###### Lampiran A

**LAMPIRAN A**

userController.php (Menampilkan form pemilihan jadwal kursus dan tampilan konfirmasi jadwal kursus)

public function showMeetingForm($student\_real\_name, $enrollment\_id, $meeting\_number) {

// Fetch enrollment and course details

$enrollment = Enrollment::findOrFail($enrollment\_id);

$course = $enrollment->course;

// Check if this is the last meeting

$isLastMeeting = $meeting\_number == $course->course\_length;

// Find the enrollment data for this student

$enrollment = enrollment::with(['schedule', 'coursePayment'])->find($enrollment\_id);

// Manipulate and localize this page to Indonesian

Carbon::setLocale('id');

// Get the Open Time of the Admin's

$openTime = \Carbon\Carbon::parse($enrollment->course->admin->open\_hours\_for\_admin);

// Get the Close Time of the Admin's

$closeTime = \Carbon\Carbon::parse($enrollment->course->admin->close\_hours\_for\_admin);

// Get the course duration of the selected schedule

$courseDuration = $enrollment->course->course\_duration;

// Generate the course time option until the start time is not more than close time

while ($openTime->lessThan($closeTime)) {

// Generate the end time from adding the open time with course duration

$endOptionTime = $openTime->copy()->addMinutes($courseDuration);

// When the end time is passed the close time, end the generation

if ($endOptionTime->greaterThan($closeTime)) {

break; // Exit the loop if it exceeds

}

// Add the slot to available slots

$availableSlots[] = [

'start' => $openTime->format('H:i'),

'end' => $endOptionTime->format('H:i'),

];

// Create new start time by adding the previous start time with course duration

$openTime->addMinutes($courseDuration);

}

// Return the form for selecting the date and time for the current meeting

return view('student-page.user-choose-schedule', [

'pageName' => "Pilih Jadwal Kursus | ",

'enrollment' => $enrollment,

'course' => $course,

'meeting\_number' => $meeting\_number,

'availableSlots' => $availableSlots,

'isLastMeeting' => $isLastMeeting,

]);

}

public function showConfirmationPage($student\_real\_name, $enrollment\_id) {

// Fetch the schedule data from the session

$enrollment = Enrollment::findOrFail($enrollment\_id);

$course = $enrollment->course;

$student\_real\_name = $enrollment->student\_real\_name;

$meetings = [];

for ($i = 1; $i <= $course->course\_length; $i++) {

$meetings[] = [

'date' => session()->get("meeting\_{$i}\_date"),

'time' => session()->get("meeting\_{$i}\_time"),

];

}

Carbon::setLocale('id');

return view('student-page.user-confirm-schedule', [

'pageName' => "Konfirmasi Jadwal Kursus | ",

'enrollment' => $enrollment,

'student\_real\_name' => $student\_real\_name,

'meetings' => $meetings,

]);

}

CourseScheduleController.php (Memeriksa tabrakan jadwal dan menyimpan jadwal yang dipilih)

public function storeMeetingData(Request $request, $student\_real\_name, $enrollment\_id, $meeting\_number) {

$request->validate([

'course\_date' => 'required|date',

'course\_time' => 'required',

], [

'course\_date.required' => 'Silahkan Pilih Tanggal Kursus',

'course\_time.required' => 'Silahkan Pilih Salah Satu Opsi',

]);

Carbon::setLocale('id'); // This can be set in a service provider or at the start of your controller

// Get the Enrollment Data

$enrollmentData = enrollment::findOrFail($enrollment\_id);

// Fetch the instructor\_id from Enrollment Data

$instructor\_id = $enrollmentData['instructor\_id'];

// Get the course\_length

$courseLength = $enrollmentData->course->course\_length;

// dd($request);

// Extract start and end times from course\_time

list($start\_time\_str, $end\_time\_str) = explode(' - ', $request->course\_time);

$currentDate = \Carbon\Carbon::parse($request->course\_date);

// Create full datetime for start and end times

$selectedStartTime = \Carbon\Carbon::parse($currentDate->format('Y-m-d') . ' ' . $start\_time\_str);

$selectedEndTime = \Carbon\Carbon::parse($currentDate->format('Y-m-d') . ' ' . $end\_time\_str);

// dd($selectedStartTime, $selectedEndTime);

// Check for earlier date than the previous meeting

if ($meeting\_number > 1) {

$previousMeetingDate = session()->get("meeting\_" . ($meeting\_number - 1) . "\_date");

$previousMeetingDateCarbon = \Carbon\Carbon::parse($previousMeetingDate);

if ($currentDate->lt($previousMeetingDateCarbon)) {

$request->session()->flash(

'error',

'Tanggal untuk Pertemuan ' . $meeting\_number . ' tidak boleh lebih awal dari tanggal Pertemuan ' . ($meeting\_number - 1) . ' (' . $previousMeetingDateCarbon->translatedFormat('d F Y') . ').'

);

return redirect()->back();

}

}

// Check for conflicts here (but don't store in database yet)

$existingSchedule = courseSchedule::where(function ($query) use ($instructor\_id, $selectedStartTime, $selectedEndTime) {

$query->where('instructor\_id', $instructor\_id)

->where(function ($q) use ($selectedStartTime, $selectedEndTime) {

$q->whereBetween('start\_time', [$selectedStartTime, $selectedEndTime])

->orWhereBetween('end\_time', [$selectedStartTime, $selectedEndTime])

->orWhere(function ($q) use ($selectedStartTime, $selectedEndTime) {

$q->where('start\_time', '<=', $selectedStartTime)

->where('end\_time', '>=', $selectedEndTime);

});

});

})->first();

// When there's a conflicting schedule, return error

if ($existingSchedule) {

$request->session()->flash('error', 'Instruktur ' . $existingSchedule->instructor->fullname . ' sudah terjadwal pada pukul ' . $request->course\_time . ' di ' . $currentDate->translatedFormat('d F Y') . '. Silakan pilih waktu atau tanggal lain.');

return redirect()->back();

}

// Store the selected meeting data in session

session()->put("meeting\_{$meeting\_number}\_date", $request->course\_date);

session()->put("meeting\_{$meeting\_number}\_time", $request->course\_time);

// dd(session()->get("meeting\_{$meeting\_number}\_date"), session()->get("meeting\_{$meeting\_number}\_time"));

// Redirect to the next meeting or confirmation page

$nextMeetingNumber = $meeting\_number + 1;

if ($nextMeetingNumber > $courseLength) {

return redirect()->route('schedule.confirmation', [

'student\_real\_name' => $student\_real\_name,

'enrollment\_id' => $enrollment\_id,

]);

} else {

return redirect()->route('schedule.form', [

'student\_real\_name' => $student\_real\_name,

'enrollment\_id' => $enrollment\_id,

'meeting\_number' => $nextMeetingNumber,

]);

}

}

public function saveSchedule(Request $request, $student\_real\_name, $enrollment\_id) {

// Save all meeting data to the database

$enrollment = Enrollment::findOrFail($enrollment\_id);

$course = $enrollment->course;

// Retrieve the submitted meetings from the request

$meetings = $request->input('meetings');

// Save each meeting

foreach ($meetings as $meetingNumber => $meetingData) {

$newSchedule = new CourseSchedule();

$newSchedule->enrollment\_id = $enrollment\_id;

$newSchedule->course\_id = $course->id;

$newSchedule->instructor\_id = $enrollment->instructor\_id;

// Parse date and time

$newSchedule->start\_time = Carbon::parse($meetingData['date'] . ' ' . $meetingData['start\_time']);

$newSchedule->end\_time = Carbon::parse($meetingData['date'] . ' ' . $meetingData['end\_time']);

$newSchedule->meeting\_number = $meetingNumber; // Convert to 1-based index

$newSchedule->theoryStatus = 0;

$newSchedule->quizStatus = 0;

// dd($newSchedule);

$newSchedule->save();

}

// Clear the session

session()->forget('meeting\_\*');

$request->session()->flash('success', 'Jadwal berhasil dibuat. Silahkan hubungi Admin Kursus untuk proses lebih lanjut.');

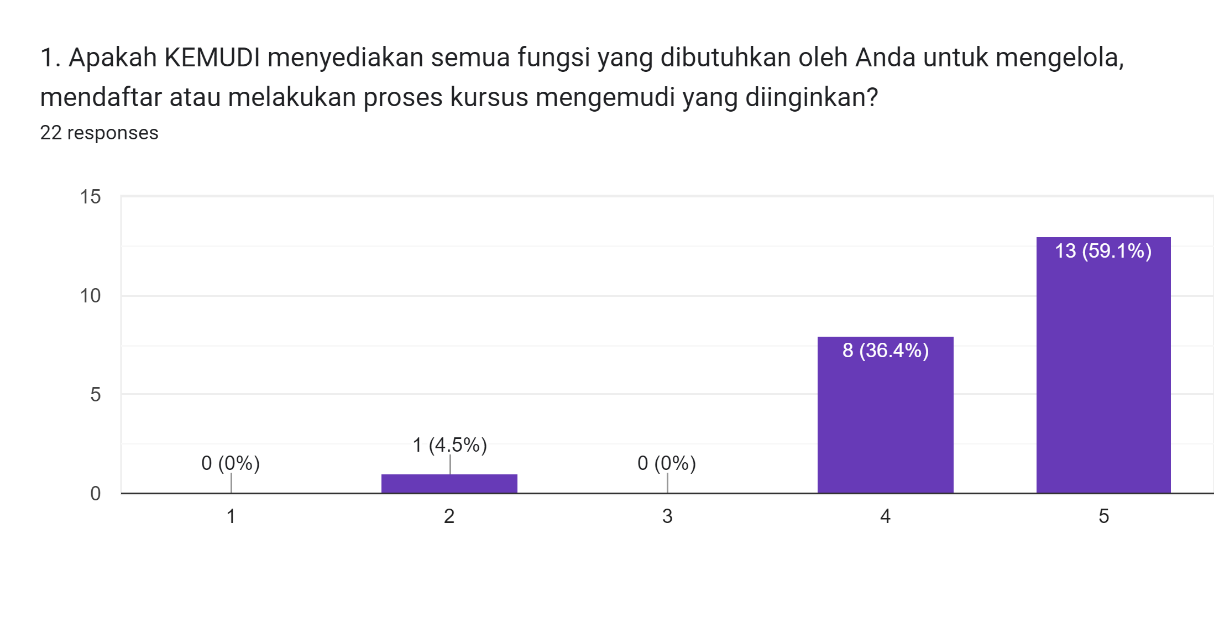
return redirect(url('/user-course-progress/' . $student\_real\_name . '/' . $enrollment\_id));

}

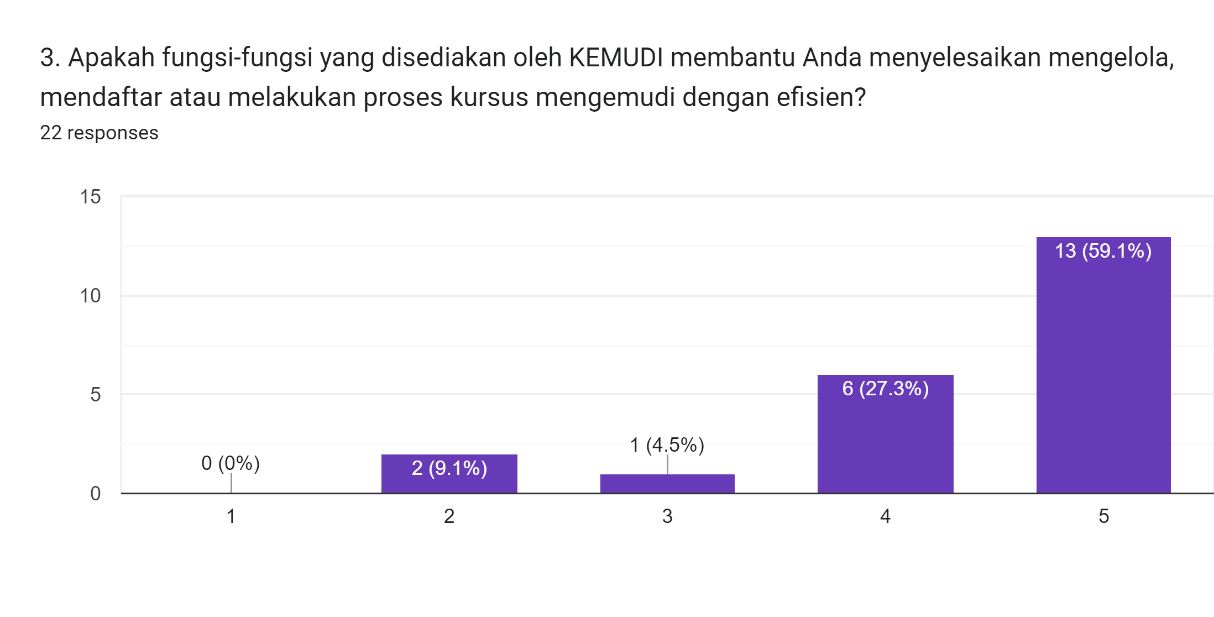
###### Lampiran B

**LAMPIRAN B**

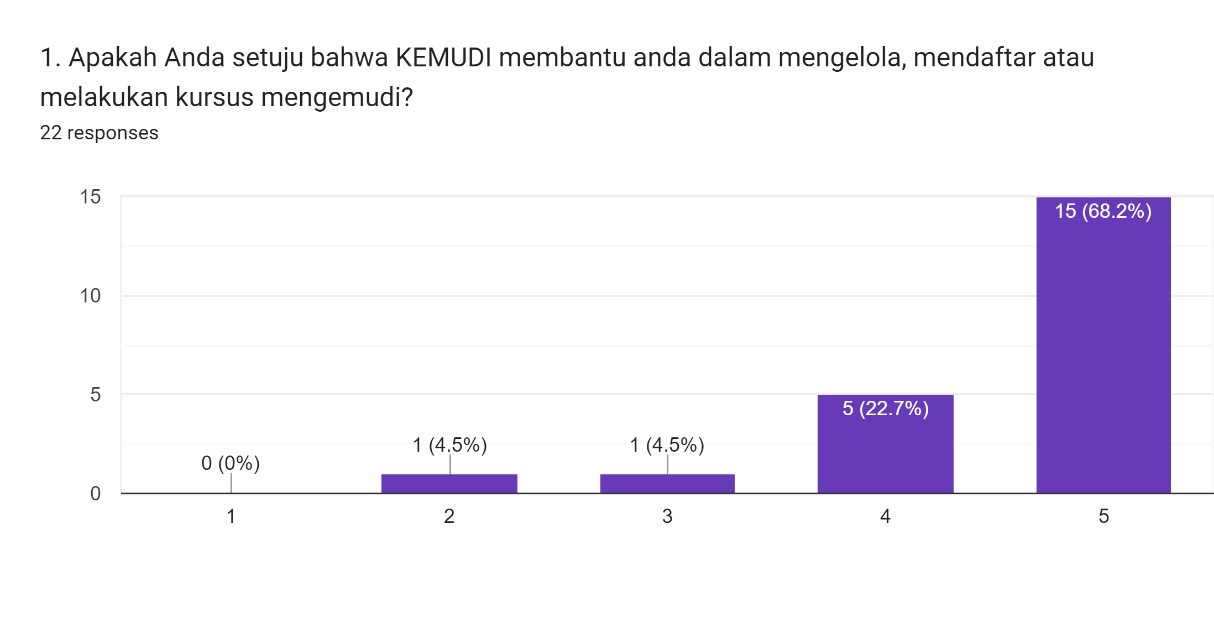
Pertanyaan dan hasil jawaban untuk Faktor *Functional Suitability*

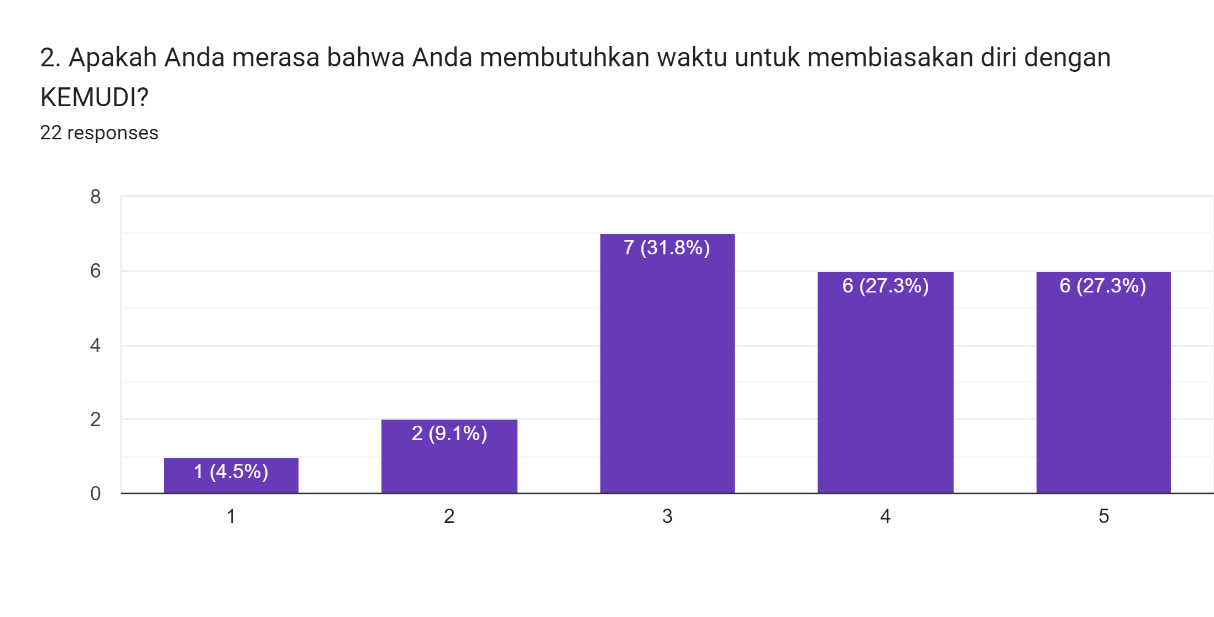


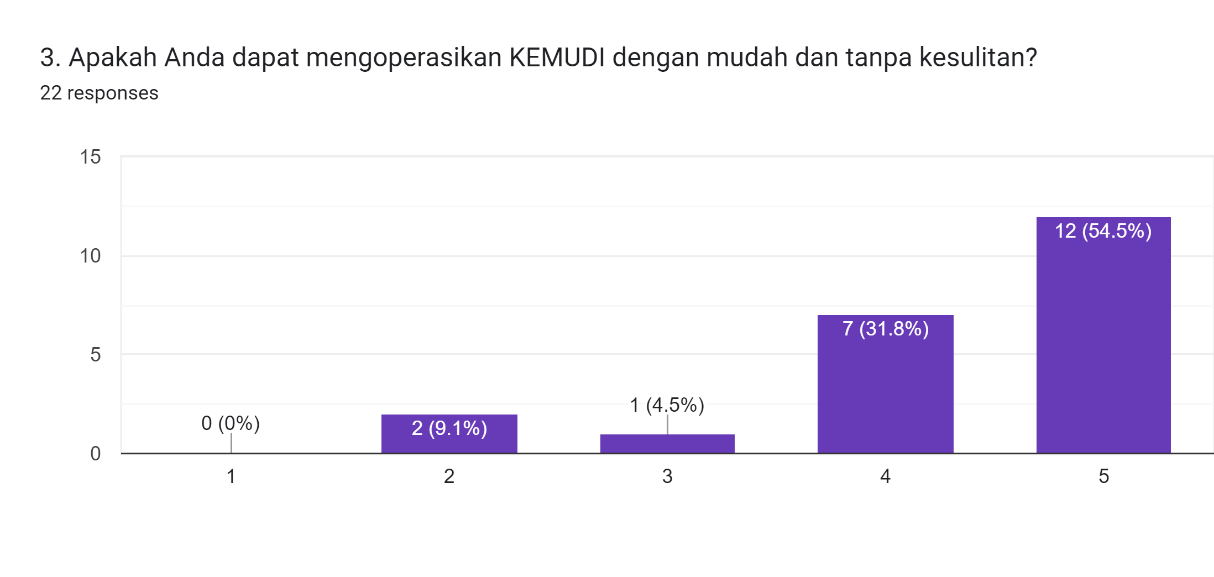




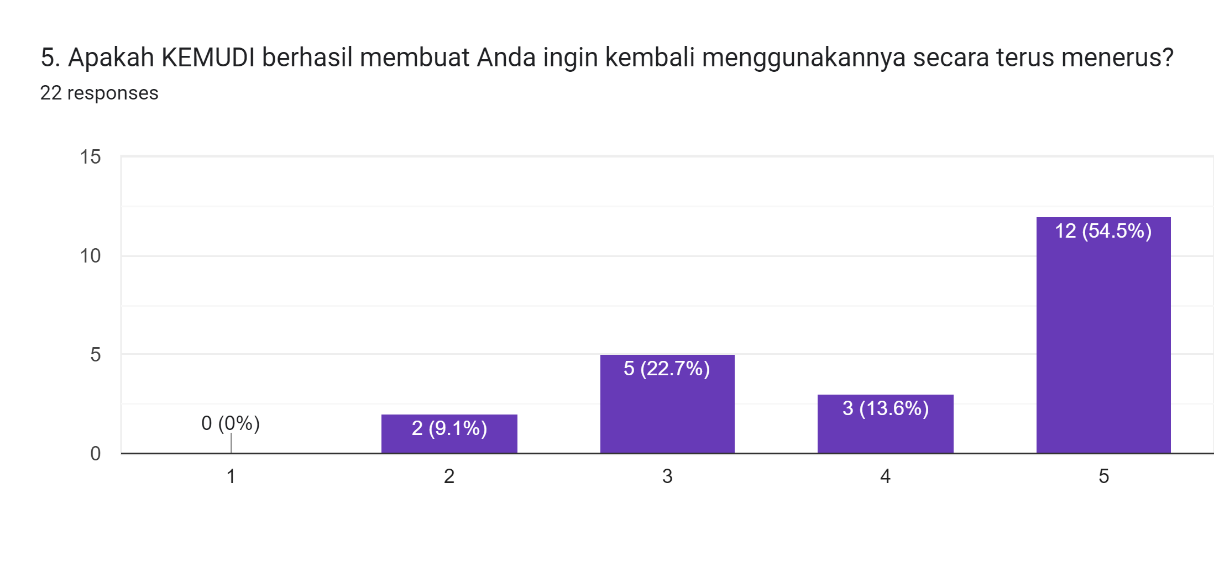
Pertanyaan dan hasil jawaban untuk Faktor *Interaction Capability*

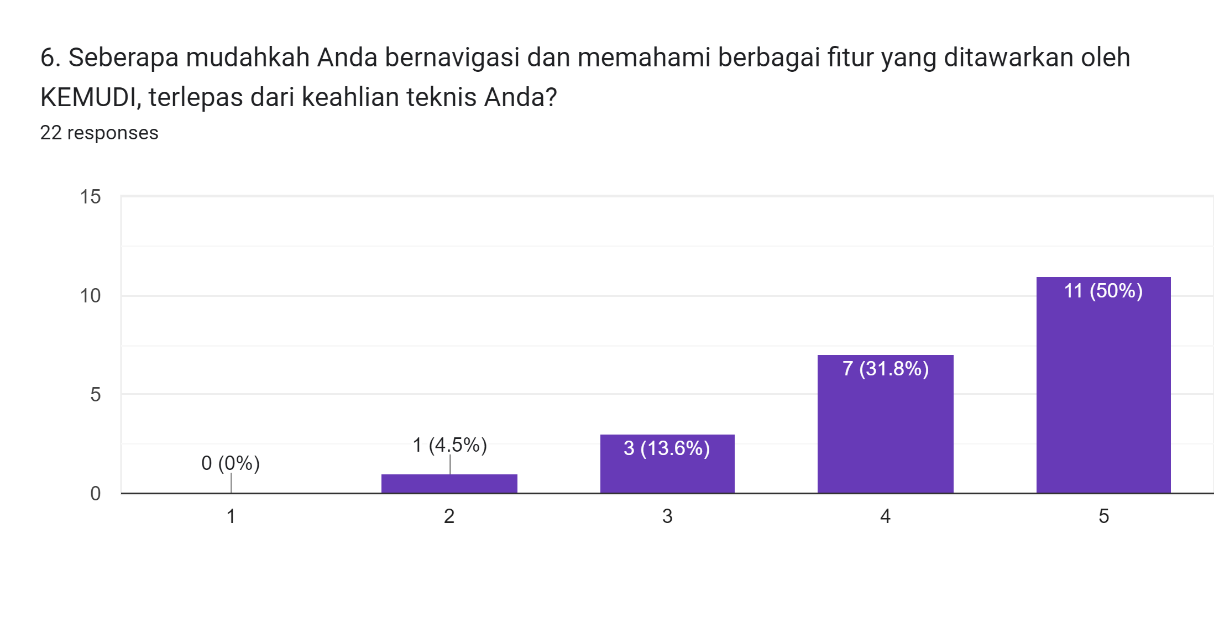




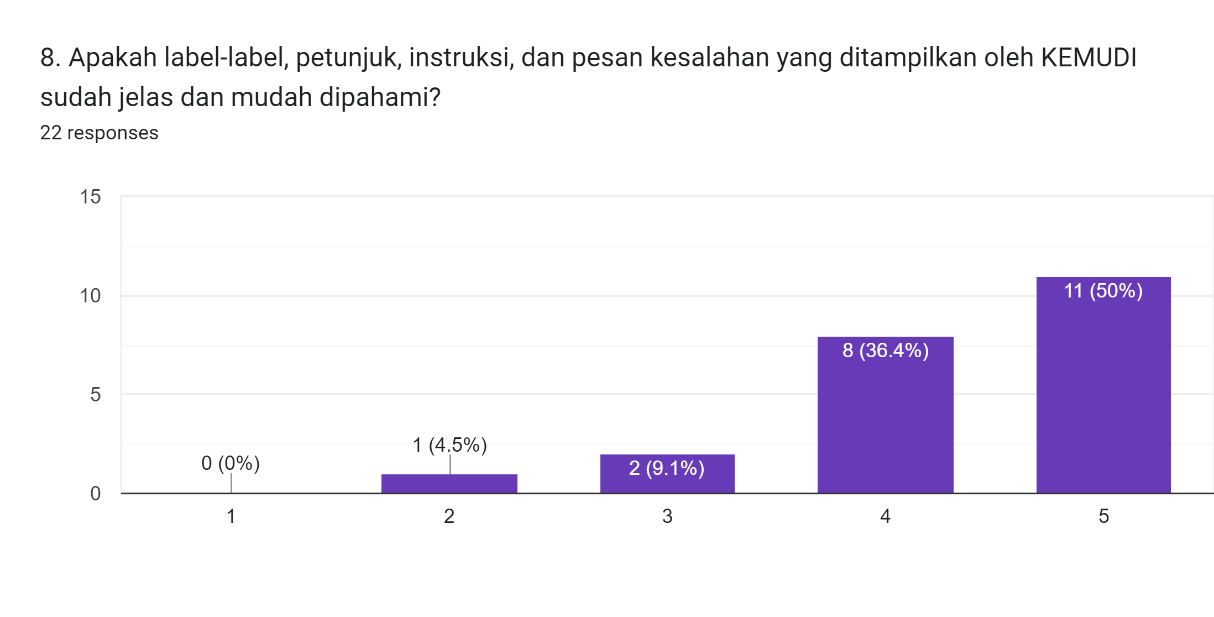












###### Lampiran C

**LAMPIRAN C**