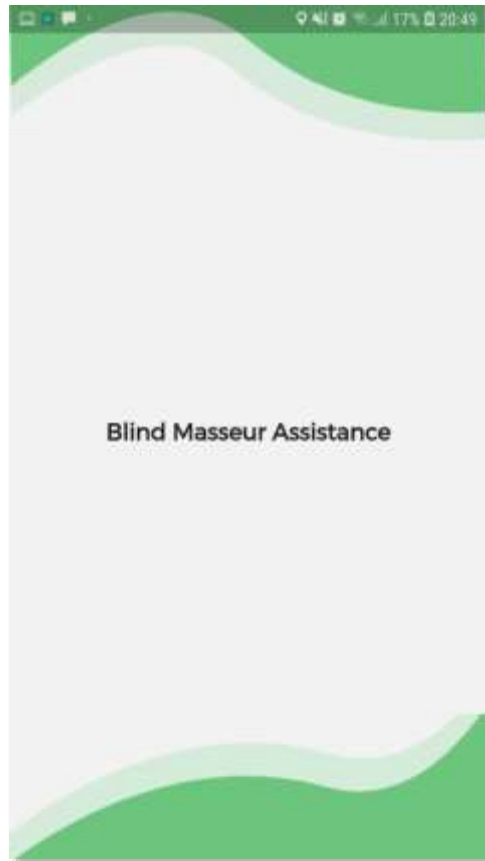


DOKUMENTASI SOURCE CODE

APLIKASI BLIND MASSEUR ASSISTANCE UNTUK KEDARURATAN SEBAGAI UPAYA PENCEGAHAN KEKERASAN PADA PEMIJAT TUNA NETRA



PENCIPTA:

**Nur Setiawati Dewi, Ph.D., Sp.Kom
Heru Agus Santoso, Ph.D
Muhammad Syaifur Rohman, M.Cs.**

**UNIVERSITAS DIPONEGORO
2021**

Dokumentasi Source Code

APLIKASI BLIND MASSEUR ASSISTANCE UNTUK KEDARURATAN SEBAGAI UPAYA PENCEGAHAN KEKERASAN PADA PEMIJAT TUNA NETRA

Blind Masseur Register Source Code

Blind Masseur Register Source Code	
Android Proses	Api Server
<pre>onOptionsItemSelected() { register() } register() { username = findViewById<TextView>(R.id.username_text), password = findViewById<TextView>(R.id.password_text), nama = findViewById<TextView>(R.id.nama_text), jk = findViewById<TextView>(R.id.jk_text), alamat = findViewById<TextView>(R.id.alamat_text), role = findViewById<TextView>(R.id.role_text) btnRegister.setOnClickListener() { validate() register() } } private fun validate() { if (username.isEmpty() password.isEmpty() nama.isEmpty() jk.isEmpty() alamat.isEmpty() role.isEmpty()) { Toast.makeText(this, "Silahkan isi semua data", Toast.LENGTH_SHORT).show() return } } private fun register() { val userData = hashMapOf<String, String>() userData["username"] = username.text.toString() userData["password"] = password.text.toString() userData["nama"] = nama.text.toString() userData["jk"] = jk.text.toString() userData["alamat"] = alamat.text.toString() userData["role"] = role.text.toString() apiClient.post("register", userData) } private fun apiClient(url: String, body: Map<String, String>): Response { return RetrofitClient.getInstance().create(ApiClient::class.java).post(url, body) }</pre>	<pre>public function register() { \$username = \$_POST['username']; \$password = \$_POST['password']; if (isset(\$username) && isset(\$password)) { \$checkUsername = count(\$this->crud->getWhere('id', 'users', ['username' => \$username])); if (\$checkUsername == 0) { \$dataUsers = [// rand(1000, 9999) 'username' => \$username, 'password' => password_hash(\$password, PASSWORD_DEFAULT), 'nama' => \$_POST['nama'], 'jk' => \$_POST['jk'], 'alamat' => \$_POST['alamat'], 'idRole' => \$_POST['idRole']]; \$this->crud->insert('users', \$dataUsers); \$idUsers = \$this->db->insert_id(); \$dataToken = ['idUsers' => \$idUsers, 'token' => strtolower(bin2hex(randam_bytes(32)))]; \$this->crud->insert('users_token', \$dataToken); \$code = count(\$this->crud->getWhere('id', 'users', ['id' => \$idUsers, 'idRole' => '1'])); if (\$code == 1) { \$dataStatus = ['idUsers' => \$idUsers, 'status' => '0']; \$this->crud->insert('status_user', \$dataStatus); \$response['code'] = '201'; \$response['message'] = 'User berhasil didaftarkan'; \$response['data'] = \$dataToken; } else { \$response['code'] = '401'; \$response['message'] = 'Username sudah ada'; } } else { \$response['code'] = '400'; \$response['message'] = 'Isian tidak sesuai'; } } \$this->output ->set_content_type('application/json') ->set_output(json_encode(\$response, JSON_PRETTY_PRINT)); }</pre>
	<pre>public function getWhere(\$select, \$table, \$where) { { \$this->db->select(\$select); return \$this->db->get_where(\$table, \$where); } }</pre>
	<pre>public function insert(\$table, \$data) { { return \$this->db->insert(\$table, \$data); } }</pre>

Ketika button ditap, maka app akan mengakses API dan mengirim data dari form. Setelah mendapatkan callback, jika berhasil maka tampilan kembali ke halaman login, jika tidak tampilan tetap di halaman register dan menampilkan error pada toast.

API Server menerima apa yang di inputkan oleh pengguna melalui aplikasi android lalu akan ada pencocokan yaitu apakah username dan password sudah terisi dan apakah username sudah digunakan yang lain.

Jika kriteria sudah memenuhi maka password akan dienkripsi dengan hash dan inputkan ke dalam data users. Selain itu dilakukan pembuatan kode pengguna untuk BM sehingga BMA bisa mendaftarkan melalui kode tersebut.

Yang terakhir status pengguna akan dijadikan tidak sedang melapor.

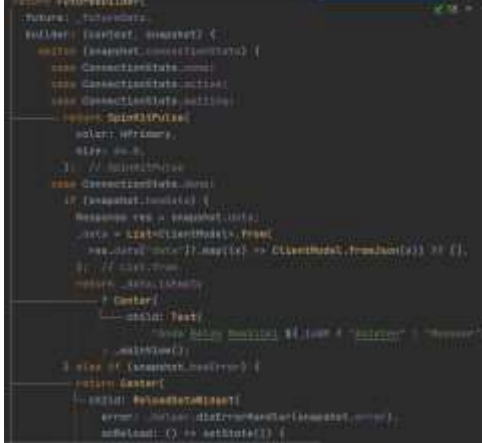
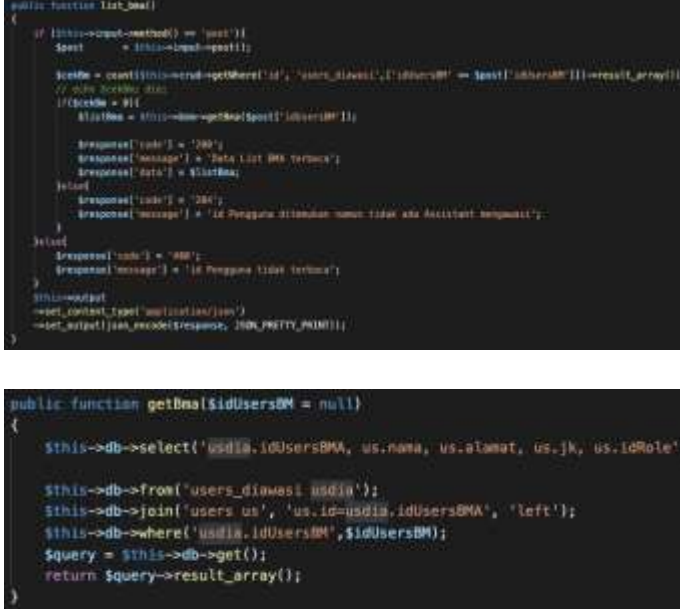
Penggunaan Model getWhere untuk mendapatkan data khusus dari tabel tertentu dan Model insert untuk memasukkan data ke tabel tertentu.

Blind Masseur Login Source Code

Blind Masseur Login Source Code	
Android Proses	Api Server
<pre> await _request.auth .login(username: _username.text, password: _password.text, token: _model.token,) .then((res) async { setState(() { _isLoading = false; }); _helper.showToast("Berhasil Login!"); _model.user = UserModel.fromJson(res.data["data"]); await \$postData().saveToSP(kDUser, SharedDataType.string, jsonEncode(_model.user.toJson()),); await Future.delayed(Duration(seconds: 1)); _helper.navigateToPage(context, page: MainPage()); }).catchError((e) { _helper.showToast("Gagal Login!"); setState(() { _msg = "Gagal Login!"; }); }); </pre> <pre> Future<dio.Response> login({ @required String username, @required String password, @required String token, }) { return _repo.auth.login(data: { "username": username, "password": password, "token": token, }); } </pre>	<pre> public function login() { if (\$this->input->method() == 'post') { // cek // input // validasi \$post = \$this->input->post(); // Check user \$user = \$this->crud->getWhere('users', ['username' => \$post['username']]); if (\$user == null) { \$response['code'] = '200'; \$response['message'] = 'User Tidak Ditemukan'; } else { if (\$password_verify(\$post['password'], \$user['password'])) { \$response['code'] = '200'; \$response['message'] = 'Password Tidak Sesuai!'; } else { \$response['code'] = '200'; \$response['message'] = 'Data Ditemukan'; \$userKode = \$this->crud->getWhere('users', ['users_id' => \$user['id']]); \$this->crud->update('users', ['token' => \$post['token'], 'id' => \$user['id']]); \$userToken = \$this->crud->getWhere('users', ['users_id' => \$user['id']]); \$dataUser = ['id' => \$user['id'], 'name' => \$user['name'], 'ps' => \$user['ps'], 'alamat' => \$user['alamat'], 'knp' => \$userToken['knp'], 'token' => \$userToken['token'], 'token_id' => \$user['id'],]; \$response['data'] = \$dataUser; } } \$response['code'] = '400'; \$response['message'] = 'Masukkan Username dan Password!'; } \$this->redirect(->set_content_type('application/json') ->set_status(json_encode(\$response, JSON_PRETTY_PRINT)); } public function getWhere(\$select, \$table, \$where) { \$this->db->select(\$select); return \$this->db->get_where(\$table, \$where); } public function update(\$table, \$data, \$where) { return \$this->db->update(\$table, \$data, \$where); } </pre>
<p>Ketika Button Masuk ditap, app mengakses API dan mengirim data dari form.</p> <p>Token dari smartphone didapatkan dari library firebase yang otomatis tergenerate.</p> <p>Setelah mendapatkan callback, jika berhasil maka app akan menyimpan data pengguna pada penyimpanan lokal dan tampilan berpindah ke halaman utama, jika tidak tampilan tetap di halaman login dan menampilkan error pada toast.</p>	<p>API Server menerima data username dan password untuk dicocokkan apakah memang sesuai dengan data yang di inputkan oleh pengguna android dengan tabel users.</p> <p>Jika sesuai maka akan memberikan informasi pengguna kepada android dan menerima token smartphone yang akan diupdate dikaitkan dengan data user.</p> <p>Penggunaan Model getWhere untuk mendapatkan data khusus dari tabel tertentu dan Model update untuk memperbarui data ke tabel tertentu.</p>

	<p>Yang terakhir dilakukan update pada status BM bahwa saat ini BM tersebut sedang meminta bantuan.</p> <p>Penggunaan Model <code>getWhere</code> untuk mendapatkan data khusus dari tabel tertentu, Model <code>update</code> untuk memperbarui data ke tabel tertentu dan Model <code>insert</code> untuk memasukkan data ke tabel tertentu. Dibuat Model baru menyesuaikan dengan permintaan seluruh token BMA yang terkait dengan BM yaitu Model <code>getTokenBma</code>. Terakhir pembuatan controller untuk mengirimkan notifikasi dengan banyak target dari bebera token user</p>
--	---

Blind Masseur List Assistant Source Code

Blind Masseur List Assistant Source Code	
Android Proses	Api Server
 <pre> Future<io.Response> dataAssist() { return _repo.massage().dataAssist(data: { "idUsersBM": SingletonModel.shared.user.id, }); } </pre>	 <pre> public function list_bma() { if (\$this->input->method() == 'post') \$post = \$this->input->post(); \$code = \$this->crud->getWhere('id', 'users_bma', ['idUsersBM' => \$post['idUsersBM']])->result_array(); // echo json_encode(\$data); // \$code = 200; \$dataBma = \$this->crud->getBma(\$post['idUsersBM']); \$response['code'] = '200'; \$response['message'] = 'Data List BMA terbacak'; \$response['data'] = \$dataBma; \$code = 200; \$response['code'] = '200'; \$response['message'] = 'Id Pengguna ditemukan namun tidak ada Asistent membantu'; } if (\$this->input->method() == 'get') \$response['code'] = '400'; \$response['message'] = 'Id Pengguna tidak terbacak'; } \$this->output => set_content_type('application/json') => set_output_header('code:' . \$code, 'X-HTTP-STATUS-1'); } public function getBma(\$idUsersBM = null) { \$this->db->select('usdia.idUsersBMA, us.nama, us.alamat, us.jk, us.idRole'); \$this->db->from('users_diawasi usdia'); \$this->db->join('users us', 'us.id-usdia.idUsersBMA', 'left'); \$this->db->where('usdia.idUsersBM', \$idUsersBM); \$query = \$this->db->get(); return \$query->result_array(); } </pre>
<p>Ketika halaman assist termuat, app akan mengakses API untuk mendapatkan data BMA dari BM berdasarkan id user yang tersimpan. Setelah mendapatkan callback, jika berhasil dan memiliki data maka akan menampilkan list dari BMA, jika berhasil tapi tidak ada data maka akan ditampilkan teks "Anda belum memiliki BMA", jika error maka akan menampilkan pesan dari error & button untuk mengakses ulang API</p>	<p>Api server mendapatkan id BM lalu dicari seluruh data BMA yang terkait dengan id BM tersebut</p> <p>Penggunaan Model <code>getWhere</code> untuk mendapatkan data khusus dari tabel tertentu. Dibuat Model baru menyesuaikan dengan permintaan seluruh data BMA yang terkait dengan id BM yaitu <code>getBma</code></p>

Blind Masseur Detail Assistant Source Code

Blind Masseur Detail Assistant Source Code	
Android Proses	Api Server
<pre> class ClientDetailPage extends StatefulWidget { final ClientModel data; ClientDetailPage({required this.data}); @override _ClientDetailPageState createState() => _ClientDetailPageState(); } </pre>	<pre> public function list_bma() { if (\$this->input-method() == 'post') \$post = \$this->input-post(); \$cekBm = count(\$this->crud->getWhere('id', 'users_diawasi', ['idUsersBM' => \$post['idUsersBM']])->result_array()); // jika \$cekBm > 0 if (\$cekBm > 0) \$listBma = \$this->crud->getBma(\$post['idUsersBM']); \$response['code'] = '200'; \$response['message'] = 'Data List BMA Terbiasa'; \$response['data'] = \$listBma; } else \$response['code'] = '200'; \$response['message'] = 'Id Pengguna ditamun namun tidak ada Asisitant terbiasa'; } } \$response['code'] = '200'; \$response['message'] = 'Id Pengguna tidak terbiasa'; } \$this->output =>set_content_type('application/json') =>set_output(json_encode(\$response, JSON_PRETTY_PRINT)); </pre>
<p>Ketika memanggil halaman detail BMA, halaman sebelumnya wajib mengirimkan data dari BMA yang didapatkan dari API List BMA, lalu menampilkannya pada halaman detail BMA</p>	<pre> public function getBma(\$idUsersBM = null) { \$this->db->select('usdia.idUsersBMA, us.nama, us.alamat, us.jk, us.idRole'); \$this->db->from('users_diawasi usdia'); \$this->db->join('users us', 'us.id=usdia.idUsersBMA', 'left'); \$this->db->where('usdia.idUsersBM', \$idUsersBM); \$query = \$this->db->get(); return \$query->result_array(); } </pre> <p>Menggunakan API Server yang sama dengan API Server List Assistent untuk memberikan seluruh data BMA yang terkait dengan id BM tersebut.</p> <p>Penggunaan Model getWhere untuk mendapatkan data khusus dari tabel tertentu. Dibuat Model baru menyesuaikan dengan permintaan seluruh data BMA yang terkait dengan id BM yaitu getBma</p>

Blind Masseur Profile Source Code

Blind Masseur Profile Source Code	
Android Proses	Api Server
<pre> class _ProfilePageState extends State<ProfilePage> { SingletonModel _model; @override void initState() { super.initState(); _model = SingletonModel.withContext(context); } @override Widget build(BuildContext context) { return Scaffold(body: _buildBody(),); } void _onLogout() async { _model.needsHelp = false; _model.user = null; await SPData().resetSPC(); Helper().navigatePage(context, page: LoginPage()); } </pre>	<pre> public function login() { if (\$this->input-method() == 'post') { // user // login() // request \$post = \$this->input-post(); // check email \$users = \$this->crud->getWhere('u', 'users', ['username' => \$post['username']]); if (\$users == null) { \$response['code'] = '204'; \$response['message'] = 'User Tidak Ditemukan'; } else { // (password_verify(\$post['password'], \$users['password'])) { \$response['code'] = '200'; \$response['message'] = 'Password Tidak Sesuai'; } else { \$response['code'] = '200'; \$response['message'] = 'Data Ditemukan'; \$usersId = \$this->crud->getWhere('users', 'users', ['idUsers' => \$users['id']]); \$usersToken = \$this->crud->getWhere('tokens', 'tokens', ['id' => \$users['id']]); \$dataUsers = ['id' => \$users['id'], 'name' => \$users['name'], 'ps' => \$users['ps'], 'email' => \$users['email'], 'token' => \$usersToken['token'], 'token' => \$usersToken['token'], 'profile' => \$users['profile']]; \$response['data'] = \$dataUsers; } } else { \$response['code'] = '401'; \$response['message'] = 'Masukkan username dan Password'; } } \$this->output(->set_content_type('application/json'), ->set_output(json_encode(\$response, JSON_PRETTY_PRINT)); } </pre>
	<pre> public function getWhere(\$select, \$table, \$where) { \$this->db->select(\$select); return \$this->db->get_where(\$table, \$where); } </pre>
	<pre> public function update(\$table, \$data, \$where) { return \$this->db->update(\$table, \$data, \$where); } </pre>
<p>Ketika halaman profile termuat, maka app akan memuat data singleton yang sebelumnya diload dari penyimpanan data lokal yang sudah di isikan pada saat berhasil login. Lalu menampilkannya pada halaman profile</p> <p>Ketika button logout di tap, maka app akan membersihkan & menghapus data user dari singleton dan penyimpanan data lokal</p>	<p>Menggunakan API Server yang sama dengan API Server Login untuk memberikan informasi pengguna kepada android.</p> <p>Penggunaan Model getWhere untuk mendapatkan data khusus dari tabel tertentu dan Model update untuk memperbarui data ke tabel tertentu.</p>

Penggunaan Model getWhere untuk mendapatkan data khusus dari tabel tertentu dan Model insert untuk memasukkan data ke tabel tertentu.

Assistant Login Source Code

Assistant Login Source Code	
Android Proses	Api Server
<pre> await _repo.auth .login(username: _username.text, password: _password.text, token: _token.fromToken,) .then((res) async { setState(() { _isLoading = false; }); _helper.showToast("Berhasil Login!"); _model.user = UserModel.fromJson(res.data["data"]); await SPData().saveToSP(xDUser, sharedDataType.toString(), jsonEncode(_model.user.toJson());); await Future.delayed(Duration(seconds: 1)); _helper.navigateToPage(context, page: MainPage()); }).catchError((e) { _helper.showToast("Gagal Login!"); setState(() { _msg = "Gagal Login!"\n\\$_helper.onErrorHandler(e)"; }); }); </pre> <pre> Future<dio.Response> login({ @required String username, @required String password, @required String token, }) { return _repo.auth.login(data: { "username": username, "password": password, "token": token, }); } </pre>	<pre> public function login() { if (\$this->input->method() == 'post') { // user // login() // success \$post = \$this->input->post(); // check user \$users = \$this->crud->getWhere('u', 'users', ['username' => \$post['username']]&=>row_array()); if (\$users == null) { \$response['code'] = '204'; \$response['message'] = 'User Tidak Ditemukan'; } else { if (\$password_verify(\$post['password'], \$users['password'])) { \$response['code'] = '200'; \$response['message'] = 'Password Tidak Sesuai'; } else { \$response['code'] = '200'; \$response['message'] = 'Data Ditemukan'; \$basicInfo = \$this->crud->getWhere('basic', 'users_basic', ['idusers' => \$users['id']]&=>row_array()); \$this->crud->update('users', ['token' => \$post['token']], ['id' => \$users['id']]); \$tokenInfo = \$this->crud->getWhere('token', 'users', ['id' => \$users['id']]&=>row_array()); \$dataUsers = ['id' => \$users['id'], 'name' => \$users['name'], 'ps' => \$users['ps'], 'alamat' => \$users['alamat'], 'role' => \$users['role'], 'token' => \$usersToken['token'], 'idbasic' => \$users['idbasic'],]; \$response['data'] = \$dataUsers; } } } else { \$response['code'] = '400'; \$response['message'] = 'Masukkan username dan Password!'; } } </pre> <pre> @Route('login') public function getWhere(\$select, \$table, \$where) { \$this->db->select(\$select); return \$this->db->get_where(\$table, \$where); } </pre> <pre> public function update(\$table, \$data, \$where) { return \$this->db->update(\$table, \$data, \$where); } </pre>
<p>Ketika Button Masuk ditap, app mengakses API dan mengirim data dari form.</p> <p>Token dari smartphone didapatkan dari library firebase yang otomatis tergenerate.</p> <p>Setelah mendapatkan callback, jika berhasil maka app akan menyimpan</p>	<p>API Server menerima data username dan password untuk dicocokkan apakah memang sesuai dengan data yang di inputkan oleh pengguna android dengan tabel users. Jika role yang termasuk assistant seperti family, nurse atau pertuni maka akan dikategorikan sebagai assistant</p> <p>Jika sesuai maka akan memberikan informasi pengguna kepada android.</p>

data pengguna pada penyimpanan lokal dan tampilan berpindah ke halaman utama, jika tidak tampilan tetap di halaman login dan menampilkan error pada toast.

Penggunaan Model getWhere untuk mendapatkan data khusus dari tabel tertentu dan Model update untuk memperbarui data ke tabel tertentu.

Assistant Get Alert Source Code

Assistant Get Alert Source Code	
Android Proses	Api Server
<pre> KirimMessageIntent intent = Intent(Intent.ACTION_SEND) .putExtra(Intent.EXTRA_TEXT, message) .putExtra(Intent.EXTRA_EMAIL, email) if (Intent.ACTION_SEND.resolveActivity(mContext) != null) { Intent intentSend = Intent(Intent.ACTION_SEND) .setAction(Intent.ACTION_SEND) .setType("text/plain") .putExtra(Intent.EXTRA_TEXT, message) .putExtra(Intent.EXTRA_EMAIL, email) mContext.startActivity(intent); } else { if (message.startsWith("mailto:")) { email = emailParser.parse(message); } } } KirimMessageIntent intent = Intent(Intent.ACTION_SEND) .putExtra(Intent.EXTRA_TEXT, message) .putExtra(Intent.EXTRA_EMAIL, email) if (Intent.ACTION_SEND.resolveActivity(mContext) != null) { Intent intentSend = Intent(Intent.ACTION_SEND) .setAction(Intent.ACTION_SEND) .setType("text/plain") .putExtra(Intent.EXTRA_TEXT, message) .putExtra(Intent.EXTRA_EMAIL, email) mContext.startActivity(intent); } else { if (message.startsWith("mailto:")) { email = emailParser.parse(message); } } } </pre>	<pre> public function listMintaBantuan(){ if (\$this->input->method() == 'post'){ \$post = \$this->input->post(); \$listJmlHistory = \$this->bean->getBmMintaBantuan(\$post['idUserBMA']); \$response['code'] = '200'; \$response['message'] = 'List Data BM Minta Bantuan'; \$response['data'] = \$listJmlHistory; }else{ \$response['code'] = '400'; \$response['message'] = 'id Pengguna tidak terbaca'; } \$this->output ->set_content_type('application/json') ->set_output(json_encode(\$response, JSON_PRETTY_PRINT)); } </pre>
<pre> class AdminNotification extends Notification { DateTime now = DateTime.now(); var AndroidPlatformChannelSpecifics = AndroidNotificationDetails(packageName, appName, packageName, playSound: true, enableVibration: true, importance: Importance.HIGH, priority: Priority.HIGH,); var iOSPlatformChannelSpecifics = new IOSNotificationDetails(); var platformChannelSpecifics = NotificationDetails(android: AndroidPlatformChannelSpecifics, iOS: iOSPlatformChannelSpecifics,); msg = [notification.title, notification.body, platformChannelSpecifics]; } </pre>	<pre> public function getBmMintaBantuan(\$idUserBMA = null) { \$this->db->select(['bm.idUsersBM', 'bm.status', 'us.nama', 'bl.gewKord', 'bl.latitude', 'bl.longitude']); \$this->db->from('status_bm_sbm'); \$this->db->join('users_usawi usaw', 'usaw.idUsersBM=\$bm.idUsersBM'); \$this->db->join('users_us', 'us.id=\$bm.idUsersBM'); \$this->db->join('bantuan_lokasi bl', 'bl.id=\$bm.idLokasi'); \$this->db->where('bm.status', '1'); \$this->db->where('usaw.idUsersBMA', \$idUserBMA); \$query = \$this->db->get(); return \$query->result_array(); } </pre>
<pre> Future _getMintaBantuan() { FutureBuilder<Snapshot> futureBuilder = FutureBuilder<Snapshot>(Future.value(ConnectionData() .connect() .then((ConnectionData connectionData) { return connectionData.isConnected() ? connectionData.getResponse() : Future.error('Tidak ada koneksi ke server', connectionData.getError()); })); return futureBuilder; } </pre>	

```
Future<dio.Response> dataNeedHelp() {  
  return _repo.assist.dataNeedHelp(data: {  
    "idUserBMA": SingletonModel.android.user.id,  
  });  
}
```

Ketika app mendapatkan notifikasi, app memiliki data user dan role user merupakan BMA, maka app akan menampilkan halaman utama pada bagian beranda/home

Ketika tampilan home termuat, maka app akan mengakses API untuk mendapatkan list BMA yang membutuhkan bantuan berdasarkan id BMA, jika callback berhasil dan ada data maka aplikasi menampilkan data, jika callback berhasil dan tidak ada data maka app akan menampilkan text "Belum ada BM yang meminta bantuan", jika callback error maka app akan menampilkan pesan dan button untuk mengakses ulang API

Api server memberikan seluruh BM terkait yang sedang meminta bantuan dengan menerima id BMA.

Dibuat Model baru menyesuaikan dengan permintaan seluruh data BM yang terkait dengan id BMA yaitu getBmMintaBantuan

Ketika button beri arahan di tap, maka app akan memunculkan dialog untuk merekam dan memutar suara.

Ketika button rekam di tap, maka app akan merekam suara dan menyimpannya pada penyimpanan lokal dengan nama berupa timestamp dan ekstensi mp4

Ketika button putar di tap, maka app akan memutar suara yang sebelumnya direkam

Setelah BMA merekam suara dan file sudah tersimpan, maka app akan mengakses API Give Feedback untuk mengunggah rekaman tersebut dan mengakses ulang API List Minta Bantuan untuk menyegarkan data

Api server menerima id BM target, id BMA dan audio hasil recording android pengguna BMA sebagai feedback. Selanjutnya nama file digunakan untuk penyimpanan sesuai dengan id pengguna dan waktu masuk.

Setelah file diupload maka nama file disimpan dalam database. Selanjutnya dilakukan pencarian data dan token BM untuk dilakukan pengiriman audio recording dalam bentuk notifikasi ke BM terkait yang meminta bantuan.

Penggunaan Model getWhere untuk mendapatkan data khusus dari tabel tertentu dan Model insert untuk memasukkan data ke tabel tertentu. Terakhir pembuatan controller untuk mengirimkan notifikasi dengan satu target dari token BM

Assistant Detail BM Source Code

Assistant Detail BM Source Code	
Android Proses	Api Server
<pre> class ClientDetailPage extends StatefulWidget { final ClientDetail data; ClientDetailPage({required this.data}); @override _ClientDetailPageState createState() => _ClientDetailPageState(); } </pre>	<pre> public function history_bm(){ if (\$this->input->method() == 'post'){ \$post = \$this->input->post(); \$listHistory = \$this->bmaw->getBmHistory(\$post['idUsersBM']); // \$listHistory['jumlahBantuan'] = count(\$listHistory); \$response['code'] = '200'; \$response['message'] = 'Data History BM terbaca'; \$response['data'] = \$listHistory; }else{ \$response['code'] = '400'; \$response['message'] = 'id Pengguna tidak terbaca'; } \$this->output ->set_content_type('application/json') ->set_output(json_encode(\$response, JSON_PRETTY_PRINT)); } public function getBmHistory(\$idUsersBM = null) { \$this->db->select('bi.idUsers, bi.latitude, bi.longitude, bi.powers, bi.create_time, bi.name'); \$this->db->from('bantuan_tambah bi'); \$this->db->join('users u', 'bi=idbi.idUsers', 'left'); \$this->db->where('bi.idUsers', \$idUsersBM); \$query = \$this->db->get(); return \$query->result_array(); } </pre>
<p>Ketika memanggil halaman detail BMA, halaman sebelumnya wajib mengirimkan data dari BMA yang didapatkan dari API List BMA, lalu menampilkannya pada halaman detail BMA</p>	<p>Api server mendapatkan id BM lalu dicari seluruh data BM dan history permintaan bantuannya yang terkait dengan id BMA tersebut</p> <p>Penggunaan Model getWhere untuk mendapatkan data khusus dari tabel tertentu. Dibuat Model baru menyesuaikan dengan permintaan seluruh data BM yang terkait dengan id BMA yaitu getBmHistory</p>

Assistant Profile Source Code

Assistant Profile Source Code	
Android Proses	Api Server
<pre> class _ProfilePageState extends State<ProfilePage> { SingletonModel _model; @override void initState() { super.initState(); _model = SingletonModel.withContext(context); } @override Widget build(BuildContext context) { return Scaffold(body: _buildBody(),); } void _onLogout() async { _model.needsHelp = false; _model.user = null; await SPData().resetSPC(); Helper().navigatePage(context, page: LoginPage()); } </pre>	<pre> public function login() { if (\$this->input-method() == 'post') { // log // login() // request \$post = \$this->input-post(); // check email \$users = \$this->crud-getWhere('u', 'users', ['username' => \$post['username']]); if (\$users == null) { \$response['code'] = '204'; \$response['message'] = 'User Tidak Ditemukan'; } else { if (\$password_verify(\$post['password'], \$users['password'])) { \$response['code'] = '200'; \$response['message'] = 'Password Tidak Sesuai'; } else { \$response['code'] = '200'; \$response['message'] = 'Data Ditemukan'; \$usersData = \$this->crud-getWhere('data', 'users_data', ['iduser' => \$users['id']]); \$usersToken = \$this->crud-getWhere('token', 'users', ['id' => \$users['id']]); \$dataUsers = ['id' => \$users['id'], 'name' => \$users['name'], 'email' => \$users['email'], 'code' => \$usersData['code'], 'token' => \$usersToken['token'], 'photo' => \$users['photo'];]; \$response['data'] = \$dataUsers; } } // login \$response['code'] = '401'; \$response['message'] = 'Masukkan username dan Password'; } } \$this->nextPage(); } // next_page -next_content_type(application/json) -next_output(json_encode(\$response, JSON_PRETTY_PRINT)); </pre>
	<pre> public function getWhere(\$select, \$table, \$where) { \$this->db->select(\$select); return \$this->db->get_where(\$table, \$where); } </pre>
	<pre> public function update(\$table, \$data, \$where) { return \$this->db->update(\$table, \$data, \$where); } </pre>
<p>Ketika halaman profile termuat, maka app akan memuat data singleton yang sebelumnya diload dari penyimpanan data lokal yang sudah di isikan pada saat berhasil login. Lalu menampilkannya pada halaman profile</p> <p>Ketika button logout di tap, maka app akan membersihkan & menghapus data user dari singleton dan penyimpanan data lokal</p>	<p>Menggunakan API Server yang sama dengan API Server Login untuk memberikan informasi pengguna kepada android.</p> <p>Penggunaan Model getWhere untuk mendapatkan data khusus dari tabel tertentu dan Model update untuk memperbarui data ke tabel tertentu.</p>

Assistant Add BM Source Code

Blind Masseur Register Source Code	
Android Proses	Api Server
<pre>void _onAdd() async { String kode = await openAddBmDialog(context); if (kode != null) { await _request.assist .addMasseur(kode: kode, idUser: _model.user.id) .then((res) async { await openAlertDialog(context, title: "Pendaftaran Berhasil!", content: "Berhasil Menambahkan BM!",); setState(() { _futureData = _createDataFuture(); }); }).catchError((e) { openAlertDialog(context, title: "Pendaftaran Gagal!", content: "Gagal Menambahkan BM!",); }); } }</pre>	<pre>public function create_dataBM() { if (\$this->input->method() == 'post') { \$kode = \$this->input->post(); //cek token \$cekTokenBM = \$this->repo->getInfoKode(\$kode); if (\$cekTokenBM['jml'] == 0) { //cek token BM \$idUserBM = \$this->ctrl->getWhere('idUsers', 'user_kode', ['kode' => \$kode]); // jika \$idUserBM['idUser'] != \$idUserBM['idUser'] \$kode = \$this->ctrl->getWhere('user_kode', 'user_kode', ['idUserBM' => \$kode]); \$idUserBM = \$kode; if (\$kode == null) { \$dataBM = ['idUserBM' => \$kode, 'idUserBM' => \$kode, 'idUserBM' => \$kode, 'idUserBM' => \$kode,]; \$response['code'] = '201'; \$response['message'] = 'Data BM berhasil ditambahkan untuk diupload!'; } else { \$response['code'] = '201'; \$response['message'] = 'Data BM sudah pernah ditambahkan!'; } } else if (\$cekTokenBM['jml'] == 0) { \$response['code'] = '204'; \$response['message'] = 'Token Bukan milik BM!'; } else { \$response['code'] = '400'; \$response['message'] = 'Data Tidak Terbaca!'; } } \$this->output ->set_content_type('application/json') ->set_output(json_encode(\$response, JSON_PRETTY_PRINT)); }</pre>
<pre>void _onSave() { if (_formKey.currentState.validate()) { Navigator.pop(context, _controller.text); } }</pre>	<pre>public function getInfoKode(\$kode = null){ \$this->db->select('count(us.id) as jml'); \$this->db->from('users_kode uk'); \$this->db->join('users us', 'us.id=uk.idUsers', 'left'); \$this->db->where('uk.kode', \$kode); \$this->db->where('us.idRole', '1'); \$query = \$this->db->get(); return \$query->row_array(); }</pre>
<pre>Future<dio.Response> addMasseur({ @required String kode, @required String idUser, }) { return _repo.assist.addMasseur(data: { "kode": kode, "idUserBMA": idUser, }); }</pre>	<pre>public function getInfoKode(\$kode = null){ \$this->db->select('count(us.id) as jml'); \$this->db->from('users_kode uk'); \$this->db->join('users us', 'us.id=uk.idUsers', 'left'); \$this->db->where('uk.kode', \$kode); \$this->db->where('us.idRole', '1'); \$query = \$this->db->get(); return \$query->row_array(); }</pre>
<p>Ketika button tambah di tap, maka app akan memunculkan dialog untuk mengisikan kode.</p> <p>Ketika button simpan pada dialog di tap, maka dialog akan ditutup dan mengakses API Add BM, setelah mendapatkan callback maka app akan memunculkan dialog dengan pesan sesuai dari callback(Berhasil/Gagal), jika callback berhasil maka app akan mengakses ulang API List Masseur untuk menyegarkan data</p>	<p>Api server menerima token dari android dan mencocokkannya untuk mendapatkan data BM. Setelah mendapatkan data, id BM dikaitkan dengan id BMA yang dikirimkan melalui android.</p> <p>Penggunaan Model getWhere untuk mendapatkan data khusus dari tabel tertentu dan Model insert untuk memasukkan data ke tabel tertentu. Dibuat Model baru menyesuaikan dengan permintaan seluruh data BM untuk dikaitkan dengan id BMA yaitu getInfoKode</p>

