2025

UJI KOMPETENSI KEAHLIAN PAKET II

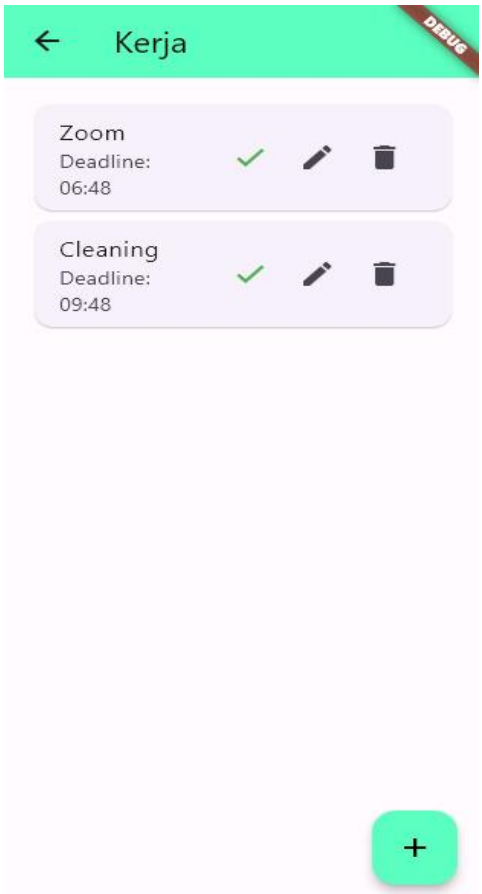
Pengembangan Perangkat Lunak dan Gim

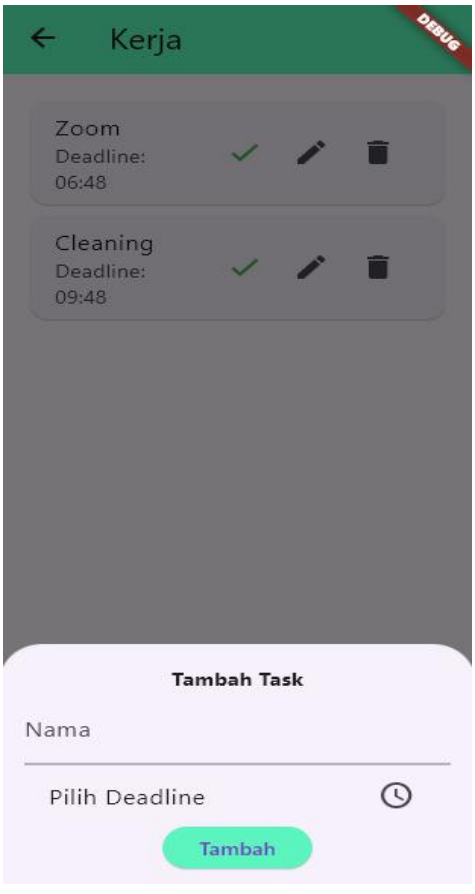
Abdi Rama Bagaskara
0076580672

Wireframe


Todos Esemka
DEBUG
Kerja
Workout
+


Todos Esemka
DEBUG
Kerja
Workout
Tambah
Nama tugas
Batal
Simpan
+


Kerja
DEBUG
Zoom
Deadline:
06:48
Cleaning
Deadline:
09:48
+


Kerja
DEBUG
Zoom
Deadline:
06:48
Cleaning
Deadline:
09:48
Tambah Task
Nama
Pilih Deadline
Tambah

1. Backend

A. Migration Database

```php
public function up(): void
{
    // ini adalah migration untuk membuat table tasks dengan kolom id, dan foreign key list
    Schema::create('tasks', function (Blueprint $table) {
        $table->id();
        $table->foreignId('list_id')->constrained('lists')->onDelete('cascade');
        $table->string('name');
        $table->time('deadline');
        $table->enum('status', ['in progress', 'completed'])->default('in progress');
        $table->timestamps();
    });
}
```

```php
return new class extends Migration
{
    Tabnine | Edit | Test | Explain | Document
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        // ini migration untuk membuat table lists dengan kolom name, id, dan timestamps(created_at, u
        Schema::create('lists', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }
```

B. Model

```php
class Tasks extends Model
{
    use HasFactory;
    protected $table = 'tasks';
    protected $fillable = ['name', 'deadline', 'status', 'list_id'];
    public $timestamps = false;
}
```

```php
class Lists extends Model
{
    use HasFactory;
    protected $table = 'lists';
    protected $fillable = ['name'];
    public $timestamps = false;
}
```

C. Controller

```php
public function index()
{
    return Lists::all();
}

Tabnine | Edit | Test | Explain | Document
public function create(Request $req)
{
    $lists = $req->validate([
        'name' => 'required|string',
    ]);

    Lists::create($lists);
    return response()->json(['message' => 'data created', 'list' => $lists]);
}


Tabnine | Edit | Test | Explain | Document
public function update(Request $req, $id)
{
    $lists = Lists::find($id);
    if (!$lists){
        return response()->json(['message' => 'data not found']);
    }

    $data = $req->validate([
        'name' => 'required|string',
    ]);

    $lists->update($data);
    return response()->json(['message' => 'data updated', 'list' => $lists]);
}


Tabnine | Edit | Test | Explain | Document
public function delete($id)
{
    $lists = Lists::find($id);
    if (!$lists){
        return response()->json(['message' => 'data not found']);
    }

    $lists->delete();
    return response()->json(['message' => 'data deleted']);
}
```

```php
{
    Tabnine | Edit | Test | Explain | Document
    public function index()
    {
        return response()->json(Tasks::all());
    }


    Tabnine | Edit | Test | Explain | Document
    public function create(Request $req)
    {
        $tasks = $req->validate([
            'name' => 'required|string',
            'deadline' => 'required|date_format:H:i',
            'status' => 'required|in:in progress,completed',
            'list_id' => 'exists:lists,id'
        ]);

        Tasks::create($tasks);
        return response()->json(['message' => 'data created', 'tasks' => $tasks]);
    }

    Tabnine | Edit | Test | Explain | Document
    public function update(Request $req, $id)
    {
        $tasks = Tasks::find($id);
        if (!$tasks){
            return response()->json(['message' => 'data not found']);
        }

        $data = $req->validate([
            'name' => 'sometimes|string',
            'deadline' => 'sometimes|date_format:H:i',
            'status' => 'sometimes|in:in progress,completed',
            'list_id' => 'sometimes|exists:lists,id'
        ]);

        $tasks->update($data);
        return response()->json(['message' => 'data updated', 'lists' => $data]);
    }


    Tabnine | Edit | Test | Explain | Document
    public function delete($id)
    {
        $tasks = Tasks::find($id);
        if (!$tasks){
            return response()->json(['message' => 'data not found']);
        }

        $tasks->delete();
        return response()->json(['message' => 'data deleted']);
    }
}
```

2. Frontend

A. List Page

```dart
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'task.dart';

class ListPage extends StatefulWidget {
  const ListPage({super.key});
  @override
  State<ListPage> createState() => _ListPageState();
}

class _ListPageState extends State<ListPage> {
  List<dynamic> lists = [];
  final baseUrl = 'http://127.0.0.1:8001/api/lists';
  final color = const Color.fromARGB(255, 91, 255, 192);

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

  Future<void> fetchLists() async {
    try {
      final res = await http.get(Uri.parse('$baseUrl/get'));
      if (res.statusCode == 200) {
        setState(() => lists = json.decode(res.body));
      }
    } catch (_) {}
  }

  Future<void> submitList(String name, {int? id}) async {
    if (name.isEmpty) return;
    final url =
        Uri.parse(id == null ? '$baseUrl/create' : '$baseUrl/update/$id');
    final res = await http.post(url,
        headers: {'Content-Type': 'application/json'},
        body: json.encode({'name': name}));
    if ([200, 201].contains(res.statusCode)) fetchLists();
  }

  Future<void> deleteList(int id) async {
    final res = await http.delete(Uri.parse('$baseUrl/delete/$id'));
    if (res.statusCode == 200) fetchLists();
  }

  void showInput({Map<String, dynamic>? list}) {
    final ctrl = TextEditingController(text: list?['name'] ?? '');
    showDialog(
      context: context,
      builder: (ctx) => AlertDialog(
        title: Text(list != null ? 'Edit' : 'Tambah'),
        content: TextField(
          controller: ctrl,
          autofocus: true,
          decoration: const InputDecoration(hintText: 'Nama tugas'),
        ),
        actions: [
          TextButton(
              onPressed: () => Navigator.pop(ctx), child: const Text('Batal')),
          ElevatedButton(
            style: ElevatedButton.styleFrom(
                backgroundColor: color, foregroundColor: Colors.black),
            onPressed: () {
              final name = ctrl.text.trim();
              if (name.isNotEmpty) {
                submitList(name, id: list?['id']);
                Navigator.pop(ctx);
              }
            },
            child: const Text('Simpan'),
          ),
        ],
      ),
    );
  }
```

```dart
Widget buildListTile(Map<String, dynamic> list) {
   return Card(
     shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),
     child: ListTile(
       title: Text(list['name'], style: const TextStyle(color: Colors.black)),
       onTap: () => Navigator.push(
         context,
         MaterialPageRoute(builder: (_) => TaskPage(list: list)),
       ).then((_) => fetchLists()),
       trailing: Wrap(
         spacing: 4,
         children: [
           IconButton(
               icon: const Icon(Icons.edit, color: Colors.black),
               onPressed: () => showInput(list: list)),
           IconButton(
               icon: const Icon(Icons.delete, color: Colors.black),
               onPressed: () => deleteList(list['id'])),
         ],
       ),
     ),
   );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: color,
        centerTitle: true,
        title:
            const Text('Todos Esemka', style: TextStyle(color: Colors.black)),
      ),
      body: lists.isEmpty
          ? const Center(child: Text('Belum ada tugas'))
          : ListView.builder(
              padding: const EdgeInsets.all(16),
              itemCount: lists.length,
              itemBuilder: (_, i) => buildListTile(lists[i])),
      floatingActionButton: FloatingActionButton(
        backgroundColor: color,
        onPressed: () => showInput(),
        child: const Icon(Icons.add, color: Colors.black),
      ),
    );
  }
}
```

B. Task Page

```dart
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

class TaskPage extends StatefulWidget {
  final Map<String, dynamic> list;
  const TaskPage({super.key, required this.list});

  @override
  State<TaskPage> createState() => _TaskPageState();
}

class _TaskPageState extends State<TaskPage> {
  List tasks = [];
  final baseUrl = 'http://127.0.0.1:8001/api/tasks';
  final color = const Color.fromARGB(255, 91, 255, 192);
  bool loading = true;

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

  Future<void> fetch() async {
    try {
      final res = await http.get(Uri.parse('$baseUrl/get'));
      if (res.statusCode == 200) {
        final data = json.decode(res.body);
        tasks = data.where((t) => t['list_id'] == widget.list['id']).toList();
      }
    } catch (_) {}
    setState(() => loading = false);
  }

  Future<void> submit(String name, TimeOfDay time,
      {int? id, String status = 'in progress'}) async {
    final body = {
      'name': name,
      'deadline':
          '${time.hour.toString().padLeft(2, '0')}:${time.minute.toString().padLeft(2, '0')}',
      'status': status,
      if (id == null) 'list_id': widget.list['id']
    };
    await http.post(
      Uri.parse(id == null ? '$baseUrl/create' : '$baseUrl/update/$id'),
      headers: {'Content-Type': 'application/json'},
      body: json.encode(body),
    );
    fetch();
    // ignore: use_build_context_synchronously
    Navigator.pop(context);
  }

  Future<void> updateStatus(int id, String status) async {
    await http.post(Uri.parse('$baseUrl/update/$id'),
        headers: {'Content-Type': 'application/json'},
        body: json.encode({'status': status}));
    fetch();
  }

  Future<void> delete(int id) async {
    await http.delete(Uri.parse('$baseUrl/delete/$id'));
    fetch();
  }

  void showForm({Map? task}) {
    final nameCtrl = TextEditingController(text: task?['name'] ?? '');
    TimeOfDay? time = task != null
        ? TimeOfDay(
            hour: int.parse(task['deadline'].split(':')[0]),
            minute: int.parse(task['deadline'].split(':')[1]))
        : null;
    String status = task?['status'] ?? 'in progress';
```

```dart
showModalBottomSheet(
        context: context,
        isScrollControlled: true,
        builder: (_) => Padding(
              padding: EdgeInsets.fromLTRB(
                  16, 16, 16, MediaQuery.of(context).viewInsets.bottom + 16),
              child: Column(mainAxisSize: MainAxisSize.min, children: [
                Text(task != null ? 'Edit Task' : 'Tambah Task',
                    style: const TextStyle(fontWeight: FontWeight.bold)),
                TextField(
                    controller: nameCtrl,
                    decoration: const InputDecoration(labelText: 'Nama')),
                ListTile(
                  title: Text(time != null
                      ? '${time?.hour.toString().padLeft(2, '0')}:${time?.minute.toString().padLeft(2, '0')}'
                      : 'Pilih Deadline'),
                  trailing: const Icon(Icons.access_time),
                  onTap: () async {
                    final t = await showTimePicker(
                        context: context, initialTime: time ?? TimeOfDay.now());
                    if (t != null) setState(() => time = t);
                  },
                ),
                if (task != null)
                  DropdownButtonFormField(
                    value: status,
                    items: ['in progress', 'completed']
                        .map((s) => DropdownMenuItem(value: s, child: Text(s)))
                        .toList(),
                    onChanged: (v) => status = v!,
                  ),
                ElevatedButton(
                    style: ElevatedButton.styleFrom(backgroundColor: color),
                    onPressed: () {
                      if (nameCtrl.text.isEmpty || time == null) return;
                      submit(nameCtrl.text, time!,
                          id: task?['id'], status: status);
                    },
                    child: Text(task != null ? 'Perbarui' : 'Tambah'))
              ]),
            ));
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            backgroundColor: color,
            title: Text(widget.list['name'],
                style: const TextStyle(color: Colors.black))),
        body: loading
            ? const Center(child: CircularProgressIndicator())
            : tasks.isEmpty
                ? const Center(child: Text('Belum ada task'))
                : ListView.builder(
                    padding: const EdgeInsets.all(16),
                    itemCount: tasks.length,
                    itemBuilder: (_, i) {
                      final t = tasks[i], done = t['status'] == 'completed';
                      return Card(
                          child: ListTile(
                              title: Text(t['name'],
                                  style: TextStyle(
                                      decoration: done
                                          ? TextDecoration.lineThrough
                                          : null)),
                              subtitle: Text(
                                  'Deadline: ${t['deadline'].substring(0, 5)}'),
                              trailing: Wrap(spacing: 4, children: [
                                IconButton(
                                    icon: Icon(
                                        done ? Icons.refresh : Icons.check,
                                        color: done
                                            ? Colors.orange
                                            : Colors.green),
                                    onPressed: () => updateStatus(t['id'],
                                        done ? 'in progress' : 'completed')),
                                IconButton(
                                    icon: const Icon(Icons.edit),
                                    onPressed: () => showForm(task: t)),
                                IconButton(
                                    icon: const Icon(Icons.delete),
                                    onPressed: () => delete(t['id']))
                              ])));
                    }),
        floatingActionButton: FloatingActionButton(
            backgroundColor: color,
            onPressed: showForm,
            child: const Icon(Icons.add, color: Colors.black)));
  }
}
```

3. Cara Penggunaan

1. Saat dijalankan aplikasi to-do list ini akan menampilkan data list tugas anda di List Page, jika anda belum punya list tugas anda bisa membuat list nya dengan button tambah dibawah. Selain itu, anda juga dapat memperbarui, dan menghapus tugas sesuai keinginan. Selanjutnya jika anda meng-klik salah satu list, maka anda akan diarahkan ke Task Page, dan akan ditunjukkan Task anda sesuai list yang diklik.

2. Anda masuk ke Task Page sekarang, disini anda bisa menambah tugas sesuai dengan list yang ada, contoh; Kerja(Meeting, Survei, dan lain lain). Selain itu anda bisa mengubah nama, status, dan deadline tugas anda. Dan juga sudah dilengkapi fitur hapus tugas.