final response = await _supabase.from('distinct_tahun_akademik').select();
            log.d("response: $response");
                        if (response.isEmpty) {
              return [];
            }

              final list = response.map((e) => e['tahun_akademik'] as String).toList();
            return list;
          } catch (e) {
            log.e(e);
          }
          return [];
        }
        /// Save or update data
                Future<void> save(PengampuModel model) async {
          try {
                        await _supabase.from(tableName).upsert(model.toJson());
            await _supabase
                .from('pengampu_kelas')
                .delete()
                .eq('id_pengampu', model.id);
            await _supabase
                .from('pengampu_kelas')
                .upsert(model.kelas.map((kelas) =>kelas.toJson()).toList());
              final index = _items.indexWhere((element) => element.id == model.id);
            if (index >= 0) {
              _items[index] = model;
            } else {
               _items.add(model);
            }
          } catch (e) {
            log.e(e);
                        throw 'Gagalmenyimpan data';
          }
        }
```

```
        /// Delete data
                Future<void>
delete(PengampuModel   model)
async {
          try {
                          await
_supabase.from(tableName).de
lete().eq('id', model.id);
                    _items.remove(mo
del);
          } catch (e) {
            log.e(e);
                              throw
'Gagalmenghapus data';
          }
        }
      }
```

Lampiran – 4 List Program jadwal_service.dart

```
class JadwalService {
  final log = getLogger('JadwalService');
  Stream<dynamic> generate(GenerateJadwalRequest request) async* {
    log.i('starting generate jadwal...');
    try {
      // generate jadwalmenggunakanmetode ant colony optimization
      // 1. ambil data pengampu
      yield 'Mengambil data pengampu...';
      final pengampuJadwalList = await locator<PengampuService>().getJadwal(
        programStudi: request.programStudi,
        semester: request.semester,
        tahunAkademik: request.tahunAkademik,
      );
      yield "Total pengampu: ${pengampuJadwalList.length}";
      if (pengampuJadwalList.isEmpty) {
        throw 'Tidak ada data pengampu yang sesuaidengankriteria';
      }
      // 2. ambil data jam
      yield 'Mengambil data jam...';
      final jamList = await locator<JamService>().gets(getAktifOnly: true);
      yield "Total jam: ${jamList.length}";
      // 3. ambil data hari
      yield 'Mengambil data hari...';
      final hariList = await locator<HariService>().gets();
      yield "Total hari: ${hariList.length}";
      // 4. ambil data ruang
      yield 'Mengambil data ruangan...';
      final ruanganList = await locator<RuanganService>().gets();
      yield "Total ruangan: ${ruanganList.length}";
      // 5. inisialisasi data slot
      yield 'Menginisialisasi data slot...';
      final antSlotList = AntSlotModel.generate(
        hariList,
        jamList,
        ruanganList,
      );
      yield "Total slot: ${antSlotList.length}";
      // 6. inisialisasisemut
      final antColony = AntColonyService(
        pengampuJadwalList,
        antSlotList,
        maxIterations: request.iterasi,
      );
      // 7. mulai generate jadwal
      request.isolate = await Isolate.spawn(
        antColony.runACO,
        request.receivePort.sendPort,
      );
      // 8. kirimhasil generate jadwalke main isolate
      yield* request.receivePort;
```

```
                            //
log.d("bestJadwal:
\n$bestJadwal");
          } on String catch
(_) {
            rethrow;
          } catch (e) {
            log.e(e);
               throw  'Gagal
generate jadwal';
          }
        }
      }
      class
GenerateJadwalRequest {
        Isolate? isolate;
                         final
ReceivePortreceivePort;
                         final
ProgramStudiModelprogramStud
i;
           final  SemesterType
semester;
             final     String
tahunAkademik;
        final int iterasi;
         GenerateJadwalReques
t({
                         required
this.isolate,
                         required
this.receivePort,
                         required
this.programStudi,
                         required
this.semester,
                         required
this.tahunAkademik,
                         required
this.iterasi,
        });
      }
```

Lampiran – 5 List Program Periode_semester_service.dart

```
      class
PeriodeSemesterService   with
ListenableServiceMixin {
         PeriodeSemesterServi
ce() {
           listenToReactiveVa
lues([_ganjil, _genap]);
         }
            final    log    =
getLogger('PeriodeSemesterSe
rvice');
         static  const  String
tableName                    =
'periode_semester';
          final  _supabase  =
Supabase.instance.client;
                          final
ReactiveValue<PeriodeSemeste
rModel?>       _ganjil       =
ReactiveValue<PeriodeSemeste
rModel?>(null);
                          final
ReactiveValue<PeriodeSemeste
rModel?>       _genap        =
ReactiveValue<PeriodeSemeste
rModel?>(null);
        PeriodeSemesterModel?
get ganjil => _ganjil.value;
        PeriodeSemesterModel?
get genap => _genap.value;
        bool _isSync = false;
         Future<void>syncData
() async {
               if    (_isSync)
return;
           await fetchData();
           _isSync = true;
         }
         Future<void>fetchDat
a() async {
           try {
              final response =
await
_supabase.from(tableName).se
lect();
             log.d("response:
$response");
                           if
(response.isEmpty) {
              _ganjil.value =
null;
                _genap.value =
null;
               return;
             }
              for (var item in
response) {
                final model =
PeriodeSemesterModel.fromJso
n(item);
               if (model.type
==
PeriodeSemesterType.ganjil) {
                 _ganjil.value
= model;
                 }  else  if
(model.type                  ==
PeriodeSemesterType.genap) {
                  _genap.value
= model;
               }
             }
           } catch (e) {
             log.e(e);
           }
         }
                  Future<void>
save(PeriodeSemesterModel
model) async {
           try {
                           await
_supabase.from(tableName).up
sert(model.toJson());
             if (model.type ==
PeriodeSemesterType.ganjil) {
               _ganjil.value =
model;
                 }   else   if
(model.type                  ==
PeriodeSemesterType.genap) {
               _genap.value =
model;
             }
           } catch (e) {
             log.e(e);
                           throw
'Gagalmenyimpan data';
           }
         }
       }
```

Lampiran – 6 List Program matakuliah_service.dart

```
class MatakuliahService with ListenableServiceMixin {
  MatakuliahService() {
    listenToReactiveValues([items]);
  }

  final log = getLogger('MatakuliahService');

  static const String tableName = 'matakuliah';
  final _supabase = Supabase.instance.client;

  final _items = ReactiveList<MatakuliahModel>();

  /// List of all data
  List<MatakuliahModel> get items => _items.toSet().toList();

  bool _isSync = false;

  Future<void>syncData() async {
    if (_isSync) return;

    await gets();

    _isSync = true;
  }

  /// Get data by id
  MatakuliahModel? getById(String id) {
    return _items.firstWhereOrNull((element) => element.id == id);
  }

  /// Get all data
  Future<List<MatakuliahModel>> gets() async {
    try {
      final response = await _supabase
          .from(tableName)
          .select()
          .order('semester', ascending: true)
          .order('id_program_studi', ascending: true)
          .order('nama', ascending: true);

      log.d("response: $response");

      if (response.isEmpty) {
        return [];
      }

      final list = response.map((e) =>MatakuliahModel.fromJson(e)).toList();

      _items.clear();

      _items.addAll(list);

      return list;
    } catch (e) {
      log.e(e);
    }

    return [];
  }

  /// Save or update data
  Future<void> save(MatakuliahModel model) async {
    try {
      await _supabase.from(tableName).upsert(model.toJson());

      final index = _items.indexWhere((element) => element.id == model.id);

      if (index >= 0) {
        _items[index] = model;
      } else {
        _items.add(model);
```

```
      }
    } catch (e) {
      log.e(e);
      throw 'Gagalmenyimpan data';
    }
  }

  /// Delete data
  Future<void> delete(MatakuliahModel model) async {
    try {
      await _supabase.from(tableName).delete().eq('id', model.id);

      _items.remove(model);
    } catch (e) {
      log.e(e);
      throw 'Gagalmenghapus data';
    }
  }
}
```

Lampiran – 7 List Program fakultas_service.dart

```
class FakultasService with ListenableServiceMixin {
  FakultasService() {
    listenToReactiveValues([items]);
  }

  final log = getLogger('FakultasService');

  static const String tableName = 'fakultas';
  final _supabase = Supabase.instance.client;

  final _items = ReactiveList<FakultasModel>();

  /// List of all data
  List<FakultasModel> get items => _items.toSet().toList();

  bool _isSync = false;

  Future<void>syncData() async {
    if (_isSync) return;

    await gets();

    _isSync = true;
  }

  /// Get all data
  Future<List<FakultasModel>> gets() async {
    try {
      final response = await _supabase.from(tableName).select();

      if (response.isEmpty) {
        return [];
      }

      final list = response.map((e) =>FakultasModel.fromJson(e)).toList();

      _items.clear();

      _items.addAll(list);

      return list;
    } catch (e) {
      log.e(e);
    }

    return [];
  }

  /// Save or update data
  Future<void> save(FakultasModelfakultas) async {
    try {
      await _supabase.from(tableName).upsert(fakultas.toJson());

      final index = _items.indexWhere((element) => element.id == fakultas.id);

      if (index >= 0) {
        _items[index] = fakultas;
      } else {
        _items.add(fakultas);
      }
    } catch (e) {
      log.e(e);
      throw 'Gagalmenyimpan data';
    }
  }

  /// Delete data
  Future<void> delete(FakultasModelfakultas) async {
    try {
      await _supabase.from(tableName).delete().eq('id', fakultas.id);
```

```
            _items.remove(fa                                    throw
kultas);                                  'Gagalmenghapus data';
          } catch (e) {                         }
            log.e(e);                         }
                                              }
```