

**RANCANG BANGUN APLIKASI
UNTUK PENYEDIA JASA KURSUS MENGEMUDI
BERBASIS WEB MENGGUNAKAN METODE PROTOTYPE**

PROPOSAL SKRIPSI



Oleh :

YODANIS ERLANDI SUTANTIO

13.2020.1.00905

PROGRAM STRATA-1

JURUSAN SISTEM INFORMASI

FAKULTAS TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI ADHI TAMA SURABAYA

2024

DAFTAR ISI

	Halaman
DAFTAR ISI.....	ii
DAFTAR GAMBAR.....	v
DAFTAR TABEL.....	vii
BAB I PENDAHULUAN.....	1
1.1. Latar Belakang.....	1
1.2. Rumusan Masalah.....	5
1.3. Tujuan Penelitian.....	6
1.4. Batasan Masalah.....	6
1.5. Sistematika Penulisan.....	6
BAB II LANDASAN TEORI.....	8
2.1. Kursus Mengemudi.....	8
2.2. Rekayasa Perangkat Lunak.....	8
2.3. Sistem Informasi Berbasis Web.....	10
2.4. Metode Rekayasa Perangkat Lunak <i>Prototyping</i>	13
2.4.1. Tahap 1 : <i>Requirements Gathering</i>	15
2.4.2. Tahap 2 : <i>Quick Design</i>	16
2.4.3. Tahap 3 : <i>Built Prototype</i>	17
2.4.4. Tahap 4 : <i>Customer Evaluation</i>	18
2.4.5. Tahap 5 : <i>Review & Updation</i>	19
2.4.6. Tahap 6 : <i>Design</i>	20
2.4.7. Tahap 7 : <i>Development</i>	21
2.4.8. Tahap 8 : <i>Test</i>	22
2.4.9. Tahap 9 : <i>Maintenance</i>	23
2.5. <i>Business Process Model and Notation (BPMN)</i>	24
2.6. <i>Unified Modeling Language (UML)</i>	26
2.7. <i>Use Case Diagram</i>	27
2.8. <i>Activity Diagram</i>	29
2.9. <i>Class Diagram</i>	32

2.10. <i>Hyper Text Markup Language (HTML)</i>	34
2.11. <i>Cascading Style Sheets (CSS)</i>	35
2.12. <i>Tailwind CSS Framework</i>	35
2.13. <i>Javascript</i>	36
2.14. <i>Framework PHP Laravel</i>	36
2.14.1. <i>Routing</i>	37
2.14.2. <i>Arsitektur MVC</i>	38
2.14.3. <i>Views dan Templates</i>	39
2.14.4. <i>Controller</i>	39
2.14.5. <i>Berinteraksi dengan database</i>	40
2.14.6. <i>Model dan Eloquent</i>	41
2.14.7. <i>Authentication dan Authorization</i>	42
2.14.8. <i>Middleware</i>	42
2.14.9. <i>Validation</i>	43
2.14.10. <i>Pengamanan aplikasi</i>	43
BAB III METODE PENELITIAN	44
3.1. <i>Requirements Gathering</i>	44
3.1.1. <i>Proses Bisnis</i>	44
3.1.2. <i>Analisa Kebutuhan</i>	50
3.2. <i>Iterasi Pertama</i>	53
3.2.1. <i>Quick Design</i>	53
3.2.1.1. <i>Alur Pengguna</i>	53
3.2.1.2. <i>Sketsa Awal Aplikasi</i>	58
3.2.2. <i>Built Prototype</i>	62
3.2.3. <i>Customer Evaluation</i>	66
3.2.4. <i>Review & Updation</i>	69
3.3. <i>Iterasi Kedua</i>	69
3.3.1. <i>Quick Design</i>	69
3.3.1.1. <i>Alur Pengguna</i>	70
3.3.1.2. <i>Sketsa Awal Aplikasi</i>	73
3.3.2. <i>Built Prototype</i>	79

3.3.3.	<i>Customer Evaluation</i>	84
3.3.4.	<i>Review & Updation</i>	85
3.4.	<i>Design</i>	85
3.4.1.	<i>Login / Daftar Akun</i>	89
3.4.2.	<i>Siswa Mendaftar Kursus</i>	92
3.4.3.	<i>Pembayaran</i>	94
3.4.4.	<i>Mengubah Jadwal Kursus</i>	96
3.4.5.	<i>Mengajukan Kursus Baru</i>	99
3.4.6.	<i>Dashboard Jadwal Kursus</i>	104
3.4.7.	<i>Pengelolaan Kelas Kursus</i>	107
3.4.8.	<i>Menampilkan Detail Progress Kursus untuk Pihak Kursus</i>	111
3.4.9.	<i>Menampilkan Detail Progress Kursus untuk Siswa</i>	113
3.4.10.	<i>Class Diagram</i>	115
3.5.	<i>Development</i>	117
3.6.	<i>Test</i>	117
3.7.	<i>Maintenance</i>	121
DAFTAR PUSTAKA		123

DAFTAR GAMBAR

	Halaman
Gambar 1. 1 Faktor Penyebab Terbesar	2
Gambar 2. 1 Siklus Metode <i>Prototyping</i>	15
Gambar 2. 2 Notasi <i>Class Diagram</i>	33
Gambar 2. 3 Skema MVC	39
Gambar 2. 4 Ilustrasi <i>Model</i> dan <i>Eloquent</i>	41
Gambar 3. 1 Proses Pendaftaran pada Kursus Mengemudi ABC	46
Gambar 3. 2 Proses Kursus di Kursus Mengemudi ABC	47
Gambar 3. 3 Proses Pendaftaran di Kursus Mengemudi Sie Bersaudara	47
Gambar 3. 4 Proses Kursus di Kursus Mengemudi Sie Bersaudara	48
Gambar 3. 5 Proses Pendaftaran di Kursus Mengemudi "Hafiz"	49
Gambar 3. 6 Proses Kursus di Kursus Mengemudi "Hafiz"	49
Gambar 3. 7 Alur Pengguna untuk Iterasi Pertama.....	54
Gambar 3. 8 Alur Tugas Proses <i>Login</i>	55
Gambar 3. 9 Alur Tugas Proses Pendaftaran Kelas Kursus	56
Gambar 3. 10 Alur Tugas Proses Perubahan Jadwal	57
Gambar 3. 11 Alur Tugas Proses <i>Login</i>	71
Gambar 3. 12 Alur Tugas Proses Pendaftaran Kelas Kursus.....	72
Gambar 3. 13 Alur Tugas Proses Perubahan Jadwal	73
Gambar 3. 14 <i>Use Case Diagram</i> Aplikasi.....	85
Gambar 3. 15 <i>Use Case</i> untuk pengguna <i>General User</i>	86
Gambar 3. 16 <i>Use Case</i> untuk pengguna Instruktur	87
Gambar 3. 17 <i>Use Case</i> untuk pengguna Admin/Pemilik Kursus	88
Gambar 3. 18 <i>Activity Diagram</i> untuk proses <i>Login</i>	89
Gambar 3. 19 <i>Activity Diagram</i> untuk proses Siswa Mendaftar Kursus	92
Gambar 3. 20 <i>Activity Diagram</i> untuk proses Pembayaran	94
Gambar 3. 21 <i>Activity Diagram</i> untuk proses Mengubah Jadwal Kursus	96
Gambar 3. 22 <i>Activity Diagram</i> untuk proses Mengajukan Kursus Baru.....	99

Gambar 3. 23 <i>Activity Diagram</i> untuk proses menampilkan <i>Dashboard</i> Jadwal Kursus	104
Gambar 3. 24 <i>Activity Diagram</i> untuk proses Mengelola Kelas Kursus	107
Gambar 3. 25 <i>Activity Diagram</i> untuk proses Menampilkan Detail Progress Kursus untuk Pihak Kursus	111
Gambar 3. 26 <i>Activity Diagram</i> untuk proses Menampilkan Detail Progress Kursus untuk Siswa	113
Gambar 3. 27 <i>Class Diagram</i> Aplikasi untuk Penyedia Jasa Kursus Mengemudi	116

DAFTAR TABEL

	Halaman
Tabel 2. 1 Notasi-notasi BPMN	24
Tabel 2. 2 Notasi-notasi BPMN (Lanjutan-1).....	25
Tabel 2. 3 Notasi-notasi BPMN (Lanjutan-1).....	26
Tabel 2. 4 Notasi-notasi <i>Use Case Diagram</i>	28
Tabel 2. 5 Notasi-notasi <i>Use Case Diagram</i> (Lanjutan-1)	29
Tabel 2. 6 Notasi-notasi <i>Activity Diagram</i>	30
Tabel 2. 7 Notasi-notasi <i>Activity Diagram</i> (Lanjutan-1)	31
Tabel 2. 8 Notasi-notasi <i>Activity Diagram</i> (Lanjutan-2)	32
Tabel 3. 1 Pemenuhan Legalitas Jasa Kursus Mengemudi	45
Tabel 3. 2 Kebutuhan Aplikasi.....	50
Tabel 3. 3 Kebutuhan Aplikasi (Lanjutan-1)	51
Tabel 3. 4 Kebutuhan Aplikasi (Lanjutan-2)	52
Tabel 3. 5 Sketsa Awal untuk Iterasi Pertama	58
Tabel 3. 6 Sketsa Awal untuk Iterasi Pertama (Lanjutan-1)	59
Tabel 3. 7 Sketsa Awal untuk Iterasi Pertama (Lanjutan-2)	60
Tabel 3. 8 Sketsa Awal untuk Iterasi Pertama (Lanjutan-3)	61
Tabel 3. 9 Desain <i>Prototype</i> untuk Iterasi Pertama.....	62
Tabel 3. 10 Desain <i>Prototype</i> untuk Iterasi Pertama (Lanjutan-1)	63
Tabel 3. 11 Desain <i>Prototype</i> untuk Iterasi Pertama (Lanjutan-2)	64
Tabel 3. 12 Desain <i>Prototype</i> untuk Iterasi Pertama (Lanjutan-3)	65
Tabel 3. 13 Evaluasi Pihak Kursus dan Pengguna Aplikasi	66
Tabel 3. 14 Evaluasi Pihak Kursus dan Pengguna Aplikasi (Lanjutan-1)	67
Tabel 3. 15 Evaluasi Pihak Kursus dan Pengguna Aplikasi (Lanjutan-2)	68
Tabel 3. 16 Evaluasi Pihak Kursus dan Pengguna Aplikasi (Lanjutan-3)	69
Tabel 3. 17 Sketsa Awal untuk Iterasi Kedua	74
Tabel 3. 18 Sketsa Awal untuk Iterasi Kedua (Lanjutan-1).....	75
Tabel 3. 19 Sketsa Awal untuk Iterasi Kedua (Lanjutan-2).....	76
Tabel 3. 20 Sketsa Awal untuk Iterasi Kedua (Lanjutan-3).....	77

Tabel 3. 21 Sketsa Awal untuk Iterasi Kedua (Lanjutan-4).....	78
Tabel 3. 22 Desain <i>Prototype</i> untuk Iterasi Kedua	79
Tabel 3. 23 Desain <i>Prototype</i> untuk Iterasi Kedua (Lanjutan-1)	80
Tabel 3. 24 Desain <i>Prototype</i> untuk Iterasi Kedua (Lanjutan-2)	81
Tabel 3. 25 Desain <i>Prototype</i> untuk Iterasi Kedua (Lanjutan-3)	82
Tabel 3. 26 Desain <i>Prototype</i> untuk Iterasi Kedua (Lanjutan-4)	83
Tabel 3. 27 Desain <i>Prototype</i> untuk Iterasi Kedua (Lanjutan-5)	84
Tabel 3. 28 <i>Use Case Scenario</i> untuk proses <i>Login</i>	89
Tabel 3. 29 <i>Use Case Scenario</i> untuk proses <i>Login</i> (Lanjutan-1)	90
Tabel 3. 30 <i>Use Case Scenario</i> untuk proses <i>Login</i> (Lanjutan-2)	91
Tabel 3. 31 <i>Use Case Scenario</i> untuk proses Siswa Mendaftar Kursus	92
Tabel 3. 32 <i>Use Case Scenario</i> untuk proses Siswa Mendaftar Kursus (Lanjutan-1)	93
Tabel 3. 33 <i>Use Case Scenario</i> untuk proses Siswa Mendaftar Kursus (Lanjutan-2)	94
Tabel 3. 34 <i>Use Case Scenario</i> untuk proses Pembayaran	95
Tabel 3. 35 <i>Use Case Scenario</i> untuk proses Pembayaran (Lanjutan-1).....	96
Tabel 3. 36 <i>Use Case Scenario</i> untuk proses Mengubah Jadwal Kursus	97
Tabel 3. 37 <i>Use Case Scenario</i> untuk proses Mengubah Jadwal Kursus (Lanjutan-1)	98
Tabel 3. 38 <i>Use Case Scenario</i> untuk proses Mengubah Jadwal Kursus (Lanjutan-2)	99
Tabel 3. 39 <i>Use Case Scenario</i> untuk proses Mengajukan Kursus Baru.....	100
Tabel 3. 40 <i>Use Case Scenario</i> untuk proses Mengajukan Kursus Baru (Lanjutan-1)	101
Tabel 3. 41 <i>Use Case Scenario</i> untuk proses Mengajukan Kursus Baru (Lanjutan-2)	102
Tabel 3. 42 <i>Use Case Scenario</i> untuk proses Mengajukan Kursus Baru (Lanjutan-3)	103
Tabel 3. 43 <i>Use Case Scenario</i> untuk proses menampilkan <i>Dashboard</i> Jadwal Kursus	104

Tabel 3. 44 <i>Use Case Scenario</i> untuk proses menampilkan <i>Dashboard</i> Jadwal Kursus (Lanjutan-1)	105
Tabel 3. 45 <i>Use Case Scenario</i> untuk proses menampilkan <i>Dashboard</i> Jadwal Kursus (Lanjutan-2)	106
Tabel 3. 46 <i>Use Case Scenario</i> untuk Mengelola Kelas Kursus.....	107
Tabel 3. 47 <i>Use Case Scenario</i> untuk Mengelola Kelas Kursus (Lanjutan-1) ...	108
Tabel 3. 48 <i>Use Case Scenario</i> untuk Mengelola Kelas Kursus (Lanjutan-2) ...	109
Tabel 3. 49 <i>Use Case Scenario</i> untuk Mengelola Kelas Kursus (Lanjutan-3) ...	110
Tabel 3. 50 <i>Use Case Scenario</i> untuk proses Menampilkan Detail Progress Kursus untuk Pihak Kursus	111
Tabel 3. 51 <i>Use Case Scenario</i> untuk proses Menampilkan Detail Progress Kursus untuk Pihak Kursus (Lanjutan-1)	112
Tabel 3. 52 <i>Use Case Scenario</i> untuk proses Menampilkan Detail Progress Kursus untuk Pihak Kursus	113
Tabel 3. 53 <i>Use Case Scenario</i> untuk proses Menampilkan Detail Progress Kursus untuk Pihak Kursus (Lanjutan-1)	114
Tabel 3. 54 Contoh Tabel <i>Black Box Testing</i>	118
Tabel 3. 55 Contoh Tabel <i>Black Box Testing</i> (Lanjutan-1)	119
Tabel 3. 56 Contoh Tabel Pengujian <i>System Usability Scale</i>	119
Tabel 3. 57 Contoh Tabel Pengujian <i>System Usability Scale</i> (Lanjutan-1)	120
Tabel 3. 58 Contoh Tabel Pengujian <i>System Usability Scale</i> (Lanjutan-2)	121

BAB I

PENDAHULUAN

1.1. Latar Belakang

Penting rasanya untuk memahami teori dasar dalam berkendara dan memiliki kemampuan berkendara yang baik agar kita selalu berhati-hati ketika di jalan raya yang terkadang memiliki banyak halangan dan rintangan. Saat ini, banyak penyedia jasa yang dapat membantu kita memahami teori dan mengajarkan kita kemampuan untuk berkendara dengan baik melalui kursus mengemudi. Sering sekali kita menemui pengemudi yang sedang belajar di jalan raya dan biasanya mereka didampingi dengan mentor/instruktur untuk memandu dan memberikan arahan. Sehingga, ketika mereka berhasil lulus dari kursus tersebut, mereka diharapkan dapat memahami aturan-aturan di jalan raya, seperti cara mengemudi melewati persimpangan atau merubah lajur berkendara dengan benar. Selain itu, mereka bisa mengoperasikan kendaraannya dengan aman dan nyaman bagi penumpang, mengadopsi kebiasaan-kebiasaan yang seringkali ditinggalkan atau tidak dihiraukan, seperti mengecek kaca spion sebelum belok, menghindari berkendara di *blind spot* dll., dengan mengetahui hal-hal tersebut diharapkan para pengemudi dapat meminimalisir risiko mereka mengalami kecelakaan.

Sebab, melalui data yang dibagikan oleh Korps Lalu Lintas Kepolisian Republik Indonesia, pada Januari tahun 2022 hingga September 2022 terjadi setidaknya 94.617 kasus kecelakaan lalu lintas di seluruh wilayah Indonesia (Sumber : dephub.go.id). Lebih lanjut lagi Korlantas Polri menjelaskan faktor-faktor yang memiliki andil pada 94.617 kasus kecelakaan tersebut, faktor-faktor tersebut diantaranya 61% kasus kecelakaan diakibatkan oleh faktor kesalahan manusia seperti kurang terampilnya pengemudi dalam mengendalikan kendaraannya, serta kelalaian-kelalaian yang lainnya seperti mengemudi dalam kondisi mengantuk, ugal-ugalan, dan lain lain.



Gambar 1. 1 Faktor Penyebab Terbesar Kecelakaan Lalu Lintas

Selanjutnya Korlantas Polri mengungkapkan pula bahwa pada sepanjang tahun 2021 sebanyak 25.266 korban merenggut jiwa akibat kecelakaan lalu lintas. Pada tahun 2022 angka ini mengalami peningkatan menjadi 26.100 korban jiwa, dengan ini dapat kita sadari bahwa, betapa bahayanya kondisi lalu lintas di negara kita, oleh karena itu, sangat penting sekali bagi kita untuk tidak berkendara apalagi kita memang tidak siap karena faktor-faktor tertentu atau merasa kurang mahir dalam mengemudi. Kementerian Perhubungan bersama dengan Korlantas Polri selanjutnya bekerja sama dengan masyarakat untuk menekan angka kecelakaan yang semakin tahun semakin naik dengan diadakannya sekolah atau kursus mengemudi yang sering kita temui.

Namun, narasi diatas bisa dikatakan narasi untuk mendeskripsikan suatu kondisi yang ideal, namun, kenyataannya terdapat beberapa kendala yang dapat menghambat seseorang untuk mengikuti kursus mengemudi. Bahkan tidak sedikit pengemudi yang sudah mendapatkan Surat Izin Mengemudi (SIM) tanpa mereka mengetahui teori dan teknik berkendara dengan aman. Akibatnya, banyak sekali pelanggaran atau bahkan kecelakaan yang diakibatkan oleh pengemudi-pengemudi

ini, contohnya, salah menginjak pedal, berkendara ugal-ugalan, atau menyalip menggunakan bahu jalan, dsb. Masalah-masalah yang menjadi faktor mengapa masyarakat pada umumnya tidak mengikuti kursus mengemudi diantaranya adalah mereka takut biaya yang harus disiapkan terlalu besar, mereka sudah memiliki kesibukan lain sehingga tidak memiliki waktu, atau mereka belum menyadari manfaat yang bisa didapatkan dari kursus mengemudi ini. Masalah tentang kemauan mereka untuk diedukasi inilah yang ingin kami jembatan dengan teknologi dan pemanfaatan sistem informasi.

Sistem informasi adalah salah satu disiplin ilmu yang mempelajari tentang Rekayasa Perangkat Lunak / *Software Engineering*. *Software Engineering* (SE) atau yang sering kita kenal Rekayasa Perangkat Lunak (RPL) merupakan sebuah disiplin yang menerapkan prinsip-prinsip *design*, *development*, *testing*, dan *maintenance* perangkat lunak. RPL adalah pendekatan sistematis untuk membangun perangkat lunak yang berkualitas, efisien, dapat diandalkan, dan mudah di *maintain*. Dengan memanfaatkan Rekayasa Perangkat Lunak, kami berharap dapat menyelesaikan masalah yang disebutkan sebelumnya, serta diharapkan dengan memanfaatkan RPL, tingkat kepuasan terhadap layanan kursus mengemudi dapat semakin baik. Penelitian tentang pemanfaatan rekayasa perangkat lunak untuk peningkatan layanan ini sebelumnya pernah dilakukan pada tahun 2022 dan tahun 2023.

Penelitian yang dilakukan oleh (Made et al., 2022) mengangkat permasalahan yang sama tentang proses bisnis yang belum dilakukan secara efektif dimana pelajar kursus tidak menerima informasi terbaru tentang kursus, metode registrasi yang masih konvensional membutuhkan waktu yang lama, dan kendala terhadap jadwal kursus. Selain itu, terdapat penelitian yang serupa (Adhiva Kurnia, 2023) dengan permasalahan yang hampir serupa. Dengan mengintegrasikan layanan kursus mengemudi dan teknologi informasi, dua sistem yang dirancang menggunakan metode *Waterfall*, berhasil mengatasi semua masalah yang disebutkan. Pelajar kursus merasa proses registrasi menjadi lebih mudah dan lebih nyaman tanpa perlu

mendatangi tempat kursus. Sistem tersebut juga mengurangi beban kerja dan waktu yang diperlukan oleh kedua belah pihak, penyedia jasa dan pelajar kursus.

Apabila kita mengacu kepada masalah yang sudah disebutkan sebelumnya, maka setidaknya kita dapat menyelesaikannya dengan memanfaatkan Rekayasa Perangkat Lunak dengan metode *Prototyping*, masalah-masalah tersebut yaitu : Ketakutan akan biaya yang tinggi, sama halnya dengan solusi yang ditawarkan oleh *online marketplace*, aplikasi yang kami bangun nantinya dapat menyelesaikan masalah transparansi biaya tersebut, dengan menampilkan harga setiap paket yang ditawarkan oleh penyedia jasa, para pengguna nantinya dapat memperkirakan rentang biaya yang harus mereka persiapkan sebelum mengikuti kursus mengemudi. *Prototyping* sendiri dipilih agar para pihak yang terlibat dalam proses pengembangan aplikasi ini mudah memberikan ide, saran, bahkan umpan balik untuk proyek ini kedepannya.

Tujuan utama dari aplikasi kami selain untuk mengedukasi masyarakat tentang betapa pentingnya peraturan-peraturan dan etika ketika berkendara adalah menjadi wadah bagi penyedia jasa kursus mengemudi untuk menawarkan jasanya kepada masyarakat umum. Selain itu, dengan pemanfaatan teknologi informasi, mutu pelayanan yang ditawarkan saat ini akan jadi lebih baik dan mengalami peningkatan daripada sebelumnya. Tidak hanya itu, dengan adanya sistem informasi ini, kami dapat membantu orang-orang yang berkeinginan untuk belajar mengemudi dengan baik dan benar agar keamanan dan kenyamanan mereka sendiri dapat terjamin. Para penyedia jasa kursus mengemudi dapat mendaftarkan dirinya ke aplikasi kami (bisa sebagai perorangan maupun sebagai perusahaan) dan menampilkan paket belajar yang mereka tawarkan. Sedangkan, pengguna yang ingin mendaftarkan dirinya untuk mengikuti kursus mengemudi dapat langsung menggunakan aplikasi kami tanpa melakukan proses *Sign-In*, namun, apabila mereka ingin mendaftarkan diri ke suatu kursus mengemudi, mereka wajib mendaftarkan diri mereka sehingga, transparansi antar kedua sisi pengguna dapat terjaga.

Singkatnya, kami ingin masyarakat untuk memahami bahaya dan risiko yang ditemui apabila mereka mengemudikan kendaraan tanpa mengikuti suatu pelatihan untuk mengasah pemahaman dan kemampuannya terlebih dahulu tentang topik tersebut. Kami ingin menjadi wadah bagi penyedia jasa kursus mengemudi untuk memberikan edukasi ke masyarakat luas bahwa mengikuti kursus mengemudi tidak memerlukan biaya yang tinggi dan walaupun jika mereka tetap menganggap bahwa biaya yang dibutuhkan masih terlalu tinggi, setidaknya, mereka mengerti betapa pentingnya untuk mengikuti kursus mengemudi sebelum berkendara langsung di jalan raya. Dengan Rekayasa Perangkat Lunak, kami ingin mengurangi beban dari penyedia kursus mengemudi konvensional yang harus mencetak formulir pendaftaran, modul, sertifikat, dan yang lainnya sehingga semuanya dapat terintegrasi di satu aplikasi, sehingga tingkat pelayanan mereka menjadi lebih baik dari sebelumnya. Sehingga kami tertarik untuk menulis skripsi dengan judul : **“RANCANG BANGUN APLIKASI UNTUK PENYEDIA JASA KURSUS MENGEMUDI BERBASIS WEB MENGGUNAKAN METODE PROTOTYPE”**

1.2. Rumusan Masalah

Dari Latar Belakang diatas, dapat disimpulkan bahwa terdapat 2 pokok permasalahan yang mendasari kami untuk membuat skripsi ini:

- Bagaimana merancang aplikasi yang dapat membantu, memudahkan peserta dan penyedia kursus dalam melakukan rangkaian proses kursus?
- Bagaimana mengembangkan aplikasi untuk penyedia jasa kursus mengemudi dengan menggunakan metode *prototype*?

1.3. Tujuan Penelitian

Tujuan dari penelitian ini yaitu untuk menjawab rumusan masalah diatas dengan:

- Merancang aplikasi yang memudahkan peserta dan penyedia kursus mengemudi dalam melakukan rangkaian proses kursus seperti proses administrasi, penjadwalan kursus, dan lain sebagainya.
- Dengan metode *prototype*, pengembangan aplikasi nantinya diharapkan membutuhkan waktu yang lebih singkat dan mampu mengadaptasi perubahan-perubahan yang disarankan oleh pemangku kepentingan.

1.4. Batasan Masalah

Dari Latar Belakang dan Rumusan Masalah aplikasi yang kami kembangkan mempunyai batasan sebagai berikut:

- Beroperasi di wilayah Kota Surabaya
- Dapat diakses dari *smartphone* dan laptop
- Menggunakan teknologi berbasis web

1.5. Sistematika Penulisan

Dalam dokumen proposal ini, kami nantinya akan menuliskan setidaknya 3 bab yang membahas proses awal penelitian, bab-bab tersebut diantaranya :

BAB I PENDAHULUAN

Pada bab ini kami menjelaskan apa yang melandasi kami melakukan penelitian ini, seperti pada umumnya, isi dari bab ini adalah latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, dan sistematika penulisan.

BAB II LANDASAN TEORI

Di bab ini kami akan memberikan pembaca pemahaman dasar tentang istilah-istilah atau teknik-teknik yang nantinya kami sering sebutkan dan sering kami gunakan dalam menyelesaikan penelitian kami.

BAB III METODOLOGI PENELITIAN

Bab ini bertujuan untuk menjelaskan tentang tahapan-tahapan penelitian serta pengembangan perangkat lunak dengan metode *prototype*, kendala yang dihadapi, sebagian kecil dari alur sistem, dan persiapan-persiapan sebelum melanjutkan proses *development* / pemrograman.

BAB II

LANDASAN TEORI

2.1. Kursus Mengemudi

Lembaga Kursus adalah salah satu penyelenggara pendidikan diluar sekolah resmi (non-formal) untuk mengembangkan kemampuan dan keterampilan diri (Mahdy et al., 2021). Kursus Mengemudi secara spesifik dapat diartikan suatu pendidikan untuk mengembangkan kemampuan dan keterampilan diri dalam mengemudikan kendaraan khususnya mobil. KBBI sendiri mendeskripsikan kursus sebagai pelajaran tentang suatu pengetahuan atau keterampilan, yang diberikan dalam waktu singkat. Atau bisa juga diartikan sebagai lembaga di luar sekolah yang memberikan pelajaran serta pengetahuan atau keterampilan yang diberikan dalam waktu singkat.

2.2. Rekayasa Perangkat Lunak

Secara bahasa, *Software Engineering* atau yang sering disebut dengan Rekayasa Perangkat Lunak dalam bahasa indonesia (RPL) tersusun dari 2 kata, *Software* dan *Engineering*. *Software* atau perangkat lunak bukan hanya sebuah program seperti yang diasumsikan banyak orang. Program adalah kode komputer yang dapat dieksekusi, yang tujuannya untuk melakukan satu atau lebih komputasi tertentu. Sedangkan *software* adalah kumpulan dari kode pemrograman komputer yang dapat dieksekusi, terorganisir dan terdokumentasi. Lalu, *Engineering* sendiri adalah semua hal yang meliputi pengembangan produk (baik fisik maupun digital) dengan menerapkan dan memanfaatkan prinsip-prinsip dan metodologi ilmiah yang tersusun dengan baik.

Sehingga, *Software Engineering* merupakan salah satu bagian dari *Engineering* yang berurusan dengan rekayasa perangkat lunak yang disusun secara baik dan menerapkan prinsip-prinsip, prosedur dan metodologi ilmiah. Hasil akhir

dari RPL tentunya adalah perangkat lunak yang efisien, stabil dan bebas dari *bug*. Namun, apabila kita mengacu kepada pengertian menurut organisasi IEEE, RPL adalah (1) Penerapan dari sebuah pendekatan yang sistematis, disiplin, dan terukur untuk mengembangkan, mengoperasikan dan memelihara sebuah perangkat lunak; dalam hal ini penerapan rekayasa perangkat lunak. (2) Studi tentang pendekatan-pendekatan dari pernyataan diatas.

Selain IEEE, Fritz Bauer, ilmuwan komputer asal Jerman, mendefinisikan RPL sebagai "...pembentukan dan pemanfaatan prinsip teknik suara agar tercipta perangkat lunak yang ekonomis, yang stabil dan bekerja secara efisien di mesin sesungguhnya.". Mengutip dari modul pembelajaran Rekayasa Perangkat Lunak – Pendekatan Terstruktur & Berorientasi Objek karya (Bahar et al., n.d.) Rekayasa perangkat lunak (*software engineering*) merupakan suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak (*software*), mulai dari tahap awal kajian spesifikasi / kebutuhan sistem sampai pemeliharaan sistem setelah digunakan (Sommerville, 2016). Pada definisi ini, ada dua istilah kunci: yang Pertama 'Disiplin rekayasa', yang berarti bahwa teknisi RPL membuat suatu alat bekerja. Mereka menerapkan teori, metode, dan alat bantu yang sesuai, selain itu mereka menggunakannya dengan selektif dan selalu mencoba mencari solusi terhadap permasalahan, walaupun tidak ada teori atau metode yang mendukung.

Selanjutnya teknisi RPL juga menyadari bahwa mereka harus bekerja dalam batasan organisasi dan keuangan, sehingga mereka berusaha mencari solusi dalam batasan-batasan ini. Kemudian yang kedua, 'Semua aspek produksi perangkat lunak', yang berarti Rekayasa Perangkat Lunak tidak hanya berhubungan dengan proses teknis dari pengembangan perangkat lunak tetapi juga dengan kegiatan seperti manajemen proyek perangkat lunak dan pengembangan alat bantu, metode, dan teori untuk mendukung produksi perangkat lunak. Bayangkan Rekayasa Perangkat Lunak seperti kita menuntun komputer untuk melakukan sebuah tugas sesuai dengan instruksi yang dituliskan sebelumnya. Teknisi RPL juga dapat diibaratkan sebagai arsitek di dunia digital, dimana mereka memanfaatkan

wawasan ilmu komputer dan keahliannya untuk merancang, mengembangkan, menguji, dan meluncurkan aplikasi perangkat lunak.

2.3. Sistem Informasi Berbasis Web

Sistem Informasi menurut Richard Vidgen adalah sebuah kumpulan komponen-komponen yang berinteraksi, komponen yang dimaksud adalah manusia, prosedur-prosedur, dan teknologi-teknologi yang ada, dimana komponen-komponen tersebut secara bersamaan mengumpulkan, memproses, menyimpan, dan mendistribusikan informasi untuk mendukung pengontrolan, pembuatan keputusan dan pengelolaan organisasi/perusahaan. Sistem informasi sendiri berisi informasi tentang organisasi/perusahaan terkait, contohnya, kondisi tentang operasional internal mereka, dan tentang lingkungan di dalam perusahaan tersebut, sebagai contoh informasi tentang para pelanggan, para supplier, dan kompetitor-kompetitor yang ada. Tanpa adanya sistem informasi, sebuah organisasi sulit untuk bertahan. Namun, bukan berarti bahwa sistem informasi harus menggunakan teknologi-teknologi informasi yang berbentuk komputer-komputer atau jaringan internet dan komunikasi, karena banyak sekali bentuk sistem informasi. Organisasi atau perusahaan sangat bergantung pada sistem informasi, meskipun aspek-aspek formal dari sistem informasi ini masih menggunakan sistem pengarsipan berupa kertas pada era sebelum adanya teknologi informasi itu sendiri.

Pengertian dari “teknologi” tidak bisa kita terjemahkan semudah yang dibayangkan. Teknologi bisa dan terkadang diaplikasikan ke perangkat-perangkat seperti komputer dan jaringan internet dan komunikasi, namun, juga bisa diaplikasikan ke praktik lapangan (contohnya : proses rekayasa perangkat lunak), dan teknik (contohnya : perancangan basis data). Sistem Informasi dapat diharapkan untuk memanfaatkan teknologi informasi, tetapi tidak berarti bahwa sistem informasi adalah teknologi. Sebuah sistem informasi pada intinya merupakan sebuah sistem yang melibatkan aktivitas manusia yang berada pada lingkup konteks organisasi/perusahaan. Tidak dapat dipungkiri, teknologi berperan

sangat penting bagi sistem informasi, namun, tidak bisa kita mengabaikan peranan manusia dan dimensi-dimensi organisasi.

Fitzgerald (1998) menulis sebuah laporan yang membahas tentang krisis perangkat lunak, dimana rata-rata durasi pengerjaan untuk proyek pemrograman sistem informasi (PSI) berkisar antara 18 bulan sampai dengan 5 tahun, kemudian, pada laporan tersebut dikemukakan juga bahwa, 68% dari keseluruhan proyek melebihi jadwal yang ditetapkan, kemudian 75% dari keseluruhan proyek harus dirancang ulang setelah proses implementasi dan 35% perusahaan setidaknya memiliki satu proyek yang sudah ditinggalkan. Dapat disimpulkan bahwa proyek pengembangan perangkat lunak merupakan sebuah proyek yang sulit dan beberapa akan berpendapat bahwa solusi dari permasalahan ini adalah dengan melakukan pendekatan yang lebih baik dan lebih profesional pada saat proses pengembangan. Salah satu area yang coba mereka lakukan peningkatan adalah metodologi pengembangan perangkat lunak. Avison & Fitzgerald (2002) mendefinisikan metodologi sebagai berikut:

“Sebuah kumpulan prosedur, teknik, alat-alat dan bantuan dokumentasi yang dapat membantu pengembang sistem dalam rangkaian usaha mereka untuk mengimplementasi sebuah sistem informasi yang baru. Sebuah metodologi biasanya terdiri dari fase-fase, yang didalamnya akan terdapat sub-sub fase, yang akan mengarahkan pengembang sistem dalam memilih teknik yang paling sesuai pada setiap tahapan sebuah proyek dan akan membantu mereka dalam merencanakan mengelola, mengontrol dan mengevaluasi proyek sistem informasi.”

Banyak sekali yang mendasari penggunaan metodologi untuk mengarahkan pemrograman sistem informasi. Menurut Fitzgerald (1996), alasan-alasan tersebut diantaranya adalah :

- Pembagian proses yang kompleks menjadi tugas-tugas yang dapat dikelola.
- Memfasilitasi kontrol dan pengelolaan proyek. Salah satu peran pengelolaan adalah untuk mengelola risiko dan ketidakpastian.

- Kerangka kerja yang mengarahkan aplikasi teknik-teknik.
- Ekonomis, spesialisasi keahlian dan pembagian beban kerja.
- Secara hakikat, sebuah kerangka kerja untuk mengakuisisi dan melakukan sistematisasi pengetahuan. Sebuah metodologi harus mampu mendukung pembelajaran organisasi.
- Standarisasi, tim pengembang yang fleksibel, peningkatan produktivitas dan kualitas.

Namun, Fitzgerald memberikan daftar-daftar permasalahan yang dibawa oleh metodologi ini. Ada satu masalah mendasar tentang apa sebenarnya sebuah metodologi (anomali pengertian) dan sebuah tendensi bagi para pencipta metodologi untuk melakukan “perang metodologi”. Banyak metodologi yang minim dasar konseptual dan empiris, selain itu, ketika ada konsep yang mendasari metodologi tersebut cenderung berakar pada sebuah paradigma saintifik dan/atau teknik. Hal ini juga didukung oleh Wastell (1996) bahwa metodologi menjadi tujuan tersendiri :

“Metodologi menjadi sebuah *fetish*, sebuah prosedur yang digunakan dengan kekakuan patologis demi kepentingannya sendiri, bukan sebagai alat untuk mencapai sebuah tujuan. Dengan pemanfaatannya seperti saat ini, metodologi memberikan kesan meringankan terhadap keragu-raguan, metodologi ini mengisolasi pelaku pengembangan sistem informasi dari risiko-risiko dan ketidakpastian yang bisa diperoleh dengan melibatkan manusia dan masalah-masalah yang ada.”

Saat ini banyak bermunculan metode-metode pengembangan sistem informasi, metode-metode tersebut diantaranya yang paling populer adalah *Waterfall*, dan salah satu metode hasil pengembangannya *Rapid Application Development* (RAD), kemudian *Prototyping*, dan juga *Agile Software Development*, yang mencakup *Extreme Programming* (XP), SCRUM, dan *Lean Development*.

2.4. Metode Rekayasa Perangkat Lunak *Prototyping*

Asosiasi antara *Prototyping* dan *Software Engineering* berawal saat *Software Engineering* sendiri baru saja dikemukakan antara tahun 1970-1980an. Metode ini menjadi salah satu yang terbaik diantara metode lain. Dibuktikan dengan banyaknya perangkat lunak yang dikembangkan dengan metode ini mendapatkan penerimaan yang baik di kalangan pengguna. Dan pengalaman pengguna (*User Experience*) terhadap penggunaan aplikasi atau sistem informasi yang dirancang dengan metode ini dianggap cukup baik. Berbeda dengan metode konvensional seperti *Waterfall*, dimana teknisi RPL harus menjelaskan terus menerus ke pemangku kepentingan dan calon pengguna tentang bagaimana bentuk aplikasi nantinya, metode *Prototyping* memvisualisasikan hasil perancangannya kepada dua belah pihak tersebut bahkan sampai melibatkan mereka pada tahap pengembangan untuk mendapatkan umpan balik agar produk yang dirancang dan dikembangkan oleh teknisi RPL sesuai dengan kemauan dan arahan mereka (pemangku kepentingan dan calon pengguna) (Bard, 2023).

Prototyping menurut Kamus Webster 1913 adalah (produk) asli atau model (sederhana) yang kemudian disalin; pola apa pun yang akan diukir, atau disalin, dicetak atau sejenisnya (ke produk akhir). Kemudian sebagai pembanding, Dictionary.com mendefinisikan *prototype* sebagai (produk) asli atau model (sederhana) yang menjadi dasar atau bentuk sesuatu. Sedangkan Kamus Besar Bahasa Indonesia (KBBI) mendefinisikan prototipe sebagai, model yang mula-mula (model asli) yang menjadi contoh; contoh baku; contoh khas. Dari 3 sumber diatas, dapat kita cermati, ketiganya selalu menyebutkan ‘model’. Sehingga, dapat disimpulkan, *prototype* adalah sebuah model untuk memvisualisasikan akan seperti apa produk akhir nantinya.

Dalam *prototyping*, umumnya terdapat 3 tujuan yang ingin dicapai (Arnowitz et al., 2007), inovasi (Leonardo Da Vinci), penyempurnaan ide dan kebutuhan (Thomas Alfa Edison), serta komunikasi dengan pemangku kepentingan dan evaluasi (Henry Dreyfuss), tujuan-tujuan diatas dianggap masih relevan dengan

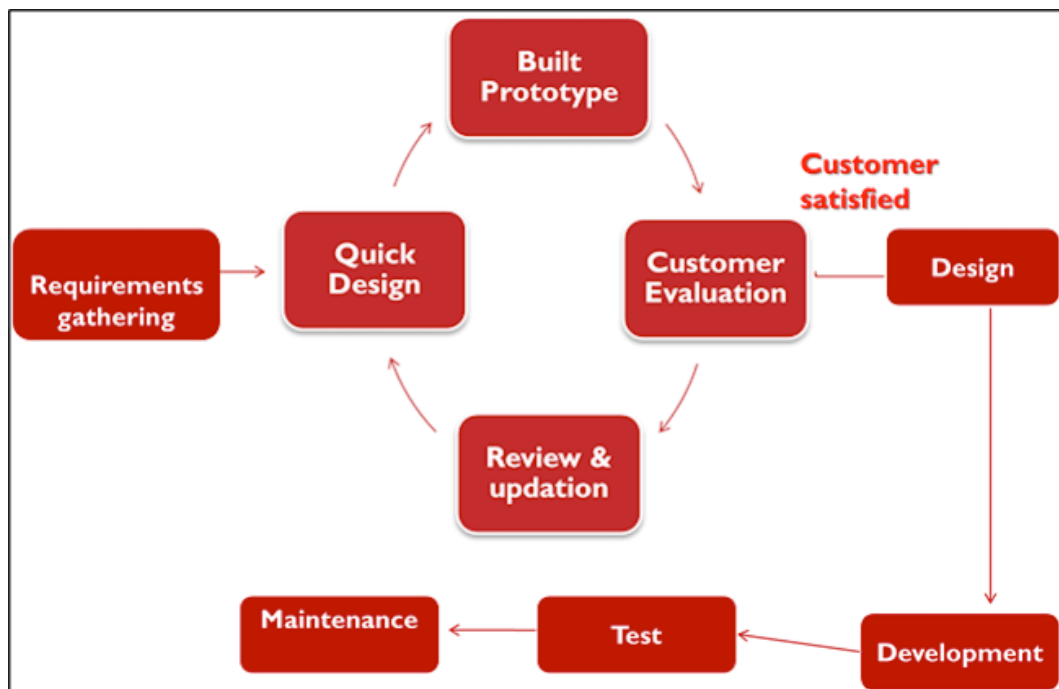
kondisi dan keadaan saat ini, dapat kita artikan juga sebagai faktor yang mendorong penggunaan *prototyping* menjadi lebih canggih dan kompleks. Rekayasa perangkat lunak menggunakan metode *prototyping* seringkali dihadapkan dengan kebutuhan yang berbeda-beda yang belum tentu saling melengkapi satu sama lain bahkan hingga bertentangan. Berikut adalah 4 contoh kebutuhan yang bertolak belakang, yang pertama, perusahaan termotivasi untuk merilis produk yang terbaik dengan sesegera mungkin untuk mendapatkan keuntungan. Kedua, perusahaan dituntut untuk merilis sebuah produk dan fitur baru sebelum pesaing mereka melakukannya terlebih dahulu. Ketiga, terdapat risiko bahwa sebuah produk atau fitur baru ini bisa jadi tidak terlalu diinginkan oleh pengguna, walaupun dirilis tepat waktu dan sesuai dengan budget perusahaan. Keempat, terkadang pengguna tidak menginginkan produk atau fitur yang dianggap menguntungkan atau terbaik bagi perusahaan, sehingga terdapat konflik kepentingan antara kebutuhan pengguna dan keinginan perusahaan.

Garis besarnya disini adalah metode *prototyping* diharapkan mampu meminimalisir risiko-risiko terkait dengan rekayasa perangkat lunak dengan menciptakan sebuah model dari perangkat lunak yang dikembangkan sebelum sepenuhnya dirilis ke publik. Hal ini memungkinkan perusahaan untuk mendapatkan umpan balik dari pengguna dan pemangku kepentingan selama proses pengembangan dan menyumbangkan ide dan gagasannya kepada produk yang dikembangkan sesuai dengan keinginannya. Pada proses rekayasa perangkat lunak, *prototype* digunakan untuk berbagai tujuan, mulai dari membuat bayangan agar kemudian dikembangkan menjadi produk untuk digunakan oleh calon pengguna. Sebuah *prototype* dapat dimanfaatkan untuk menguji ide-ide kecil sampai besar.

Sebuah *prototype* yang dirancang sedini mungkin dapat dimanfaatkan untuk merencanakan inovasi, konsep fitur dan produk yang lebih mudah untuk ditangani. Dengan ini, *prototyping* memberikan kita penentuan lingkup yang lebih akurat, sehingga memungkinkan penghitungan waktu pengerjaan, penjadwalan pengerjaan, dan perencanaan biaya yang lebih baik. Pekerjaan diawal ini dapat membuat perencanaan proyek menjadi lebih baik. Sebuah *prototype* diawal yang

memiliki tata letak tampilan, alur interaksi, status transisi, dan syarat kinerja dapat memberikan kita informasi yang dibutuhkan untuk mengukur penghitungan dan perencanaan kedepannya dengan tingkat kepercayaan diri yang tinggi.

Aktivitas *prototyping* memiliki ikatan historikal yang kuat dalam proses penemuan dan pembuktian dan perlindungan kekayaan intelektual. Meskipun pengertian *prototyping* tidak banyak berubah walaupun sudah lebih dari 90 tahun, *prototyping* berevolusi menjadi aktifitas canggih yang tujuannya tidak hanya untuk pengkonseptualan dan perancangan produk tetapi juga untuk strategi, definisi, spesifikasi, dan perencanaan sebuah produk. Lalu apa saja tahapan-tahapan *prototyping*?



Gambar 2. 1 Siklus Metode *Prototyping*
(Sumber: Hoang, 2022)

2.4.1. Tahap 1 : *Requirements Gathering*

Tahapan awal metode *prototyping*, hampir serupa dengan metode-metode pengembangan perangkat lunak yang lain, yaitu pengumpulan kebutuhan. Kebutuhan sistem secara spesifik ditentukan pada tahapan ini. Pengguna sistem

diberikan pertanyaan mengenai keseluruhan proses untuk selanjutnya digunakan oleh tim pengembang perangkat lunak sebagai acuan dan pengetahuan tentang apa yang diharapkan oleh pengguna terhadap sistem yang baru nantinya. Inti dari pelaksanaan tahap ini adalah untuk mengidentifikasi fungsionalitas dan kebutuhan sistem. Dapat diibaratkan juga seperti menggambar sketsa biru dari sebuah rumah—tanpa pemahaman yang mendasar tentang ruangan, corak, dan tata letak yang diinginkan pemilik rumah, proses konstruksi rumah tersebut menjadi sembarangan.

Jika kita melanjutkan perumpamaan membangun rumah diatas, tahap mengumpulkan kebutuhan sama dengan melakukan tanya jawab dengan pemilik rumah terkait rumah impian mereka. Analisa kebutuhan melibatkan interaksi bersama dengan calon pengguna dari perangkat lunak, dalam hal ini peserta dan penyedia jasa kursus mengemudi. Tim pengembang mengadakan sesi tanya jawab untuk mencatat ekspektasi dan masalah-masalah yang dihadapi pengguna. Proses pengumpulan data yang komprehensif ini tidak hanya mengungkap fitur-fitur yang diinginkan pengguna, namun juga mengidentifikasi tantangan-tantangan potensial yang mungkin akan dihadapi oleh pengguna. Dengan melakukan dokumentasi yang baik pada tahapan awal ini, tim pengembang dijamin akan membangun sistem sebuah sistem yang benar-benar menjembatani kebutuhan pengguna.

Pada tahap ini biasanya terdapat analisa proses bisnis yang saat ini berlangsung, kemudian kebutuhan pengguna, dan fitur-fitur dasar aplikasi.

2.4.2. Tahap 2 : *Quick Design*

Tahap selanjutnya terdiri atas proses yang biasa dikenal dengan desain singkat. Selama tahapan ini, dasar rancangan sistem mulai terbentuk, rancangan ini tidak harus sebuah rancangan yang utuh dan lengkap, melainkan gambaran singkat tentang sistem nantinya. Secara singkat, tujuan dari pelaksanaan tahap ini adalah menerjemahkan kebutuhan pengguna yang teridentifikasi sebelumnya menjadi representasi sistem yang lebih jelas, meskipun sederhana. Ibarat sebuah sketsa awal

dari seorang arsitek—bukan sebuah sketsa biru yang mendetail, tetapi sudah mewakili pokok dari struktur-struktur yang diinginkan dari bangunan nantinya.

Yang perlu ditekankan pada proses ini adalah kecepatan dan efisiensi. Tim pengembang diharap dapat memanfaatkan teknik-teknik desain singkat untuk membangun sebuah *prototype* dasar, sederhana, yang mewakili inti dari fungsionalitas aplikasi hasil terjemahan proses analisa kebutuhan sebelumnya. *Prototype* ini mungkin belum benar-benar berfungsi atau sudah berpenampilan indah, selama memenuhi tujuan pembuatan *prototype*—yaitu memberikan pengguna gambaran sepintas dari alur kerja dan rancangan sistem yang dibuat. Fokus dari tahap ini masih sama dengan tahap sebelumnya, yakni memperoleh informasi mengenai pengalaman pengguna selama ini, permasalahan-permasalahan potensial, dan menjamin *prototype* yang dihasilkan pada tahap ini sejajar dengan fitur yang dianggap pengguna penting.

Dengan memprioritaskan kecepatan dan umpan balik pengguna daripada detail-detail kecil selama tahap ini, tim pengembang bisa mengidentifikasi permasalahan potensial sedini mungkin. Hal ini memungkinkan koreksi arah pengembangan sebelum proses pengembangan sebenarnya dilakukan. Umpan balik yang didapat dari proses pengujian dengan pengguna atas *prototype* kasar yang dihasilkan menjadi bekal untuk memperbaiki rancangan dan memungkinkan perangkat lunak yang dihasilkan nantinya bersifat lebih baik dan berpusat pada pengguna.

Pada tahap ini, biasanya sketsa awal aplikasi beserta alur-alur kerja dari aplikasi mulai terbentuk.

2.4.3. Tahap 3 : *Built Prototype*

Tahap ini adalah tahapan dimana *prototype* yang lebih lengkap dan detail mulai didesain menggunakan data yang didapat dari desain singkat sebelumnya. Alur aplikasi yang dibentuk pada tahap ini biasanya adalah sebagian kecil dari alur aplikasi pada produk akhir nantinya. Tim pengembang mulai memperluas desain

prototype kasar dari tahap sebelumnya menjadi model yang lebih konkrit. *Prototype* ini ditujukan sebagai perwujudan fitur inti dari aplikasi, dimana pada akhirnya pengguna dapat berinteraksi dengan *prototype* ini dan memberikan umpan balik akan kemudahan dan efektifitas penggunaan fitur tersebut.

Kita bisa mengumpamakan tahapan ini seperti miniatur dari pembangunan sebuah gedung. Secara langsung bukan gedung yang asli, tapi para pihak terkait dapat mengetahui tata letak, ruangan-ruangan, dan alur masuk sampai keluar gedung. Tim pengembang memanfaatkan pengetahuan dari tahap sebelumnya untuk meningkatkan kualitas desain dan membuat model yang lebih baik lagi. Fokus dari tahap ini beralih dari *prototype* yang hanya merepresentasikan visual, tata letak, dan sebagainya, dikombinasikan dengan fitur-fitur dasar yang dapat berinteraksi oleh pengguna.

Model sederhana ini memungkinkan evaluasi lebih dalam terhadap kemudahan dan kegunaan aplikasi. Pengguna bisa menguji seberapa jelas atau mudah antarmuka yang ditampilkan, mengidentifikasi kemungkinan-kemungkinan navigasi aplikasi, dan memberikan umpan balik terhadap alur kerja secara keseluruhan. Masukan pengguna yang diterima ini berperan sebagai kritik untuk proses pengembangan nantinya, yang menjamin produk akhir nanti benar-benar memenuhi kebutuhan dan ekspektasi pengguna. Pada tahap ini, iterasi *prototype* pertama dihasilkan, *prototype* yang murni berasal dari penerjemahan kebutuhan yang dilakukan tim pengembang.

2.4.4. Tahap 4 : Customer Evaluation

Pada tahap ini, pengguna diberikan konsep sistem awal untuk dilakukan pengujian. Mengetahui model-model yang berfungsi dengan baik dan model-model yang tidak berfungsi dengan baik akan membantu tim pengembang nanti. Umpan balik pengguna pada tahap ini akan dikumpulkan, kemudian dikirimkan ke tim pengembang.

Tahap ini berfokus seputar validasi pengguna dan peningkatan iteratif. Dengan memiliki *prototype* yang dapat berfungsi pada tahap sebelumnya, tim pengembang mempresentasikannya ke klien, umumnya target pengguna atau pemangku kepentingan aplikasi. Hal ini berperan sebagai sebuah peluang kritis untuk memperoleh umpan balik berharga atas kelebihan dan kekurangan *prototype*.

Dapat kita misalkan seperti menampilkan draf kasar sebuah buku kepada calon pembaca. Penulis ingin mengukur tanggapan pembaca, memahami apa yang berhasil mempengaruhi emosi pembaca, dan mengidentifikasi area-area yang butuh ditingkatkan. Sama dengan apa yang terjadi pada tahap ini, pengujian dengan pengguna terhadap *prototype* yang berfungsi memungkinkan tim pengembang untuk menilai efektivitasnya secara langsung.

Melalui demonstrasi, panduan, dan interaksi pengguna dengan *prototype* tim pengembang secara aktif mengidentifikasi umpan balik klien. Umpan balik ini meliputi baik aspek-aspek positif-fitur-fitur yang disukai pengguna—dan area-area peningkatan—fitur-fitur yang kurang jelas, susah dinavigasi, atau yang tidak memenuhi ekspektasi pengguna. Dengan mengumpulkan dan menganalisa umpan balik ini, tim pengembang memperoleh pengetahuan yang dapat dijadikan panduan untuk memperbaiki *prototype* pada iterasi selanjutnya.

Pada tahap ini, biasanya berisi masukan-masukan dari klien, dalam hal ini, penyedia jasa kursus mengemudi, apakah *prototype* pertama aplikasi sudah memenuhi ekspektasi klien.

2.4.5. Tahap 5 : Review & Updation

Jika pengguna merasa kurang senang dengan *prototype* saat ini, kita harus bisa meningkatkan *prototype* berdasarkan komen-komen dan rekomendasi yang diberikan oleh pengguna. Sampai semua kriteria pengguna terpenuhi, tidak dapat berlanjut ke tahapan selanjutnya. Sebuah sistem akhir diciptakan berdasarkan *prototype* akhir yang diterima sampai pengguna merasa puas dengan *prototype* yang dikembangkan.

Kita ibaratkan seorang pemahat patung menambahkan detail-detail yang memperindah model tanah liat nya berdasarkan umpan balik. Serupa dengan perumpamaan tersebut tim pengembang menganalisa umpan balik yang diterima selama pengujian dengan pengguna. Apabila aspek-aspek *prototype* gagal memenuhi ekspektasi pengguna, tim pengembang tidak akan melanjutkan fase pengembangan aplikasi dari sistem akhir. Mereka harus kembali lagi ke tahap desain singkat dan melakukan perbaikan *prototype* berdasarkan komen-komen dan rekomendasi pengguna. Tahap ini melibatkan proses penambahan detail untuk memenuhi ekspektasi pengguna dan menyederhanakan fitur-fitur yang membingungkan, meningkatkan antarmuka pengguna, atau mengkolaborasikan fitur-fitur baru yang dianggap berharga menurut pengguna.

Proses iteratif ini akan terus berlanjut sampai pengguna merasa puas dengan *prototype*. *Prototype* akhir ini, yang mewakili pokok-pokok umpan balik dan perbaikan pengguna berperan sebagai sketsa biru untuk proses pemrograman sistem akhir. Dengan memprioritaskan umpan balik pengguna selama siklus ini, model *prototype* menjamin bahwa produk akhir tidak hanya berfungsi, tapi benar-benar berfokus dengan pengguna dan menjawab kebutuhan yang diidentifikasi pada tahap analisa kebutuhan sebelumnya.

Pada tahap ini, biasanya *prototype* hasil perbaikan dari tahap sebelumnya akan ditampilkan dan umpan balik dari pengguna terhadap *prototype*, dan yang terakhir adalah hasil keputusan terkait apakah pengguna sudah puas dengan hasil *prototype*.

2.4.6. Tahap 6 : Design

Setelah menyelesaikan 4 tahap iteratif sebelumnya, tim pengembang selanjutnya akan menetapkan desain akhir. Tahap ini dilakukan untuk menerjemahkan *prototype* yang divalidasi oleh pengguna ke sketsa biru yang lebih mendetail dan komprehensif untuk proses pemrograman. Tidak sama dengan proses desain singkat sebelumnya, tujuan tahap ini beralih kepada perencanaan dan

dokumentasi. Beberapa diagram-diagram UML akan dirancang untuk menampilkan fitur-fitur, struktur, dan perilaku aplikasi yang lebih jelas dan terstandarisasi.

Dengan merancang diagram-diagram UML, tim pengembang merumuskan *roadmap* komprehensif untuk pengembangan aplikasi nantinya. Diagram-diagram ini berperan sebagai alat untuk berkomunikasi, menjamin semua orang yang terlibat dalam proses pengembangan aplikasi memiliki pemahaman yang sama akan fitur-fitur, komponen, dan bagaimana aplikasi berinteraksi. Tahap desain akhir ini dapat menjembatani antara model *prototype* yang berfokus pada pengguna dengan proses pengembangan sesungguhnya. Diagram-diagram UML ini akan diterjemahkan menjadi fitur-fitur *prototype* yang tervalidasi menjadi bahasa teknis, membuka jalan bagi *programmer* untuk mulai melakukan pemrograman dan membangun sistem akhir.

Seperti yang disebutkan diatas, pada tahap ini akan menghasilkan *use case diagram*, *use case scenario*, *activity diagram*, dan *class diagram*.

2.4.7. Tahap 7 : Development

Dengan tahap desain akhir sebelumnya sudah menghasilkan diagram-diagram UML, tahap pemrograman aplikasi dapat dimulai dengan lebih lancar. *Programmer* memanfaatkan sketsa biru yang komprehensif yang dihasilkan dari tahap sebelumnya yang harus diterjemahkan menjadi fitur-fitur yang tervalidasi pengguna menjadi aplikasi yang dapat bekerja. Tahap ini melibatkan proses pemrograman, mengintegrasikan macam-macam komponen, dan secara ketat melakukan pengujian terhadap sistem untuk menjamin aplikasi yang dihasilkan menaati spesifikasi-spesifikasi yang dirumuskan pada diagram-diagram UML sebelumnya. Proses pengembangan yang bersifat iteratif mendukung perbaikan-perbaikan *bug* berdasarkan hasil pengujian.

Setelah tahap pemrograman mencapai titik stabilitas dan fungsional, sistem selanjutnya diluncurkan ke lingkup produksi. Tahap ini melibatkan perencanaan

dan eksekusi dengan cermat demi meminimalisir permasalahan yang mungkin dialami oleh pengguna nantinya. Dengan selesainya tahap pemrograman, tidak serta merta menandakan selesainya siklus pengembangan perangkat lunak. Pengawasan yang terus berlanjut dan umpan balik pengguna terus dianggap krusial. *Programmer* secara aktif mendeteksi *bug* atau masalah-masalah yang muncul, dan selama sistem dapat ditingkatkan berdasarkan saran-saran pengguna dan kebutuhan yang berubah-ubah. Putaran umpan balik yang terus berlangsung ini menjamin bahwa aplikasi tetap relevan, ramah pengguna, dan secara efektif memenuhi kebutuhan pengguna dari waktu ke waktu.

Pada tahap ini, aplikasi sudah berbentuk dan dapat diakses pada web masing-masing pengguna, untuk selanjutnya dilakukan pengujian.

2.4.8. Tahap 8 : Test

Tahap pengujian aplikasi pada metode pengembangan perangkat lunak *prototype* memiliki peran krusial untuk menjamin kualitas dan fungsionalitas aplikasi nantinya. Bukan hanya untuk mengidentifikasi *bug*, namun, juga untuk mengevaluasi kinerja sistem terhadap kriteria yang sudah dirumuskan. Pengujian mencakup berbagai macam aspek:

- **Pengujian Fitur:** Pengujian ini menjamin seluruh fitur yang dirancang pada diagram-diagram UML dan kebutuhan pengguna sudah diimplementasikan dengan benar.
- **Pengujian Kemudahan Penggunaan:** Pengujian ini menilai seberapa ramah sistem yang dihasilkan terhadap pengguna.

Dengan mengimplementasikan strategi pengujian yang komprehensif yang menjawab macam-macam aspek, tim pengembang dapat mengidentifikasi dan memperbaiki masalah sebelum sistem digunakan untuk masyarakat luas. Pengujian yang dilakukan ini menjaga kualitas produk akhir dan menjamin produk yang dihasilkan memenuhi standar tinggi yang dirumuskan selama proses *prototyping*.

2.4.9. Tahap 9 : *Maintenance*

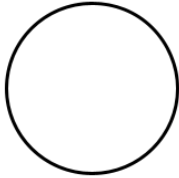
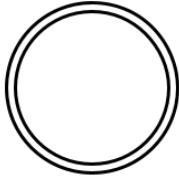
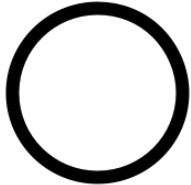

Setelah dilakukan pengujian secara komprehensif, demi mengurangi *lag* dan menghindari kerusakan yang parah, sistem harus dipelihara secara berkala. Proses ini bukan merupakan pendekatan reaktif untuk menunggu masalah-masalah muncul; melainkan strategi proaktif yang berfokus untuk mencegah masalah sebelum masalah tersebut mengganggu proses operasional pada sistem. Berikut adalah pemeliharaan sistem umumnya bekerja:

- **Mengidentifikasi dan Menjawab Masalah Potensial:** Memelihara aktivitas melibatkan pemeriksaan rutin dari kesehatan sistem, metrik kinerja sistem, dan kerentanan keamanan sistem. Pemeriksaan ini mampu mengungkap masalah-masalah potensial sedini mungkin, melakukan intervensi dengan tepat waktu sebelum masalah tersebut berkembang menjadi gangguan yang lebih besar.
- **Menerapkan Pembaruan:** Pembaruan perangkat lunak seringkali menjawab kerentanan dan *bug* yang diketahui. Penerapan secara berkala dari pembaruan ini menjamin sistem tetap aman dan terlindungi dari ancaman-ancaman yang muncul.
- **Mengoptimisasi Kinerja:** Secara berkala, sistem mengakumulasi penurunan kinerja data dan pengalaman. Pemeliharaan berkala meliputi tugas-tugas seperti optimalisasi data dan pembersihan sistem, menjamin sistem berjalan dengan lancar dan efisien.
- **Mencadangkan Pengujian dan Rencana Pemulihan Bencana:** Secara berkala mencadangkan prosedur pengujian dan rencana pemulihan bencana memverifikasi efektivitas rencana tersebut. Dengan melakukan hal ini, menjamin sistem mampu memulihkan fungsionalitasnya jika terjadi kejadian yang tidak terduga, meminimalisir hambatan dan kehilangan data.

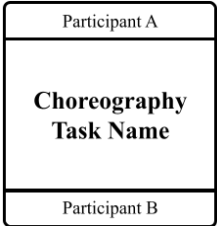
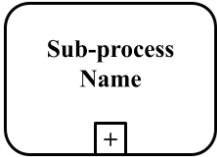
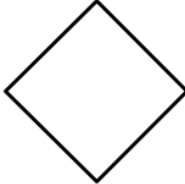

2.5. Business Process Model and Notation (BPMN)

Business Process Model and Notation adalah standar khusus untuk digunakan sebagai benchmark untuk pemodelan proses bisnis yang menghasilkan notasi grafis untuk memvisualisasikan proses bisnis (Firdaus et al., 2022). Notasi ini kemudian dapat kami konversikan menjadi *user flow* atau alur tugas yang dibutuhkan dalam melakukan proses *prototyping*. Notasi-notasi pada BPMN diantaranya adalah:

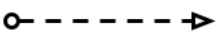
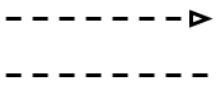
Tabel 2. 1 Notasi-notasi BPMN
(Sumber: OMG, 2011)

No.	Notasi	Nama	Deskripsi
1		<i>Start Event</i>	Sesuai dengan nama notasinya, notasi ini mengindikasikan dimana sebuah rangkaian proses dimulai.
2		<i>Intermediate Event</i>	Notasi ini digunakan ketika terjadi sebuah proses diantara rangkaian proses.
3		<i>End Event</i>	Sebagaimana nama dari notasi disamping, notasi ini mengindikasikan akhir dari rangkaian proses.
4		<i>Task</i>	Sebuah tugas adalah aktivitas atomik yang dimasukkan di rangkaian proses bisnis. Sebuah tugas digunakan ketika sebuah pekerjaan dalam proses bisnis sudah tidak bisa lagi dipecah menjadi tugas yang lebih kecil

Tabel 2. 2 Notasi-notasi BPMN (Lanjutan-1)
(Sumber: OMG, 2011)

No.	Notasi	Nama	Deskripsi
			lagi.
5		<i>Choreography Task</i>	Notasi ini merepresentasikan sebuah tugas yang melibatkan dua partisipan untuk menyelesaikannya.
6		<i>Sub-process</i>	Sebuah sub-proses adalah gabungan aktivitas yang dimasukkan pada sebuah proses.
7		<i>Gateway</i>	Sebuah gateway tidak sama dengan notasi kondisional pada flowchart, notasi ini digunakan untuk mengontrol pemecahan dan pertemuan pada sebuah rangkaian proses bisnis. Banyak macam dari gateway, lebih lanjutnya (OMG, 2011).
8		<i>Sequence Flow</i>	Notasi ini mengindikasikan sebuah alur yang tidak dimulai dari <i>Intermediate Event</i> yang terhubung ke sebuah tugas / sub-proses.

Tabel 2. 3 Notasi-notasi BPMN (Lanjutan-1)
(Sumber: OMG, 2011)

No.	Notasi	Nama	Deskripsi
9		<i>Message Flow</i>	Notasi ini digunakan untuk menunjukkan alur informasi antara dua partisipan yang bersiap untuk mengirim dan menerima informasi tersebut.
10		<i>Association</i>	Notasi ini digunakan untuk menghubungkan informasi dan artifak terkait teks anotasi dan artifak lain yang terasosiasikan dengan elemen grafik lainnya.

2.6. *Unified Modeling Language (UML)*

Menurut buku dengan judul “*The Unified Modeling Language Reference Manual*” UML adalah tujuan umum dari bahasa pemodelan visual yang digunakan untuk menspesifikasi, memvisualisasikan, menyusun, dan mendokumentasi hal-hal yang terkait sistem perangkat lunak. UML mencatat semua keputusan dan pemahaman mengenai sistem-sistem yang wajib dibangun nantinya. Selain itu, UML digunakan untuk memahami, merancang, mengeksplorasi, mengkonfigurasi, memelihara, dan mengontrol informasi terkait sistem yang dikembangkan. Namun, UML bukan merupakan bahasa pemrograman, melainkan sebuah *tools* yang dapat menginspirasi pembuatan program yang selanjutnya bisa dikembangkan menggunakan bahasa pemrograman lainnya.

Tujuan dari UML sendiri diantaranya adalah UML dibuat sebagai tujuan utama untuk bahasa pemodelan dimana semua pelaku pengembangan perangkat lunak bisa menggunakannya. Bertujuan untuk mencakup konsep-konsep metode

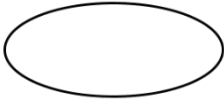
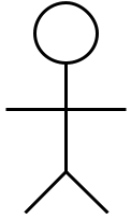
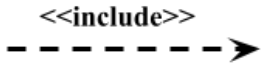
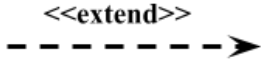
ternama yang nantinya mereka gunakan sebagai bahasa pemodelan. UML tidak dimaksudkan untuk dijadikan metode pengembangan yang lengkap, sebab UML tidak memiliki langkah-langkah mendetail tentang proses pengembangan perangkat lunak. Sekali lagi ditegaskan oleh (Jacobson et al., 2021) UML mencakup konsep-konsep yang dianggap penting untuk mendukung sebuah proses iteratif yang modern berdasarkan penerapan dengan arsitektur yang kuat untuk menyelesaikan kebutuhan berdasarkan masalah-masalah dan kasus-kasus yang dialami pengguna.

Yang terakhir, inti dari tujuan UML adalah pemodelan yang sesederhana mungkin selama masih mampu memenuhi syarat-syarat pemodelan sistem praktis secara penuh yang nantinya akan dibuat. Teknik-teknik pemodelan dari UML yang kita kenal diantaranya adalah *Use Case Diagram*, seperti yang kami jelaskan sebelumnya, kemudian terdapat *Activity Diagram*, *Class Diagram*, *Sequence Diagram*, kemudian banyak pemodelan-pemodelan lain yang kurang populer seperti *Statechart Diagram*, *Collaboration Diagram*, *Component Diagram*, *Deployment Diagram*, *Extensibility Construct*, dan pemodelan-pemodelan yang lain.

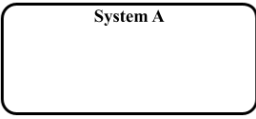
2.7. *Use Case Diagram*

Menurut (Simanullang et al., 2021) pada jurnalnya dengan judul Sistem Informasi Pemesanan Menu Makanan Pada Rm Sedep Roso Rantauprapat Berbasis Web, peneliti sempat sedikit menjelaskan tentang istilah ini. *Use Case Diagram* adalah suatu pola atau gambaran yang menunjukkan kelakuan atau kebiasaan sistem. Sedangkan (Setiawansyah et al., 2022) menjelaskan bahwa *Use Case Diagram* adalah sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem. Berikut adalah simbol-simbol yang digunakan pada *Use Case Diagram*:

Tabel 2. 4 Notasi-notasi *Use Case Diagram*
(Sumber: OMG, 2011)

No.	Notasi	Nama	Deskripsi
1		<i>Use Cases</i>	Merepresentasikan fungsionalitas sebuah sistem, terkadang juga tujuan akhir dari aktor. Simbol ini selalu berada di dalam <i>Boundary Box</i> .
2		<i>Actors</i>	Aktor yang berinteraksi dengan sistem biasanya memicu <i>use case</i> . Simbol ini selalu berada di luar <i>Boundary Box</i> .
3	 	<i>Association</i>	<p>Arah panah digunakan untuk menandakan sebuah hubungan antara aktor dan <i>use case</i> atau antara dua <i>use case</i>.</p> <p>Panah dengan anotasi <<extend>> menandakan bahwa sebuah <i>use case</i> mungkin mengadopsi perilaku dari <i>use case</i> lain.</p> <p>Panah dengan anotasi <<include>> menandakan bahwa sebuah <i>use case</i> menggunakan fungsionalitas <i>use case</i> lain.</p>




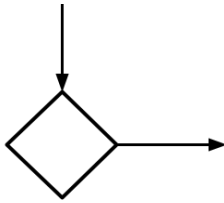
Tabel 2. 5 Notasi-notasi *Use Case Diagram* (Lanjutan-1)
(Sumber: OMG, 2011)

No.	Notasi	Nama	Deskripsi
4		Boundary Box	Simbol ini mengindikasikan batas lingkup sistem.

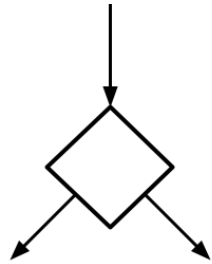
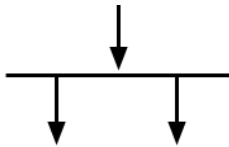
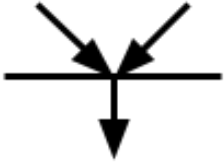
2.8. *Activity Diagram*

(Jacobson et al., 2021) menjelaskan *activity diagram* sebagai perwujudan khusus dari kondisi mesin yang ditujukan untuk memodel komputasi dan alur kerja sistem. Kondisi yang digambarkan pada *activity graph* mewakili kondisi eksekusi komputasi yang dilakukan, bukan kondisi suatu objek secara spesifik. Umumnya, *activity graph* berasumsi bahwa komputasi yang terjadi tidak dipengaruhi oleh kejadian eksternal. *Activity diagram* biasanya berisi percabangan, lebih sering lagi percabangan kendali yang terbagi dua, dimana selanjutnya percabangan tersebut berjalan atau diproses bersamaan. Alur yang diproses secara bersamaan ini merepresentasikan aktivitas-aktivitas yang dapat dikerjakan oleh objek-objek atau orang-orang berbeda secara bersamaan pada sebuah organisasi. Seringkali, kejadian yang terjadi bersamaan ini muncul dari adanya agregasi, dimana objek memiliki proses yang harus dieksekusi secara bersamaan sendiri. Aktivitas yang terjadi secara bersamaan ini sebetulnya dapat dieksekusi secara bersamaan atau satu persatu. *Activity graph* sama halnya dengan *flowchart* yang sering kita kenal, perbedaan yang mendasar adalah *activity graph* memungkinkan terjadinya kendali proses secara bersamaan, selain kontrol sekuensial saja. Berikut adalah notasi-notasi yang ada pada *activity diagram*.

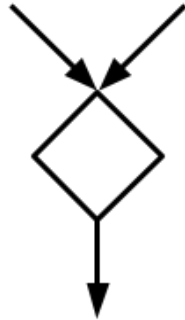
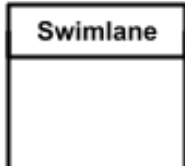
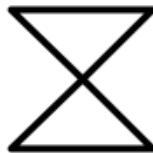

Tabel 2. 6 Notasi-notasi *Activity Diagram*
(Sumber: GeeksforGeeks, 2024)

No.	Notasi	Nama	Deskripsi
1		<i>Initial state</i>	Sebuah rangkaian proses hanya membutuhkan satu kondisi awal kecuali kita menggambarkan aktivitas yang terjadi didalam aktivitas.
2		<i>Action atau Activity State</i>	Sebuah aktivitas merepresentasikan eksekusi dari sebuah aksi atau objek oleh objek.
3		<i>Action Flow atau Control Flow</i>	Alur aksi atau alur kontrol ini biasa digunakan untuk menunjukkan transisi dari satu kondisi aktivitas ke kondisi aktivitas lain
4		<i>Decision Node dan Branching</i>	Ketika kita akan membuat keputusan sebelum melanjutkan alur kontrol, kita bisa gunakan simpul keputusan.

Tabel 2. 7 Notasi-notasi *Activity Diagram* (Lanjutan-1)
(Sumber: GeeksforGeeks, 2024)

No.	Notasi	Nama	Deskripsi
5		<i>Guard</i>	Sebuah pelindung merujuk pada sebuah pernyataan yang ditulis disamping arah panah simpul keputusan. Pelindung membantu kita mengetahui batasan dan kondisi-kondisi yang menentukan alur sebuah proses
6		<i>Fork</i>	Simpul garpu digunakan untuk menggambarkan dua aktivitas yang terjadi secara bersamaan. Aktivitas yang berada dibawah simpul ini akan di eksekusi secara bersamaan.
7		<i>Join</i>	Simpul gabung digunakan untuk melebur dua aktivitas yang sebelumnya diproses secara bersamaan menjadi satu proses utuh kembali.

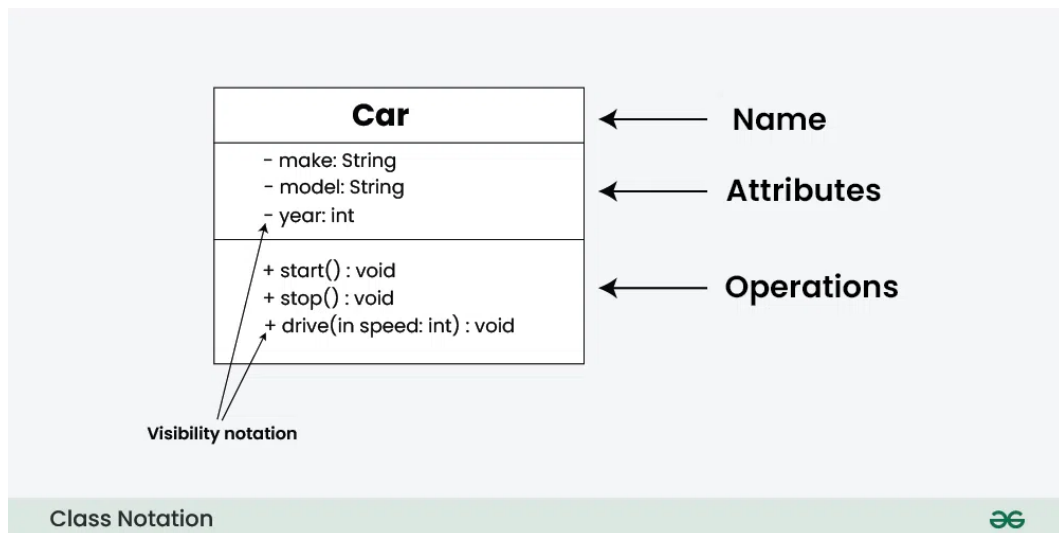
Tabel 2. 8 Notasi-notasi *Activity Diagram* (Lanjutan-2)
(Sumber: GeeksforGeeks, 2024)

No.	Notasi	Nama	Deskripsi
8		Merge atau Merge Event	Penggabungan aktivitas ini dapat dilakukan apabila kita ingin menggabungkan dua atau lebih aktivitas yang sebelumnya tidak diproses secara bersamaan, namun harus diselesaikan sebelum aktivitas selanjutnya dapat dieksekusi.
9		Swimlanes	<i>Swimlane</i> digunakan untuk mengelompokkan aktivitas-aktivitas yang dieksekusi oleh objek atau orang yang sama.
10		Time Event	Notasi ini merujuk pada aktivitas yang membutuhkan waktu lebih lama untuk diselesaikan.
11		Final State atau End State	Kondisi akhir digunakan untuk menandakan akhir dari rangkaian proses.

2.9. Class Diagram

(Oktriwina, 2021) menjelaskan *class diagram* sebagai salah satu diagram struktur statis yang dimiliki UML dimana *class diagram* menggambarkan struktur sistem dengan menunjukkan sistem *class*, atribut-atribut yang dimiliki, metode-

metode yang dapat dieksekusi, dan hubungan antar objek. (Jacobson et al., 2021) menjelaskan *class* itu sendiri adalah penjelasan dari konsep wewenang aplikasi atau solusi dari aplikasi. *Class* dapat dideskripsikan dari berbagai tingkatan presisi dan seberapa konkrit kita ingin mendeskripsikannya. Pada awal tahapan pengembangan perangkat lunak, *class diagram* seringkali berisi aspek-aspek logika dari masalah yang ingin diatasi. Sedangkan pada akhir tahapan pengembangan, *class diagram* dapat diisi keputusan-keputusan desain dan detail-detail implementasi di dalamnya. Berikut adalah contoh notasi sebuah *class*.



Gambar 2. 2 Notasi *Class Diagram*
(Sumber: GeeksforGeeks, 2024)

- **Nama Class** : Biasanya ditulis di bagian atas kotak *class* dan ditulis tebal dengan perataan tengah.
- **Atribut** : Sering disebut dengan properti, merepresentasikan anggota-anggota data dari sebuah *class*. Biasanya berada di bawah nama *class* dan disertai oleh notasi visibilitas (contoh: [+] untuk *public*, [-] untuk *private*) dan tipe data.
- **Methods** : Dikenal juga sebagai fungsi atau operasi, merepresentasikan perilaku atau fungsionalitas sebuah *class*. Komponen ini berada di bawah atribut dan memiliki notasi visibilitas, jenis pengembalian, dan parameter-parameter setiap *methods*.

- **Notasi visibilitas** : Mengindikasikan tingkat aksesibilitas dari atribut atau *methods*. Notasi visibilitas umumnya ada 4 :
 - **[+]** untuk *public* (dapat diakses oleh semua *class*)
 - **[-]** untuk *private* (dapat diakses didalam *class* itu sendiri)
 - **[#]** untuk *protected* (dapat diakses oleh *subclass* dari *class* tersebut)
 - **[~]** untuk *package* atau *default visibility* (dapat diakses pada *class* dengan *package* yang sama)

2.10. *Hyper Text Markup Language (HTML)*

HTML adalah salah satu istilah pemrograman yang paling dikenal oleh masyarakat umum, namun, banyak perdebatan yang mengatakan bahwa sebenarnya HTML bukanlah sebuah bahasa pemrograman. Ada juga beberapa orang yang beranggapan bahwa karena dalam menulis HTML diperlukan setidaknya pemahaman dasar tentang pemrograman, maka HTML dianggap sebagai bahasa pemrograman. Mengacu dari jurnal yang ditulis oleh (Sari et al., 2022) HTML merupakan salah satu bahasa pemrograman yang digunakan untuk membuat website. HTML biasa ditulis untuk membantu perancangan struktur dasar halaman website atau bisa juga dianggap sebagai pondasi awal untuk menyusun kerangka halaman website secara lebih terstruktur sebelum masuk ke tahap desain dan fungsionalitas.

Terdapat pihak yang menyatakan bahwa sesungguhnya HTML merupakan sebuah bahasa *markup*, disebut demikian karena penulisan HTML didominasi oleh penulisan *tag* atau tulisan di dalam tanda kurung lancip (<html>) dimana masing-masing *tag* yang kita tulis nantinya akan dibaca kemudian dikonversi menjadi tampilan website yang kita baca setiap harinya. Singkatnya, HTML hanyalah kumpulan dari berbagai macam *tag* (seperti: <h1>, <p>, <div>, dll.) yang bisa kita gunakan untuk membangun sebuah halaman website dan masing-masing tag tersebut hanyalah sebuah cara untuk kita menjelaskan kepada mesin konten apa yang ingin kita tampilkan.

2.11. *Cascading Style Sheets (CSS)*

Pada era modern saat ini, hampir tidak bisa kita temui rangkaian kode HTML tanpa dilengkapi CSS. Menurut (Sari et al., 2022) CSS adalah bahasa pemrograman yang ditujukan untuk memberikan modifikasi tampilan elemen-elemen web seperti *font*, *outline*, *background*, menyesuaikan tampilan website dengan ukuran layar, dan sebagainya. Jika HTML digunakan untuk menempatkan konten-konten apa saja yang ingin ditampilkan pada sebuah halaman web, CSS digunakan untuk memberikan pemahaman kepada mesin untuk melakukan modifikasi terhadap tampilan elemen dan penataan tata letak lebih lanjut.

Karena sejatinya, HTML tidak dirancang untuk menentukan aspek visual pada sebuah desain website. Sebab, fokus utama dari HTML adalah membagi struktur sebuah halaman website. Oleh karena itu, dikenalkan skrip “pendamping” untuk memperindah tag-tag HTML yaitu CSS. Selain itu, tujuan penggunaan CSS adalah untuk memberikan kesan konsisten di seluruh website.

2.12. *Tailwind CSS Framework*

(Somi, M., 2023) dalam jurnalnya menjelaskan bahwa *Framework Tailwind* CSS adalah *framework* CSS yang mengutamakan penggunaan kelas utilitas yang paling populer dan bertujuan untuk membangun tampilan antarmuka khusus dengan cepat dan mudah. Maksudnya, berbeda dengan *Bootstrap* yang tergolong *framework UI kits*, *tailwind* tidak menyediakan komponen-komponen siap pakai. *Tailwind* tidak mempunyai tema bawaan. Dengan *tailwind*, kita memberikan *style* dengan mengetikkan kelas-kelas yang sudah ditentukan sebelumnya ke kodingan HTML yang kita kerjakan.

Lebih lanjut lagi, (Somi, M., 2023) menjelaskan bahwa dengan menggunakan *tailwind*, memberikan kita kemampuan untuk mempercepat proses pemrograman tanpa kita harus menulis kode CSS di *file* lain melainkan, menulisnya secara bersamaan di kode markup HTML. Selain itu, *tailwind* memberikan kita kemampuan untuk melakukan kustomisasi secara penuh sesuai keinginan kita. Efek

samping dari menggunakan *tailwind* adalah kode HTML yang dihasilkan akan jauh lebih panjang

2.13. Javascript

Halaman website yang dihasilkan dari hanya menggunakan bahasa HTML & CSS cenderung statis dan kurang menarik. Untuk membuat tampilan yang lebih dinamis diciptakan sebuah bahasa pemrograman baru demi mengatasi kekurangan ini, yakni *Javascript*. Sebagai referensi, (Noviantoro et al., 2022) menjelaskan bahwa *Javascript* adalah salah satu bahasa pemrograman yang digunakan untuk dapat berjalan di web browser. Pada awal pengembangan bahasa pemrograman ini sempat disebut dengan nama *Mocha*, kemudian berubah penamaannya menjadi *Live-Script*, dimana ketika masa rilis, diubah lagi menjadi *Javascript*. Lebih jauh lagi dijelaskan bahwa *Javascript* adalah *script* program berbasis *client* yang dieksekusi oleh browser sehingga membuat halaman web melakukan tugas-tugas tambahan yang tidak bisa dilakukan hanya dengan memanfaatkan HTML biasa. Selain alasan yang kami sebut diatas, beberapa interaksi yang ingin kami munculkan pada aplikasi ini tidak dapat diselesaikan hanya dengan menggunakan HTML dan CSS.

2.14. Framework PHP Laravel

Sebelum kita lebih jauh membahas tentang *framework Laravel*, akan kami jelaskan secara singkat apa yang dimaksud dengan PHP, karena istilah ini juga bukan bagian yang terpisahkan dari perancangan dan pengembangan sistem informasi atau perangkat lunak berbasis web. PHP atau *Hypertext Preprocessor* sebagaimana yang dijelaskan (Adrianto S., 2021) adalah sebuah bahasa pemrograman untuk membuat web yang bersifat *server-side scripting*, PHP memungkinkan untuk membuat halaman web bersifat dinamis. Selain itu, PHP membutuhkan *Database Management System* (DBMS) untuk dijalankan secara bersamaan. DBMS yang paling populer di kalangan pelajar pemrograman salah

satunya adalah MySQL, namun, PHP juga mendukung DBMS lain seperti Oracle, Microsoft Access, Interbase, D-Base, PostgreSQL, dan DBMS yang lainnya.

Satu hal yang mungkin menjadi pertanyaan adalah apabila PHP dikenal dengan *Hypertext Preprocessor*, lalu apa kepanjangan PHP? Menurut tulisan yang diterbitkan melalui web sekawanstudio.com yang ditulis oleh (Miranda R.A., 2023) Pada tahun 1994, ketika Rasmus Lerdorf pertama kali menemukan *hypertext preprocessor*, beliau menggunakannya untuk memantau jumlah pengunjung atau yang sering kita sebut dengan *traffic website* dari halaman web pribadi nya atau dalam bahasa inggris dikenal dengan *Personal Home Page*. Alasan tersebutlah yang menyebabkan bahasa pemrograman ini dijuluki sebagai PHP.

Menurut penjelasan dari (Subecs, 2021) Laravel merupakan *framework* PHP yang paling sering digunakan untuk *programmer* pemula dan berpengalaman. Laravel dianggap mampu mengurangi durasi pengembangan sistem perangkat lunak serta mempersiapkan pasar dengan metode PHP berorientasi objek yang lebih modern. *Syntax-syntax* ekspresif dan *function-function* modern yang dimiliki Laravel disukai oleh para *programmer* yang ingin mengembangkan web atau aplikasi yang lebih kompleks. Dengan menggunakan *framework* ini diyakini dapat mempermudah proses pengembangan karena Laravel menggunakan sistem paket modular dimana modul-modul yang disediakan saling terkait satu sama lain, dimana kita bisa mengembangkan sistem perangkat lunak yang lebih luas lagi. *Framework* ini memberikan kita jalan pintas yang memungkinkan *programmer* berkonsentrasi terhadap masalah-masalah yang lebih penting.

Lebih lanjut (Subecs, 2021) menyebutkan fitur-fitur atau kelebihan-kelebihan yang dimiliki Laravel yang menjadikan sebab mengapa banyak *programmer* menyukai *framework* ini. Kelebihan-kelebihan tersebut diantaranya adalah:

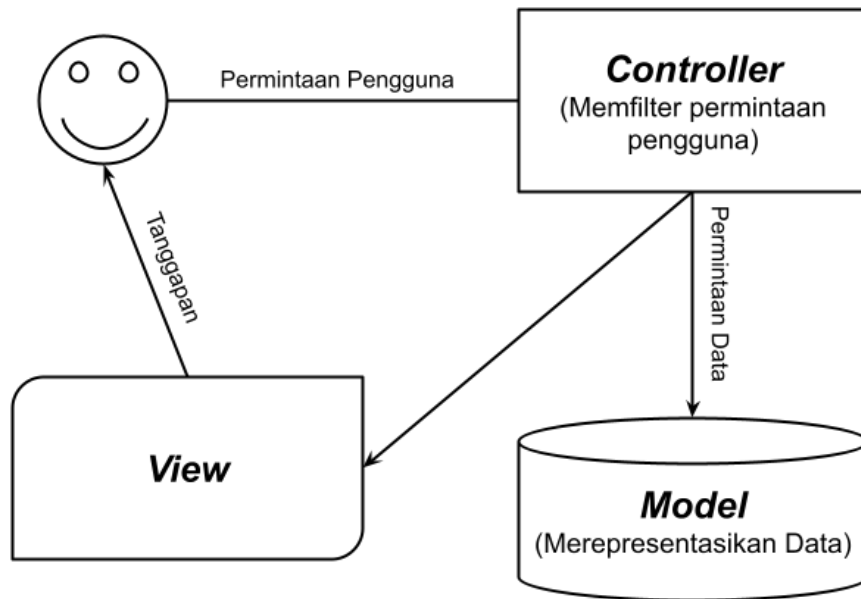
2.14.1. Routing

Fungsi utama dari sebuah *framework* sistem perangkat lunak adalah untuk menerima permintaan dari pengguna dan mengirim tanggapan mereka biasanya

melalui HTTP(S). Artinya bahwa tugas pertama selama proses penerapan ini ialah menentukan rute-rute yang dibutuhkan. Tanpa rute, sebuah sistem tidak akan mampu mencapai pengguna. Rute itu sendiri pada dasarnya merupakan sebuah URI (*Uniform Resource Identifier*) yang memungkinkan komunikasi dengan dunia luar dengan alamat URL yang diketahui. Pada dasarnya *client* mengirimkan sebuah permintaan untuk mendapatkan rute yang ditentukan, sesuai dengan kebutuhan, meneruskannya ke sebuah *controller* yang memproses permintaan tersebut. Folder */routes* yang berada di dalam folder *root* proyek yang berisi *file-file* rute tersebut.

2.14.2. Arsitektur MVC

Laravel dibangun diatas pondasi arsitektur MVC. *Model-View-Controller* (MVC) merupakan sebuah pola perancangan yang mencakup tiga bagian utama: *model*, *view*, dan *controller*. Komponen-komponen ini sudah dikonfigurasi untuk menangani aspek-aspek pengembangan khusus dari aplikasi. Komponen *View* ditujukan untuk menangani logika tampilan pengguna. Komponen *Controller* menerima input data dan memproses input tersebut. Komponen *Controller* ini bekerja sebagai antarmuka antara komponen *model* dan *view*. Sedangkan komponen *Model* adalah logika yang memiliki koneksi dengan data-data yang terkait user. Komponen ini merupakan komponen utama dari pola arsitektur MVC dan merepresentasikan data-data yang ditransaksikan antara *view* dan *controller*.



Gambar 2. 3 Skema MVC
(Sumber: Subecz, Z., 2021)

2.14.3. Views dan Templates

Views bertanggung jawab untuk menampilkan tanggapan dari *controller* sesuai dengan format yang tepat, biasanya sebagai halaman website. *Views* dan relasinya dengan *Controller* ini bisa dikembangkan dengan mudah menggunakan bahasa *Blade template* atau lebih sederhana lagi dengan skrip-skrip PHP biasa. *View* adalah objek yang paling umum kembali ke rute-rute. *View* mendapatkan data dari rute-rute atau *controller* dan menyisipkannya ke sebuah template, sehingga mereka membantu satu sama lain untuk memisahkan logika bisnis dari logika presentasi.

2.14.4. Controller

Pada sebuah *framework MVC*, *Controller* sebagaimana namanya, mengontrol alur data antara dataset-dataset dan *views*. Hal ini merupakan mekanisme yang membawa pengguna-pengguna ke lapisan presentasi. *Controller* menerima permintaan, memproses mereka dan mengirimkan tanggapan yang sesuai. *Controller* menangani tugas-tugas seperti menarik data dari *database*, menangani

penerimaan data dari form-form dan menyimpan data ke *database*. *Controller* biasanya berada di folder *app/http/controllers*.

2.14.5. Berinteraksi dengan *database*

Ada 3 proses interaksi dengan *database* yang terbaik dari *framework* Laravel menurut (Subecs, 2021):

- *Migration*

Migrasi adalah skrip-skrip PHP yang digunakan untuk memanipulasi struktur dan konten *database*. Seorang *programmer* yang bekerja dalam sebuah tim diharapkan untuk mampu dengan baik mensinkronisasi tugas-tugas *database* dengan rekan-rekan *programmer* lain. Dalam hal ini, migrasi beroperasi sebagai sebuah *version control*. Migrasi-migrasi ini dikirimkan bersama dengan detail waktu migrasi, sehingga migrasi ini dieksekusi berdasarkan urutan yang benar. Dengan memanfaatkan migrasi ini memungkinkan *database* memiliki struktur keadaan yang konsisten.

- *Database Seeding*

Dalam *framework* Laravel, kita bisa mengisi *database* dengan pembantu pengisian *database*, daripada melakukannya dengan manual. Dengan melakukan pengisian ini, kita bisa meng-*upload* data pengujian ke *database* menggunakan sebuah cara sederhana. Kita mampu mencari *file-file* pengisian ini di folder *database/seeders*.

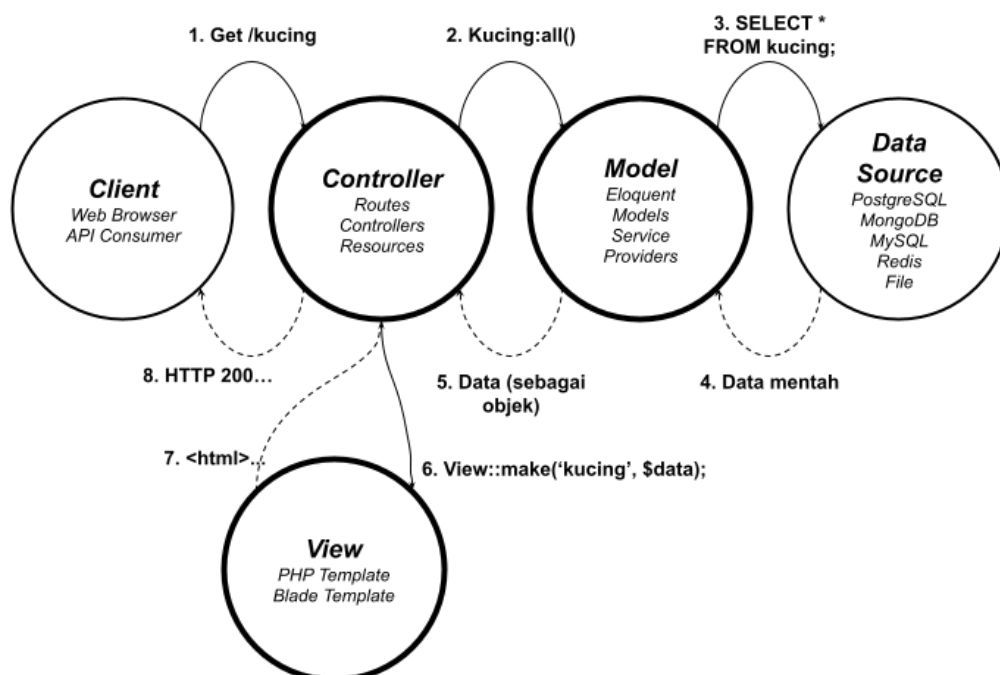
- *Query Builder*

Laravel memberikan kita *query builder* yang mudah digunakan yang ditujukan untuk membuat dan mengeksekusi *query-query database*. *Programmer* dapat menggunakannya untuk bermacam-macam operasi *database* dan bekerja dengan sempurna dengan sistem-sistem *database* yang didukung Laravel.

2.14.6. Model dan Eloquent

Model seringkali dihubungkan dengan sumber-sumber informasi aplikasi dan seringkali digunakan bersama dengan rekaman-rekaman *database*. *Model* ini merupakan kelas-kelas yang merepresentasikan entitas-entitas di dalam aplikasi, seperti pengguna, artikel berita, atau *event-event*. Laravel meluncurkan sebuah fitur yang disebut dengan *Eloquent*, yang merupakan pemetaan hubungan antar objek yang efisien dimana para *programmer* dapat menentukan entitas, kemudian memasukkannya ke tabel-tabel *database* terkait dan memanfaatkan metode-metode PHP untuk mengeksekusi *entry* entitas-entitas tersebut, daripada melakukan *statement* SQL secara mentah. Dengan fitur ini, kita bisa menjalankan *query* database ORM dengan efisien, tanpa menuliskan instruksi-instruksi SQL apapun. *Eloquent* berjalan sebagai sebuah lapisan-model pada aplikasi yang dikembangkan.

Berikut adalah ilustrasi bagaimana interaksi yang terjadi dari masing-masing komponen ketika diberikan perintah untuk menampilkan semua data yang ada pada tabel kucing.



Gambar 2. 4 Ilustrasi *Model* dan *Eloquent*
(Sumber: Subecz, Z., 2021)

Kita ketahui *client* (melalui *web browser* atau *API Consumer*) ingin menampilkan semua data yang tersimpan pada tabel kucing. Perintah ini diteruskan menuju *Controller* terlebih dahulu untuk diproses permintaan data *client* kemudian menentukan *routes* sumber pengambilan data. Selanjutnya *Model* menangkap perintah pengambilan data dari tabel kucing dan mengirimkannya ke sumber data. Selanjutnya, data yang diambil dari sumber data ini dikembalikan lagi ke *Model* untuk diperiksa apakah data yang diambil hasilnya sesuai dengan perintah yang diberikan, selanjutnya karena Laravel mengusung konsep OOP, data ini ditangkap sebagai objek oleh *Controller*, yang kemudian dibawa ke *View* untuk diberikan skrip PHP yang selanjutnya disisipkan kedalam kode HTML untuk dikembalikan lagi ke *Controller* agar ditampilkan ke *Client*.

2.14.7. Authentication dan Authorization

Proses pendaftaran dan *login* pengguna pada sistem perangkat lunak adalah suatu fitur yang umum dan penting. Laravel memberikan *tools-tools* berbeda untuk memanfaatkan fungsi-fungsi yang dibutuhkan pada proses tersebut menjadi lebih aman dan mudah. Disamping sistem autentikasi dari Laravel, kita bisa menggunakan metode sederhana untuk proses otorisasi operasi pengguna ketika mereka akan mengakses sebuah sumber data. Meskipun pengguna sudah sah secara autentikasi, tetapi belum tentu mereka berhak membaca atau mengubah isi beberapa model-model *Eloquent* yang ditangani aplikasi atau rekaman rekaman *database*.

2.14.8. Middleware

Middleware dimanfaatkan untuk mem-*filter* permintaan HTTP pada saat pengguna / *client* memasuki aplikasi. *Programmer* dapat menggunakan *middleware* untuk lakukan tugas-tugas berikut ini: pemeriksaan autentikasi dan otorisasi, validasi sesi dan modifikasi variabel sesi, membaca, mengatur atau memodifikasi *header* permintaan dan tanggapan; merekam transaksi data; pemanggilan API, dan lain-lain.

2.14.9. Validation

Kelas validator Laravel membantu memvalidasi fungsionalitas elemen-elemen seperti form, model *database*. Validator ini mendukung penerimaan input data; mendeklarasikan peran-peran validasi khusus dan mendeklarasikan pesan-pesan validasi khusus.

2.14.10. Pengamanan aplikasi

Sebelum meluncurkan sistem perangkat lunak kita ke lingkungan publik, *programmer* harus mempertimbangkan beberapa pertanyaan-pertanyaan tentang keamanan. Laravel melindungi sistem perangkat lunak terhadap serangan-serangan yang sering terjadi. Beberapa diantaranya adalah: *Cross-site request forgery* (CSRF), *Cross-site scripting* (XSS), *SQL injection*, *mass assignment vulnerability*.

BAB III

METODE PENELITIAN

3.1. *Requirements Gathering*

Menganalisa kebutuhan adalah tahap yang tidak dapat terpisahkan dari proses pengembangan perangkat lunak, begitu pula penelitian kami dalam merancang aplikasi untuk penyedia jasa kursus mengemudi, hal pertama yang kita lakukan adalah melakukan wawancara dan observasi terhadap tiga penyedia jasa kursus mengemudi di Surabaya untuk mendapatkan pengetahuan tentang proses bisnis dan kendala-kendala yang selama ini dihadapi oleh mereka. Berikut adalah hasil dari proses pengumpulan dan analisa kebutuhan yang kami lakukan.

3.1.1. Proses Bisnis

Seperti yang kita ketahui, berkendara dengan tertib di jalan raya merupakan tanggung jawab kita semua, namun, dalam menjaga ketertiban tersebut dibutuhkan pihak yang memiliki kewenangan untuk mengatur, mengarahkan, bahkan memberikan sanksi apabila terjadi pelanggaran di jalan raya, pihak tersebut tidak lain adalah kepolisian dan dinas perhubungan. Oleh karena itu, para penyedia jasa kursus mengemudi wajib bekerja sama dengan pihak-pihak tersebut untuk mendapatkan izin sebelum melangsungkan bisnisnya. Jika kita mengacu pada Peraturan Daerah Kota Surabaya Nomor 22 Tahun 2012 tentang Izin Penyelenggaraan Pendidikan dan Pelatihan Mengemudi Kendaraan Bermotor, para penyedia jasa kursus mengemudi, baik perorangan maupun lembaga pendidikan, yang ingin menawarkan jasanya kepada masyarakat diharapkan dapat memenuhi persyaratan yang diperlukan.

Namun, setelah kami lakukan wawancara dengan tiga penyedia jasa kursus mengemudi, ternyata pemenuhan persyaratan yang dibutuhkan hanya dilakukan

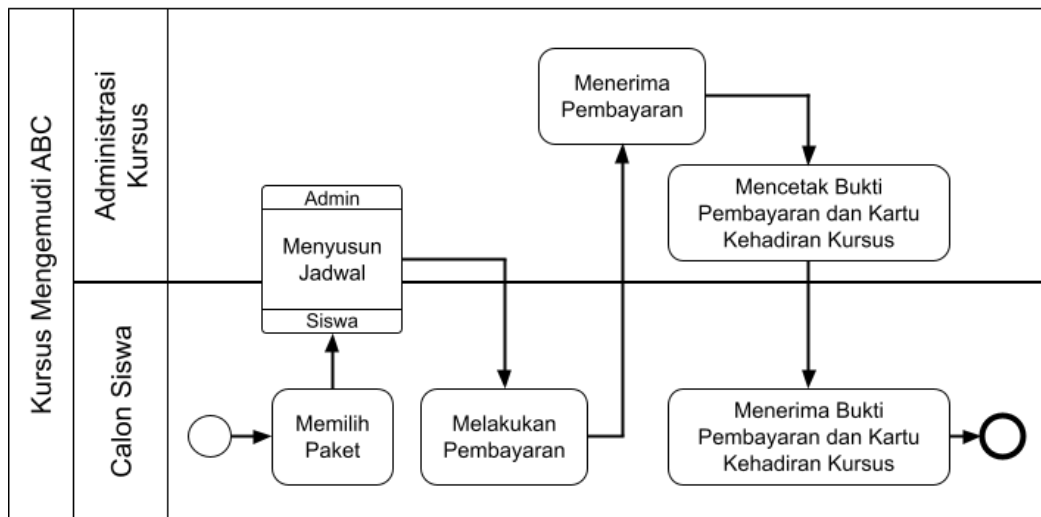
sebagian saja. Berikut merupakan hasil analisa pemenuhan persyaratan kami dengan tiga penyedia jasa kursus mengemudi.

Tabel 3. 1 Pemenuhan Legalitas Jasa Kursus Mengemudi

No.	Dokumen Persyaratan	Kursus ABC	Sie Bersaudara	“Hafiz”
1	Akte pendirian perusahaan (Lembaga Kursus) atau KTP (Perorangan)	✓	✓	✓
2	Struktur organisasi penyelenggara pendidikan mengemudi		✓	
3	Daftar nama personil, riwayat hidup pengelola dan instruktur & Sertifikat Instruktur	✓	✓	✓
4	Tata tertib penyelenggaraan pendidikan dan pelatihan mengemudi kendaraan bermotor	✓	✓	
5	Rekomendasi penyelenggaraan kursus mengemudi dari Satlantas Polrestabes Surabaya			
6	Kurikulum pendidikan dan pelatihan mengemudi kendaraan bermotor			✓

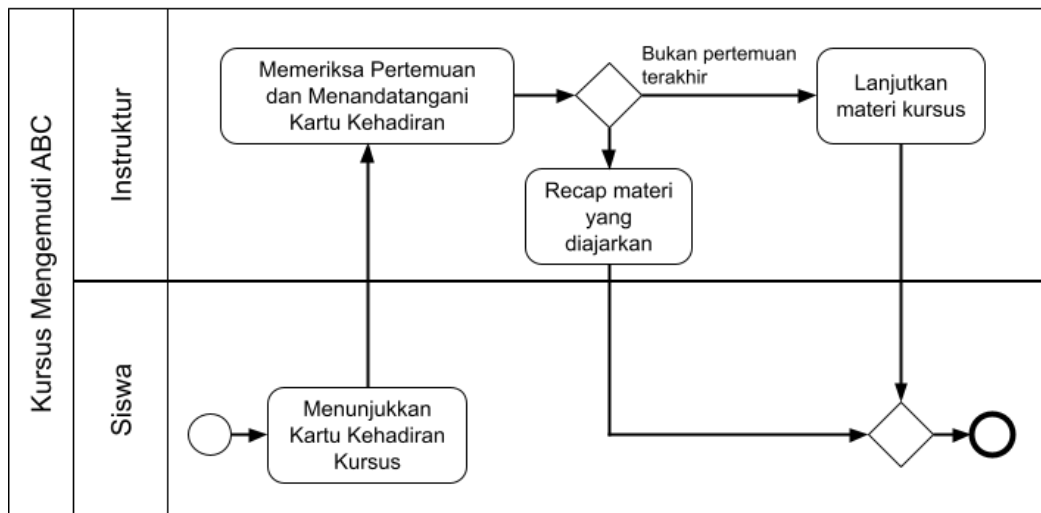
Pada saat kami melakukan penelitian ini belum ada standar atau peraturan yang mengatur atau mewajibkan para penyedia jasa kursus mengemudi untuk

memiliki suatu proses tertentu. Sehingga, langkah selanjutnya adalah kami menanyakan bagaimana proses bisnis yang ada pada ketiga kursus mengemudi tersebut. Agar lebih mudah dipahami, kami bagi proses bisnis menjadi 2 alur, Pendaftaran Kursus dan Proses Kursus Mengemudi.



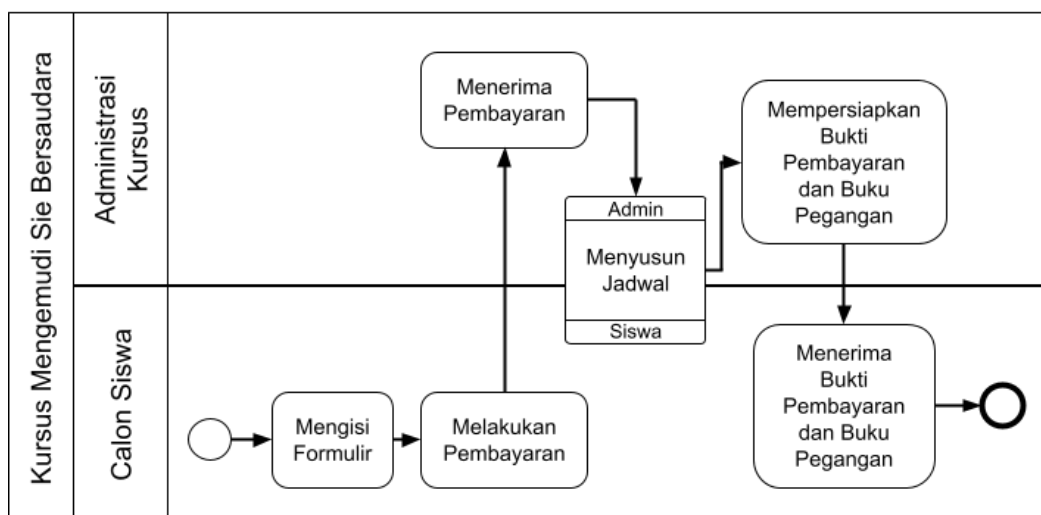
Gambar 3. 1 Proses Pendaftaran pada Kursus Mengemudi ABC

Untuk proses pendaftaran pada Kursus ABC dimulai dengan calon siswa memilih kelas kursus kemudian menyusun jadwal, jika calon siswa merasa sudah puas proses selanjutnya adalah pembayaran, dimana nominal uang yang dibayarkan harus diperiksa terlebih dahulu oleh pihak administrasi Kursus ABC, jika sudah benar, pihak administrasi Kursus ABC memberikan tanda terima pembayaran ke calon siswa beserta kartu kehadiran kursus yang nantinya wajib dibawa setiap kursus.



Gambar 3. 2 Proses Kursus di Kursus Mengemudi ABC

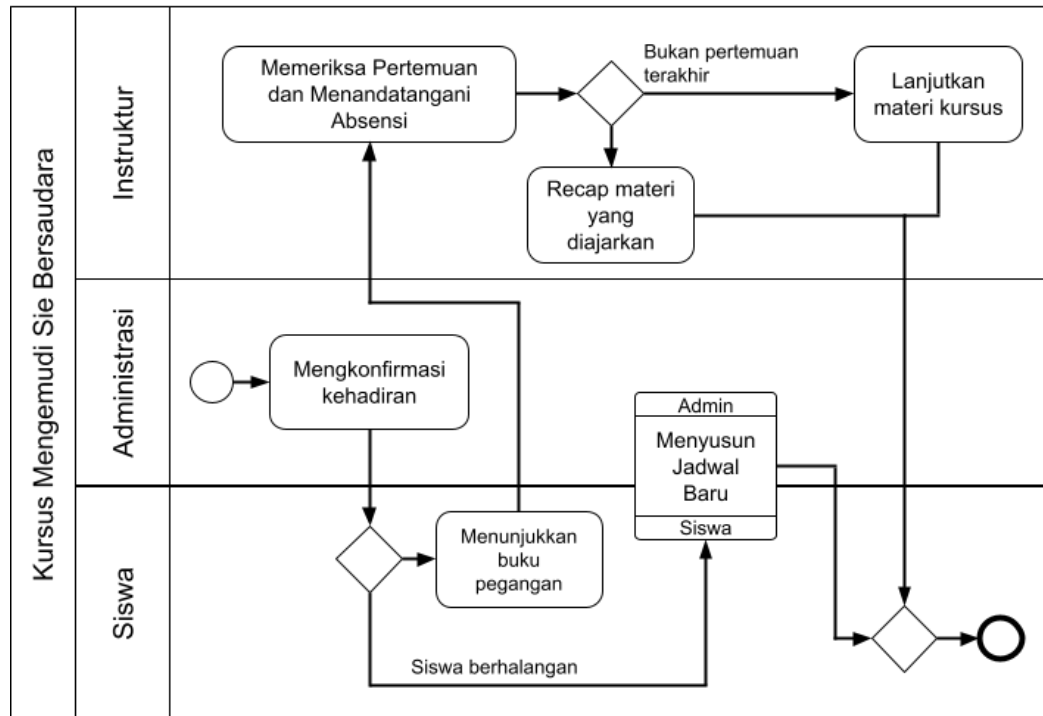
Proses kursus di kursus mengemudi ABC dimulai dengan siswa hadir ke tempat kursus atau siswa bisa memilih untuk dijemput oleh instruktur kursus di rumahnya. Sebelum memulai praktik, siswa menunjukkan kartu kehadiran kursus ke instruktur untuk ditandatangani kehadirannya bersamaan dengan instruktur memeriksa siswa saat ini berada di pertemuan ke berapa, dimana siswa selanjutnya mempelajari materi sesuai dengan pertemuan saat ini.



Gambar 3. 3 Proses Pendaftaran di Kursus Mengemudi Sie Bersaudara

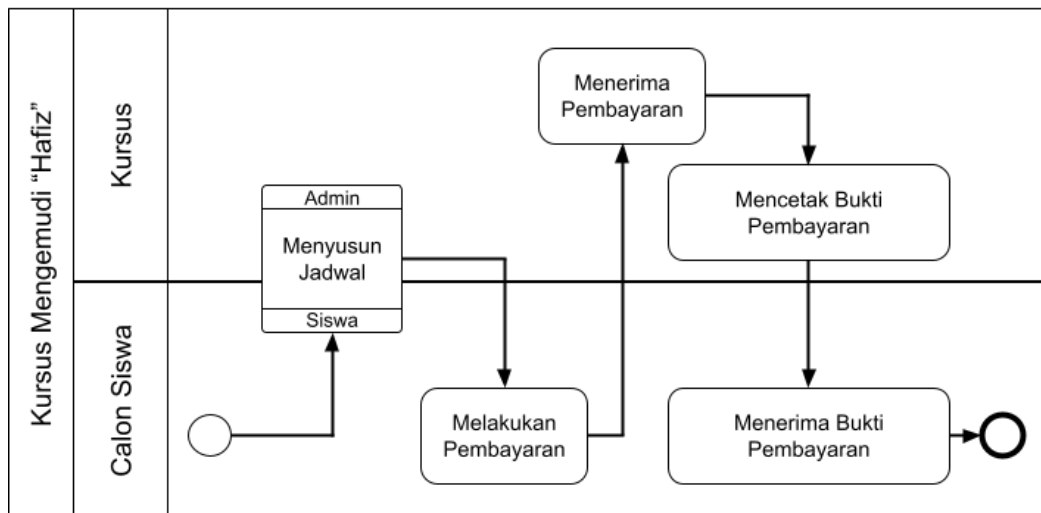
BPMN di atas menunjukkan alur proses pendaftaran di Kursus Sie Bersaudara, dimulai dengan siswa mengisi formulir, di dalam formulir tersebut terdapat opsi kelas dan deskripsi dari masing-masing kelas, setelah mengisi

formulir, siswa melakukan pembayaran, sama dengan kursus ABC, pihak administrasi kursus Sie Bersaudara memeriksa apakah nominal yang dibayarkan sudah benar, jika sudah, selanjutnya siswa memilih jadwal kursus. Kemudian, pihak administrasi memberikan bukti pembayaran dan buku pegangan yang berisi materi yang nantinya akan diajarkan serta tertera lembar absensi di dalamnya.



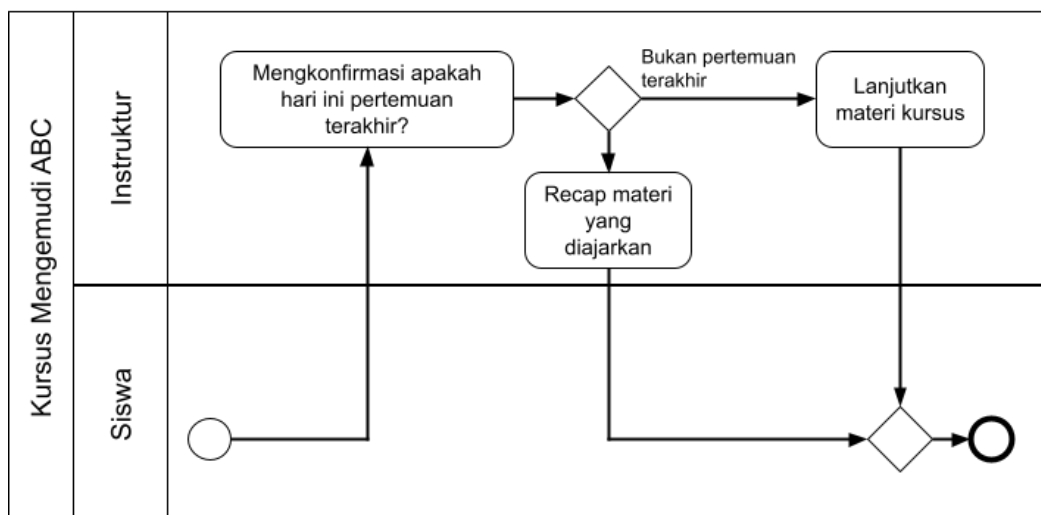
Gambar 3. 4 Proses Kursus di Kursus Mengemudi Sie Bersaudara

Untuk proses kursus di kursus Sie Bersaudara dimulai dengan pihak administrasi mengkonfirmasi kehadiran siswa 1-2 jam sebelum jadwal kursus, jika siswa ternyata tidak bisa hadir, pihak administrasi akan menawarkan jadwal baru ke siswa yang tidak hadir tersebut, jika siswa hadir, selanjutnya siswa akan bertemu dengan instruktur, siswa harus menunjukkan buku pegangannya untuk ditandatangani lembar absensinya. Sama dengan kursus mengemudi ABC, instruktur memeriksa di pertemuan ke berapa siswa saat ini, apabila siswa ada di pertemuan terakhir, instruktur nantinya akan mengulang materi dari awal hingga pertemuan sebelumnya, kemudian, sebelum siswa pulang, instruktur wajib memfoto siswa didepan mobil kursus untuk selanjutnya dicetak menjadi sertifikat seandainya siswa ingin mendapatkan sertifikat kursus.



Gambar 3. 5 Proses Pendaftaran di Kursus Mengemudi "Hafiz"

Berbeda dengan dua kursus sebelumnya, kursus mengemudi “Hafiz” tergolong kursus mengemudi perorangan. Proses pendaftaran dimulai dengan siswa langsung memilih jadwal, kemudian siswa melakukan pembayaran, dan yang terakhir pihak kursus mencetak bukti pembayaran untuk diberikan ke siswa.



Gambar 3. 6 Proses Kursus di Kursus Mengemudi "Hafiz"

Hampir sama dengan proses kursus sebelumnya, dimulai dengan siswa hadir ke tempat kursus atau dijemput oleh instruktur, selanjutnya instruktur menanyakan apakah pertemuan saat ini pertemuan terakhir, jika tidak, instruktur dan siswa bisa melanjutkan materi dari pertemuan sebelumnya, jika iya, instruktur bersama siswa akan mempraktekkan materi kursus dari awal hingga pertemuan sebelumnya.

3.1.2. Analisa Kebutuhan

Dalam mengembangkan perangkat lunak, analisa kebutuhan merupakan hasil riset dengan beberapa pihak dan bidang yang berbeda, seperti dari bidang bisnis diantaranya adalah situasi pasar, pengguna akhir, calon konsumen, dan peluang-peluang teknis. Sedangkan dari perspektif desain dan *prototype*, analisa kebutuhan dapat dibagi menjadi 4 aspek: bisnis/pemasaran, fungsional/fitur, teknologi, dan kegunaan.

Kebutuhan bisnis/pemasaran memberikan pemahaman tentang kebutuhan bisnis atau kondisi *marketplace* yang ada. Kebutuhan fungsional memberikan pemahaman akan fitur-fitur yang dibutuhkan untuk mendukung kebutuhan bisnis atau pemasaran. Kebutuhan fungsional juga terkadang merupakan hasil dari hasil riset pengguna dan pengujian kemudahan penggunaan. Selanjutnya spesifikasi fitur ini nantinya akan diterapkan pada *prototype*. Kebutuhan teknologi memberikan pemahaman tentang teknologi yang dibutuhkan untuk mengimplementasikan fitur yang diperlukan. Sedangkan kebutuhan kegunaan memberikan pemahaman tentang *User Experience* dan syarat-syarat kegunaan yang dibutuhkan untuk memudahkan pengguna dalam mengadopsi perangkat lunak yang baru nantinya.

Tabel 3. 2 Kebutuhan Aplikasi

No.	Kebutuhan / Persyaratan	Aspek
Sisi Admin Kursus (Lembaga) / Pemilik (Perorangan & Lembaga)		
1	Fungsi <i>Login & Logout</i> sebagai admin / pemilik	Fungsional
2	Dashboard yang berisi jadwal kursus setiap Instruktur per hari	Fungsional

Tabel 3. 3 Kebutuhan Aplikasi (Lanjutan-1)

No.	Kebutuhan / Persyaratan	Aspek
3	Menambah kelas kursus beserta deskripsi, harga, jumlah pertemuan, benefit, dan kategori kelas kursus	Fungsional
4	Mengubah nama, deskripsi, harga, jumlah pertemuan, benefit, dan kategori kelas kursus	Fungsional
5	Menghapus / menonaktifkan kelas kursus	Fungsional
6	Menampilkan detail <i>progress</i> setiap siswa	Fungsional
7	Mengajukan perubahan jadwal kursus kepada siswa dan instruktur	Kegunaan
8	Menampilkan hasil pengisian formulir pendaftaran kursus	Fungsional
9	Menghubungi siswa dan instruktur	Fungsional
10	Mengkonfirmasi status pembayaran kursus dari siswa	Bisnis/Pemasaran
Sisi Instruktur (Lembaga)		
1	Fungsi <i>Login & Logout</i> sebagai Instruktur	Fungsional
2	Dashboard yang berisi jadwal kursus per hari	Fungsional
3	Menampilkan daftar siswa aktif	Fungsional
4	Menampilkan detail <i>progress</i> siswa	Fungsional
5	Mengajukan atau mengkonfirmasi perubahan jadwal kursus kepada siswa	Kegunaan

Tabel 3. 4 Kebutuhan Aplikasi (Lanjutan-2)

No.	Kebutuhan / Persyaratan	Aspek
6	Menghubungi siswa	Fungsional
<i>Sisi General User</i>		
1	Fungsi <i>Login & Logout</i> sebagai <i>General User</i>	Fungsional
2	Dashboard yang berisi rekomendasi kelas kursus dan <i>progress</i> kursus yang sedang aktif	Fungsional
3	Mendaftarkan diri ke kursus mengemudi	Fungsional
4	Menghubungi admin / pemilik kursus dan instruktur	Fungsional
5	Menampilkan detail <i>progress</i> kursus yang berlangsung	Fungsional
6	Mengajukan perubahan jadwal kursus kepada instruktur	Kegunaan
7	Menampilkan hasil pengisian formulir pendaftaran kursus	Fungsional
8	Menjadi pemilik kursus dengan mengirimkan surat izin penyelenggaraan kursus mengemudi	Fungsional
<i>Kebutuhan Teknis</i>		
1	Sistem mampu dijalankan melalui Smartphone semua pihak	Teknologi
2	Sistem mudah diakses di web/internet	Teknologi

3.2. Iterasi Pertama

Selama rangkaian proses *prototyping* yang kami lakukan, terjadi dua kali iterasi dimana setelah kami melakukan pengujian awal dengan pengguna, kami harus kembali lagi ke proses desain singkat, untuk menyisipkan saran dan masukan yang kami terima pada proses pengujian.

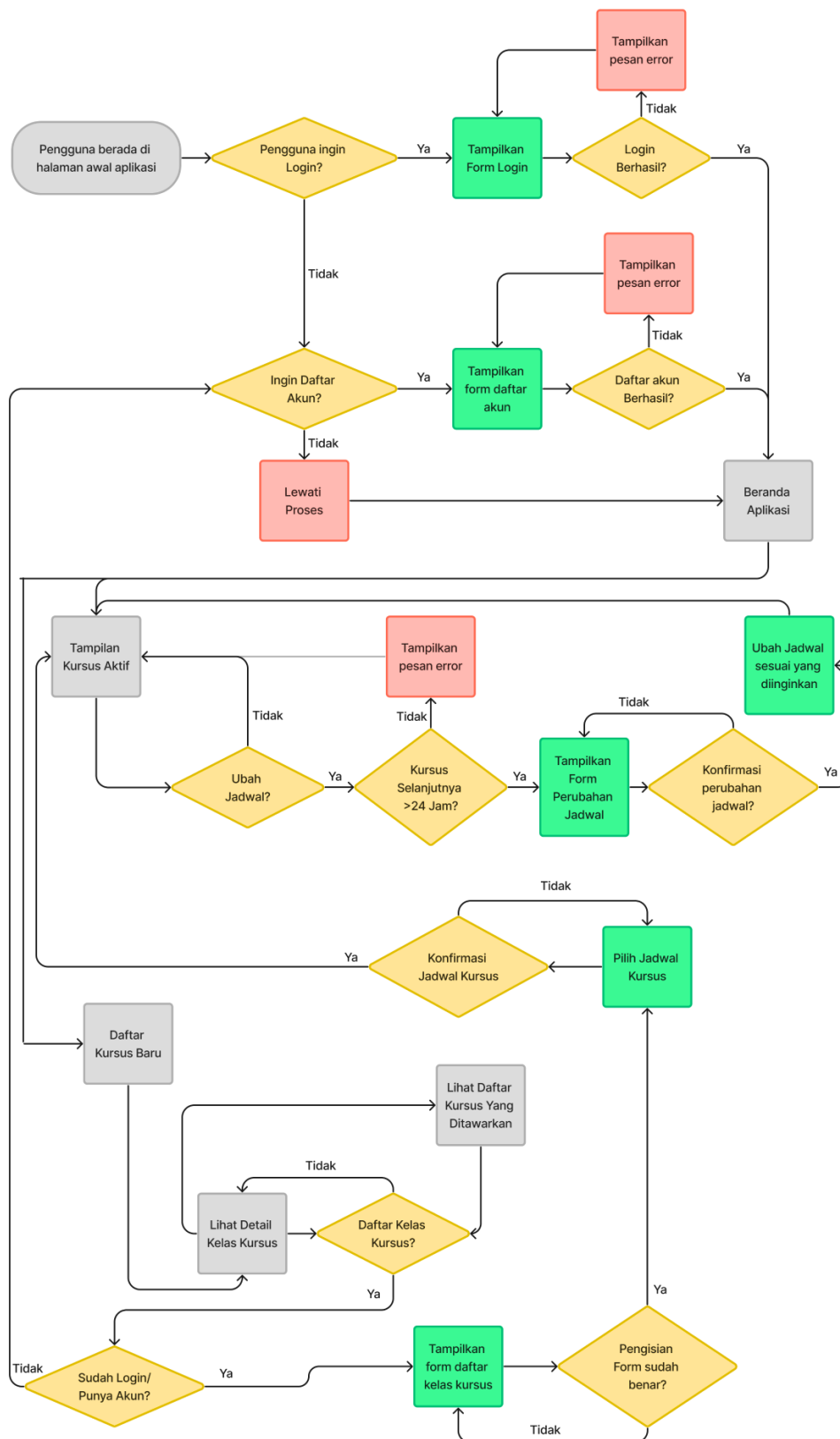
3.2.1. Quick Design

Dari analisa kebutuhan diatas, kami mulai memahami kebutuhan dan kendala-kendala yang dihadapi oleh pihak penyedia jasa kursus mengemudi. Selanjutnya, pada tahap ini, kami mulai dengan membuat rangkaian alur pengguna atau sering dikenal dengan *user flow*, alur ini nantinya akan menjadi basis bagi kami untuk membuat sketsa awal aplikasi.

3.2.1.1. Alur Pengguna

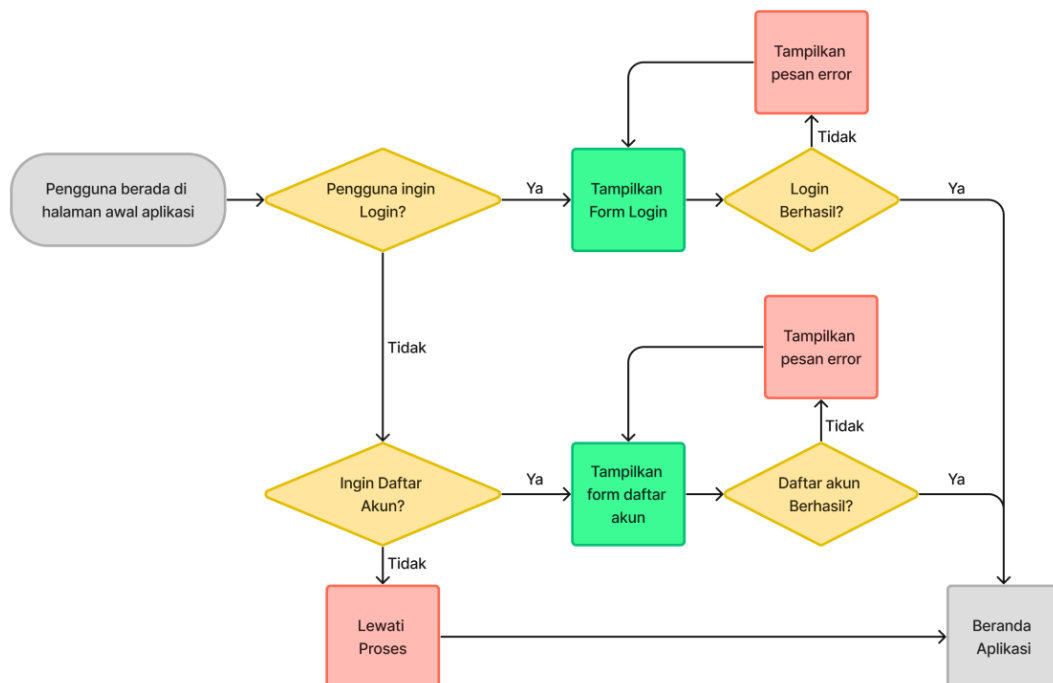
Alur pengguna bertujuan tidak hanya membantu tim pengembang dalam menghasilkan *prototype*, pihak lain yang merasakan manfaat dari alur pengguna ini adalah pengguna itu sendiri, dalam hal ini adalah pihak penyedia jasa kursus mengemudi. Dengan mengetahui alur pengguna, pihak penyedia jasa kursus mengemudi dapat mengevaluasi apakah alur pengguna yang dirancang oleh tim pengembang sudah sesuai dengan proses bisnis yang saat ini sedang berjalan.

Alur yang menjadi fokus kami untuk saat ini adalah alur yang sama ketika seorang calon siswa / peserta kursus akan mendaftar untuk mengikuti kelas kursus mengemudi, yang sebelumnya sudah kami lakukan wawancara dan observasi bersama dengan pihak penyedia jasa kursus mengemudi pada sub bab sebelumnya. Berikut adalah alur pengguna untuk *prototype* iterasi pertama kami.



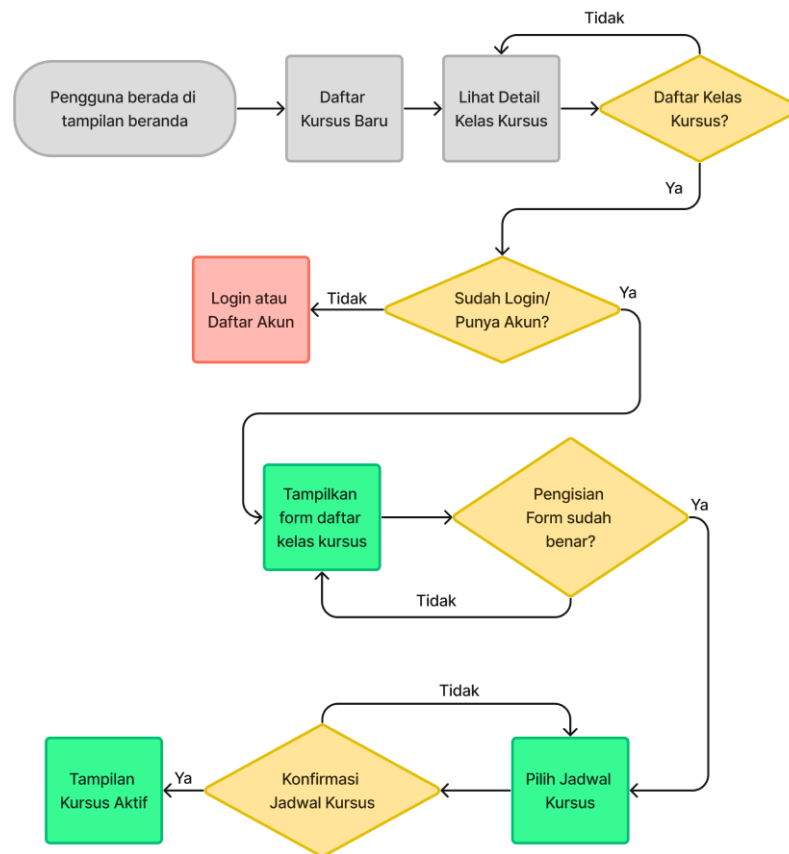
Gambar 3. 7 Alur Pengguna untuk Iterasi Pertama

Untuk menyederhanakan alur pengguna yang kompleks diatas, kami akan membagi alur pengguna menjadi alur tugas (*task flow*), alur tugas untuk *login* atau mendaftarkan akun, alur tugas untuk melakukan pendaftaran kelas kursus, dan alur tugas untuk mengubah jadwal kursus. Dibawah ini adalah hasil penyederhanaan alur pengguna diatas.



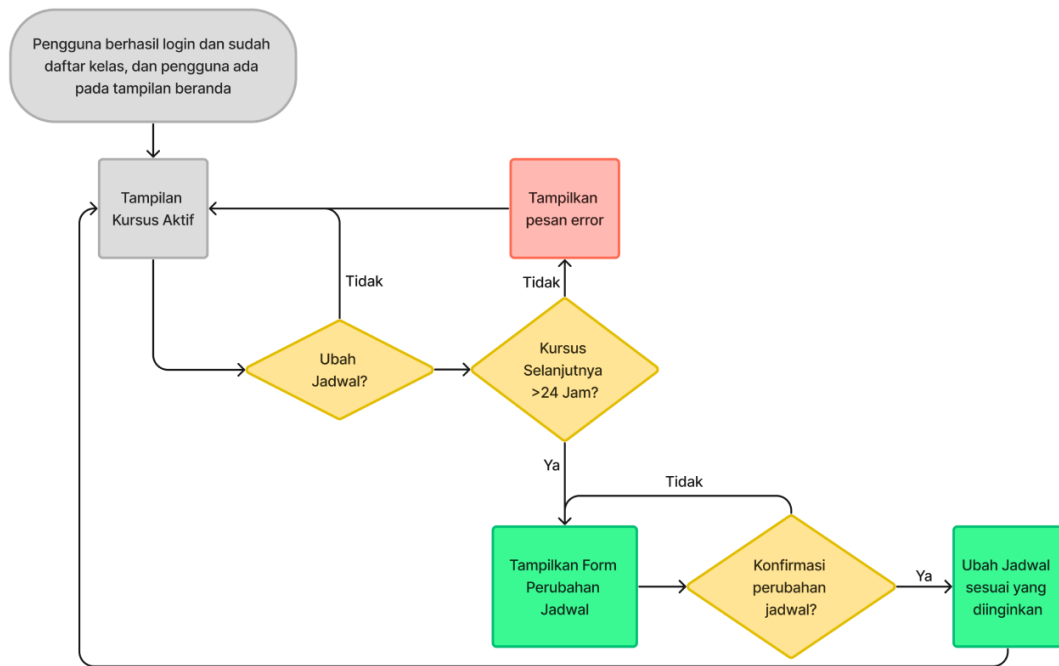
Gambar 3. 8 Alur Tugas Proses *Login*

Alur proses *login* dimulai dengan pengguna berada di tampilan awal aplikasi. Kemudian pengguna dapat memilih ingin *login* atau daftar akun baru, atau opsi selanjutnya adalah memilih untuk melakukan proses ini nanti apabila memang dibutuhkan. Apabila pengguna memilih untuk *login*, selanjutnya tampilan yang muncul adalah *form login*. Jika, pengguna berhasil *login*, tampilan selanjutnya adalah beranda aplikasi, namun, jika pengguna gagal, maka akan tampil pesan *error* pada *form login*. Sedangkan jika pengguna memilih untuk mendaftarkan akun baru, tampilan yang muncul selanjutnya adalah tampilan *form* daftar akun. Serupa dengan alur *login*, apabila pengguna berhasil melakukan pendaftaran akun, pengguna akan melihat tampilan beranda, namun, jika pengguna gagal, akan timbul pesan *error* pada form pendaftaran akun.



Gambar 3. 9 Alur Tugas Proses Pendaftaran Kelas Kursus

Untuk alur pendaftaran kelas kursus, dimulai dengan pengguna berada di tampilan beranda, pengguna memilih kelas kursus yang ingin diikuti. Ketika pengguna memilih kelas kursus yang akan diikuti, tampilan selanjutnya adalah detail kelas kursus, jika pengguna memilih untuk mendaftarkan diri ke kelas tersebut, sistem akan memeriksa ulang apakah pengguna sudah melakukan *login* atau daftar akun pada alur tugas sebelumnya. Jika belum, tampilan berikutnya adalah tampilan awal aplikasi, jika pengguna sudah melakukan *login* atau daftar akun, tampilan selanjutnya adalah *form* pendaftaran kelas kursus. Jika *form* sudah diisi dengan benar, tampilan berikutnya adalah tampilan untuk memilih jadwal kursus, jika *form* tidak diisi dengan lengkap, maka akan dikembalikan lagi ke tampilan *form* pendaftaran kursus. Jika pengguna puas dengan jadwal yang dipilih, pengguna akan dialihkan ke tampilan kursus aktif, jika pengguna merasa tidak puas, pengguna dapat mengubah ulang jadwal kursus.




Gambar 3. 10 Alur Tugas Proses Perubahan Jadwal

Alur tugas dapat dimulai apabila pengguna memilih untuk melakukan *login* atau daftar akun sebelumnya, kemudian sudah pernah mendaftar kursus dan berada di tampilan beranda. Selanjutnya, pengguna dapat beralih ke tampilan kursus aktif. Jika pengguna memilih untuk mengubah jadwal kursus, sistem akan memeriksa apakah pertemuan kursus selanjutnya lebih dari 24 jam? Jika kurang dari 24 jam, akan muncul pesan *error* pada tampilan kursus aktif. Jika lebih dari 24 jam, pengguna akan ditampilkan oleh *form* perubahan jadwal. Jika pengguna mengkonfirmasi perubahan, selanjutnya jadwal baru akan ditampilkan pada tampilan kursus aktif, jika pengguna belum yakin dengan jadwal baru, pengguna akan diarahkan kembali ke *form* perubahan jadwal.

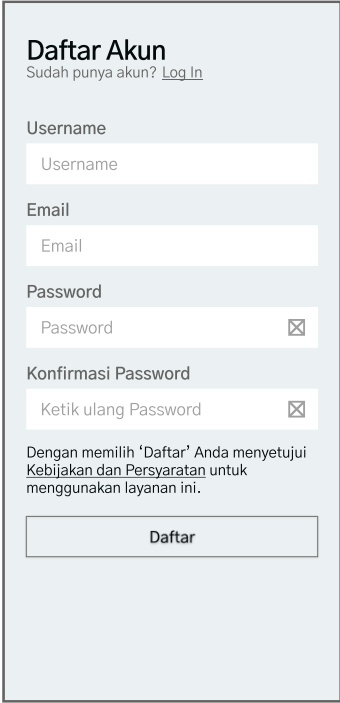
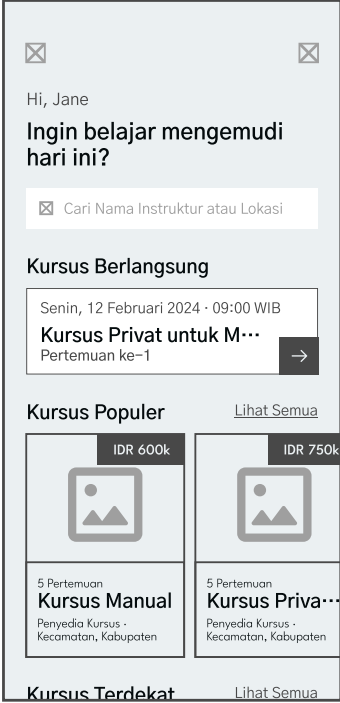
3.2.1.2. Sketsa Awal Aplikasi

Berikut adalah daftar sketsa awal aplikasi yang sudah kami desain untuk iterasi pertama.


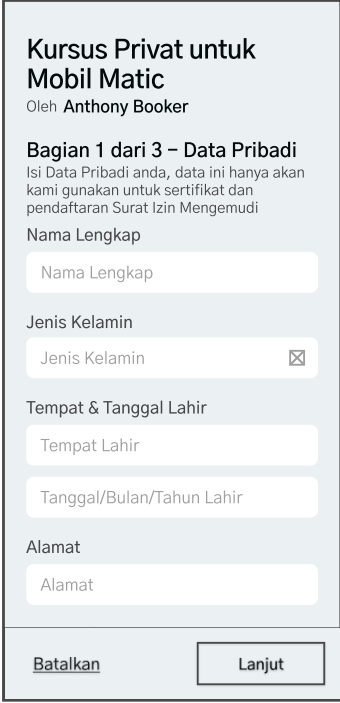
Tabel 3. 5 Sketsa Awal untuk Iterasi Pertama

No.	Sketsa Aplikasi	Keterangan
1		Sketsa awal untuk tampilan depan aplikasi. Selanjutnya, peserta kursus dapat melakukan dua aksi, yaitu, <i>Login</i> / Daftar akun atau langsung masuk ke beranda aplikasi


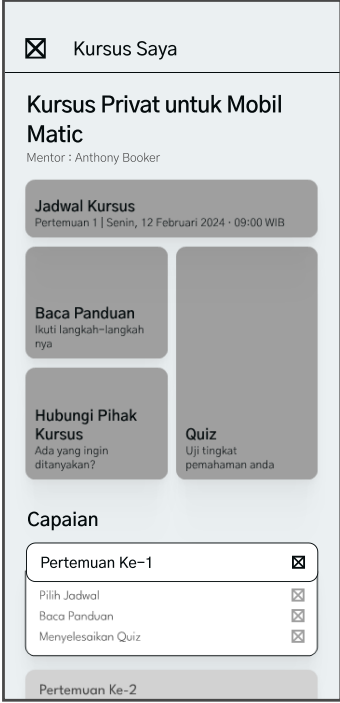
Tabel 3. 6 Sketsa Awal untuk Iterasi Pertama (Lanjutan-1)

No.	Sketsa Aplikasi	Keterangan
2	 <p>Daftar Akun Sudah punya akun? Log In</p> <p>Username Username</p> <p>Email Email</p> <p>Password Password</p> <p>Konfirmasi Password Ketik ulang Password</p> <p>Dengan memilih 'Daftar' Anda menyetujui Kebijakan dan Persyaratan untuk menggunakan layanan ini.</p> <p>Daftar</p>	<p>Sketsa awal <i>form</i> pendaftaran akun, dimana peserta kursus dapat membuat akun baru sebelum menggunakan aplikasi</p>
3	 <p>Hi, Jane</p> <p>Ingin belajar mengemudi hari ini?</p> <p>Cari Nama Instruktur atau Lokasi</p> <p>Kursus Berlangsung</p> <p>Senin, 12 Februari 2024 · 09:00 WIB</p> <p>Kursus Privat untuk M... Pertemuan ke-1</p> <p>Kursus Populer Lihat Semua</p> <p>IDR 600k</p> <p>5 Pertemuan Kursus Manual Penyedia Kursus - Kecamatan, Kabupaten</p> <p>IDR 750k</p> <p>5 Pertemuan Kursus Priva... Penyedia Kursus - Kecamatan, Kabupaten</p> <p>Kursus Terdekat Lihat Semua</p>	<p>Sketsa awal beranda aplikasi yang menampilkan kursus aktif, bagian tersebut akan muncul apabila peserta sudah pernah mendaftarkan diri ke sebuah kelas kursus dan berisi rekomendasi-rekomendasi kelas kursus yang lain</p>

Tabel 3. 7 Sketsa Awal untuk Iterasi Pertama (Lanjutan-2)

No.	Sketsa Aplikasi	Keterangan
4	 <p>The sketch shows a mobile app interface for a private driving course. At the top, there's a header with a close icon and a thumbnail image. Below it, the title 'Kursus Privat untuk Mobil Matic' is displayed, followed by the instructor's name 'Oleh Anthony Booker'. A section titled 'Fasilitas & Keuntungan' lists three benefits: 'Mobil anda sendiri', '2 Jam Pertemuan', and 'Instruktur datang ke rumah'. Below this is a 'Deskripsi Kelas' section with a paragraph of text. At the bottom, the price 'Rp. 750.000,-' is shown next to a 'Daftar Kelas' button.</p>	<p>Sketsa awal detail kelas kursus, pada tampilan ini, peserta kursus nantinya mampu mendapatkan informasi singkat tentang kelas kursus sebagai pertimbangan</p>
5	 <p>The sketch shows a mobile app interface for a private driving course registration form. The title 'Kursus Privat untuk Mobil Matic' is at the top, followed by the instructor's name 'Oleh Anthony Booker'. Below this is a section titled 'Bagian 1 dari 3 – Data Pribadi' with a paragraph of text. The form includes input fields for 'Nama Lengkap', 'Jenis Kelamin', 'Tempat & Tanggal Lahir' (with sub-fields for 'Tempat Lahir' and 'Tanggal/Bulan/Tahun Lahir'), and 'Alamat'. At the bottom, there are two buttons: 'Batalkan' and 'Lanjut'.</p>	<p>Sketsa <i>form</i> pendaftaran kelas kursus sebagai pengganti formulir pendaftaran yang untuk saat ini menggunakan media kertas yang membutuhkan proses pengarsipan lebih lanjut</p>

Tabel 3. 8 Sketsa Awal untuk Iterasi Pertama (Lanjutan-3)

No.	Sketsa Aplikasi	Keterangan
6		<p>Sketsa awal pemilihan atau perubahan jadwal kursus, dimana peserta kursus dapat melakukan pemilihan jadwal untuk setiap pertemuan</p>
7		<p>Sketsa awal detail kursus aktif, dimana peserta kursus dapat memperoleh informasi-informasi terkait kursus yang diikuti</p>

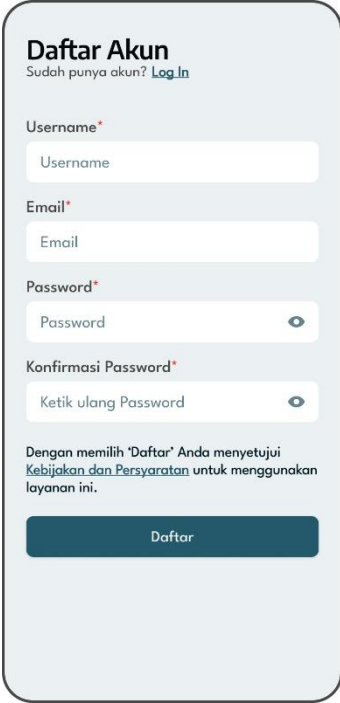

3.2.2. *Built Prototype*

Hasil dari desain singkat sebelumnya akan dikonversi dan ditambahkan detail-detail pendukung didalamnya. Dengan dibangunnya *prototype* pertama ini, fitur-fitur aplikasi mulai dapat dievaluasi kemudahan dan efektifitas penggunaannya. Berikut adalah hasil proses *prototyping* untuk iterasi pertama.



Tabel 3. 9 Desain *Prototype* untuk Iterasi Pertama

No.	Desain <i>Prototype</i>	Keterangan
1		Tampilan depan aplikasi kami tambahkan gambar pendukung dan membedakan warna tombol untuk memberikan kesan hierarki yang lebih jelas


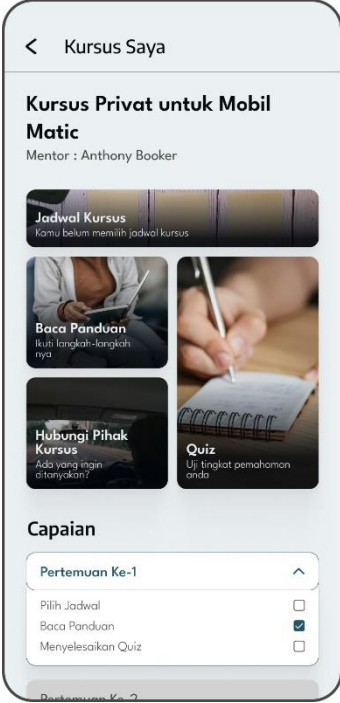
Tabel 3. 10 Desain *Prototype* untuk Iterasi Pertama (Lanjutan-1)

No.	Sketsa Aplikasi	Keterangan
2		<p>Tampilan <i>form</i> pendaftaran akun, ikon mata untuk memberi tahu kan peserta kursus bahwa <i>password</i> dapat disembunyikan/diperlihatk an</p>
3		<p>Tampilan beranda aplikasi, penambahan warna dan ikon-ikon pendukung untuk memudahkan pencernaan informasi</p>

Tabel 3. 11 Desain *Prototype* untuk Iterasi Pertama (Lanjutan-2)

No.	Sketsa Aplikasi	Keterangan
4	 <p>The prototype for 'Kursus Privat untuk Mobil Matic' features a car interior image at the top. Below it, the title 'Kursus Privat untuk Mobil Matic' is displayed, followed by the instructor's name 'Oleh Anthony Booker'. A section titled 'Fasilitas & Keuntungan' contains three icons: a car for 'Mobil anda sendiri', a clock for '2 Jam Pertemuan', and a person for 'Instruktur datang ke rumah'. A 'Deskripsi Kelas' section follows, and at the bottom, the price 'Rp. 750.000,-' is shown next to a 'Daftar Kelas' button.</p>	<p>Tampilan detail kelas kursus, sama seperti sebelumnya, penambahan gambar, ikon, dan pemberian warna tombol untuk memberikan kontras agar memudahkan peserta untuk mengidentifikasi keberadaan tombol</p>
5	 <p>The prototype for 'Kursus Privat untuk Mobil Matic' shows a registration form titled 'Bagian 1 dari 3 - Data Pribadi'. It includes a disclaimer about data usage for certification and license application. The form contains input fields for 'Nama Lengkap', a dropdown for 'Jenis Kelamin', input fields for 'Tempat & Tanggal Lahir', and an 'Alamat' field. At the bottom, there are 'Batalkan' and 'Lanjut' buttons.</p>	<p>Tampilan <i>form</i> pendaftaran kursus sesuai dengan pemilihan warna sebelumnya</p>

Tabel 3. 12 Desain *Prototype* untuk Iterasi Pertama (Lanjutan-3)

No.	Sketsa Aplikasi	Keterangan
6		<p>Tampilan pemilihan jadwal, pemberian warna, ikon, dan pemilihan tipografi untuk memberikan kontras dan hierarki yang lebih jelas</p>
7		<p>Tampilan detail kursus aktif, menu-menu diberikan gambar ilustratif untuk mengurangi beban kognitif peserta kursus</p>

3.2.3. Customer Evaluation

Pengujian awal kami lakukan dengan menunjukkan hasil desain *prototype* sesuai dengan alur pengguna yang sudah dirancang sebelumnya. Selanjutnya, pengguna kami berikan kesempatan untuk mencoba *prototype* yang kami buat. Selain itu, kami mencatat semua komentar, masukan, dan pertanyaan yang diajukan oleh peserta pengguna *prototype* pada proses ini. Komentar, masukan, dan pertanyaan tersebut nantinya akan kami gunakan untuk menyesuaikan alur *prototype* pada iterasi selanjutnya dan meningkatkan kualitas *prototype*. Berikut adalah daftar peserta pengujian dan komentar/masukan yang diberikan oleh peserta pengujian.

Tabel 3. 13 Evaluasi Pihak Kursus dan Pengguna Aplikasi

No.	Aspek <i>Prototype</i>	Masukan/Saran
Pihak Kursus ABC		
1	-	-
Pihak Kursus Mengemudi Sie Bersaudara		
1	Alur Tugas Pendaftaran Kelas Kursus	Tambahkan status pembayaran (Lunas, Diproses, dll.)
2	Alur Tugas Perubahan Jadwal Kursus	Jika pertemuan selanjutnya < 24 jam, jadwal yang bisa diubah adalah jadwal untuk pertemuan lusa
Pihak Kursus Mengemudi “HAFIZ”		
1	Alur Tugas Pendaftaran Kelas Kursus	Tambahkan <i>prototype</i> yang menampilkan daftar kelas

Tabel 3. 14 Evaluasi Pihak Kursus dan Pengguna Aplikasi (Lanjutan-1)

No.	Aspek <i>Prototype</i>	Masukan/Saran
Pengguna diluar pihak kursus		
		kursus yang ditawarkan oleh pihak kursus
1	Tampilan <i>form</i> pendaftaran akun	<ul style="list-style-type: none"> - Tambahkan jarak antara <i>subheading</i> dan <i>heading</i> - Teks pada tombol dibuat lebih tebal - Teks pengisi sementara pada kolom pengisian dibuat lebih gelap - Label kolom pengisian dibuat lebih tebal
2	Tampilan beranda aplikasi	<ul style="list-style-type: none"> - Kurangi jarak untuk <i>Heading</i> dibagian atas beranda - Teks untuk tombol “Lihat Semua” dibuat lebih tebal lagi - Warna hitam sekunder di tulisan “- Pemula” dan “Pertemuan ke-3” dibuat lebih muda lagi - Kolom pencarian dipindah ke atas ke bagian <i>navbar</i>
3	Tampilan detail kelas kursus	<ul style="list-style-type: none"> - Teks deskripsi kelas dibuat

Tabel 3. 15 Evaluasi Pihak Kursus dan Pengguna Aplikasi (Lanjutan-2)

No.	Aspek <i>Prototype</i>	Masukan/Saran
Pengguna diluar pihak kursus		
		<p>abu-abu</p> <ul style="list-style-type: none"> - Gambar pendukung dibuat lebih besar - Garis luar komponen bagian “Fasilitas & Keuntungan” ubah warnanya jadi lebih gelap - Teks pengisi sementara pada kolom pengisian dibuat lebih gelap - Label kolom pengisian dibuat lebih tebal - Teks untuk harga dibuat lebih tebal
4	Tampilan <i>form</i> pendaftaran kelas kursus	<ul style="list-style-type: none"> - Tambah jarak antara bagian <i>form</i> dan <i>subheading</i> - Label diatas kolom pengisian dibuat lebih tebal
5	Tampilan pemilihan jadwal	<ul style="list-style-type: none"> - Teks pada <i>Tab</i> untuk pertemuan yang tidak terpilih diberikan warna yang lebih muda

Tabel 3. 16 Evaluasi Pihak Kursus dan Pengguna Aplikasi (Lanjutan-3)

No.	Aspek <i>Prototype</i>	Masukan/Saran
Pengguna diluar pihak kursus		
6	Tampilan detail kursus aktif	<ul style="list-style-type: none">- Tombol untuk “Jadwal Kursus” kurang tinggi- Ikon untuk kembali disamakan dengan tampilan detail kursus

3.2.4. Review & Updation

Tahapan ini berfokus untuk memmenyesuaikan antarmuka *prototype* sesuai dengan keinginan pengguna *prototype*, serta mengakomodasi perubahan yang disarankan pengguna *prototype* pada tahap pengujian sebelumnya, jika pengguna merasa kurang senang dengan desain yang kami tawarkan. Semua masukan atau komentar terkait warna, tipografi, ukuran tombol, navigasi, dan semua hal yang terkait dengan antarmuka pengguna akan diperbaiki pada tahapan ini. Peningkatan kualitas *prototype* inilah yang menghantarkan kita ke iterasi kedua atau dalam proyek pengembangan ini merupakan iterasi terakhir.

3.3. Iterasi Kedua

Pada iterasi kedua perbaikan dan peningkatan kualitas *prototype*, kami mengakomodasi semua masukan yang didapatkan pada iterasi sebelumnya. Berikut adalah hasil dari iterasi kedua.

3.3.1. Quick Design

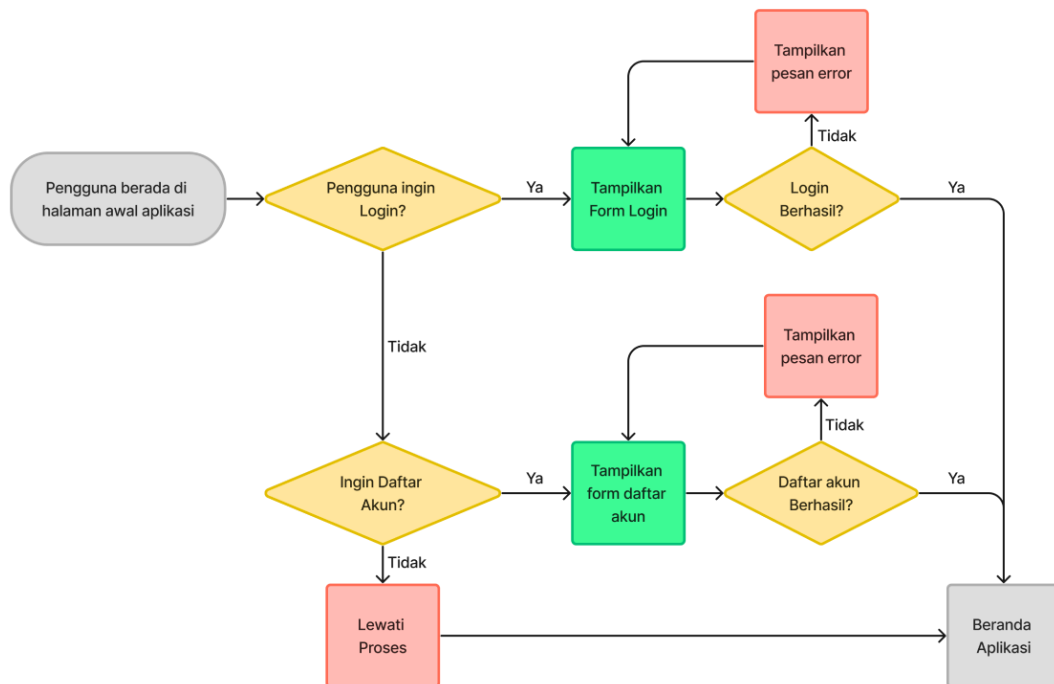
Pada iterasi sebelumnya, terdapat masukan untuk menambahkan beberapa proses tambahan yang dibutuhkan demi memberikan penjelasan yang lebih baik

akan hasil dari produk akhir nantinya. Berikut adalah langkah-langkah kami mengkomodasi perubahan tersebut.

3.3.1.1. Alur Pengguna

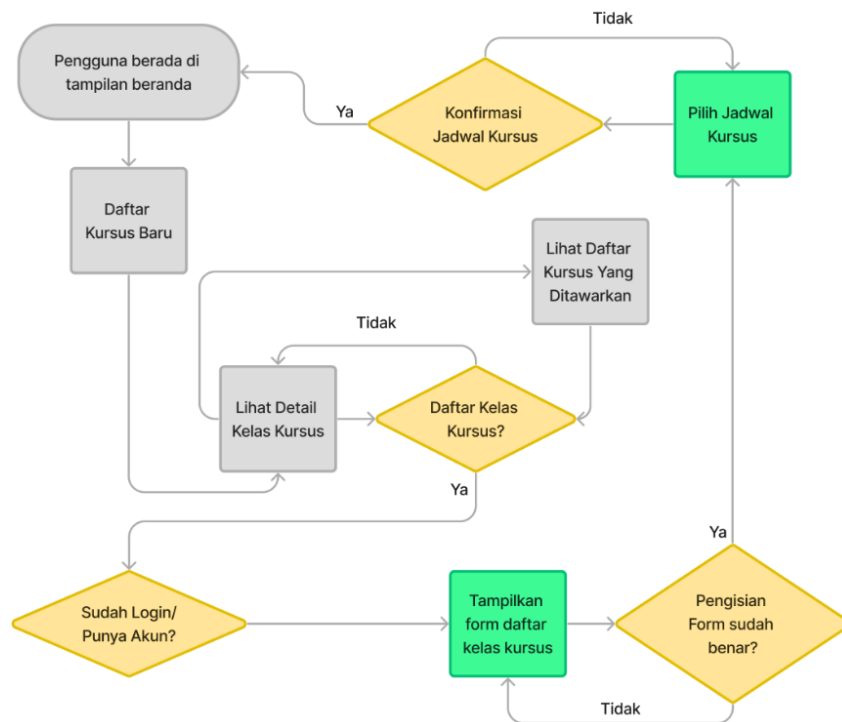
Pihak kursus berpendapat dengan menambahkan proses yang mengindikasikan status pembayaran akan memberikan kesan transparansi yang lebih baik kepada calon peserta kursus. Masukan selanjutnya, ada pada alur perubahan jadwal, dimana jika sebelumnya syarat untuk mengubah jadwal adalah pertemuan selanjutnya harus lebih dari 24 jam, jika tidak, sistem akan menampilkan pesan *error*, pada iterasi ini, pengguna tetap bisa mengganti jadwal kursus walaupun kursus selanjutnya kurang dari 24 jam, namun, yang dapat diubah jadwalnya adalah pertemuan lusa sampai pertemuan terakhir. Selain itu, aplikasi diharapkan dapat menampilkan daftar kelas kursus yang ditawarkan kepada calon peserta kursus sehingga pihak kursus berkesempatan untuk melakukan pemasaran jasa. Berikut adalah rencana perubahan yang kami lakukan untuk memenuhi tiga poin masukan diatas.

Karena keterbatasan ukuran kertas dan tingkat kompleksitas alur yang disebabkan oleh penambahan fitur, kami tidak dapat menampilkan alur pengguna secara utuh untuk iterasi kedua ini, yang dapat kami tampilkan adalah penjelasan lebih rinci tentang alur tugas dari perubahan alur pengguna sebagai berikut.



Gambar 3. 11 Alur Tugas Proses *Login*

Tidak ada perubahan yang disarankan dari pihak kursus mengemudi maupun pengguna diluar pihak kursus untuk alur tugas *login* atau daftar akun.



Gambar 3. 12 Alur Tugas Proses Pendaftaran Kelas Kursus


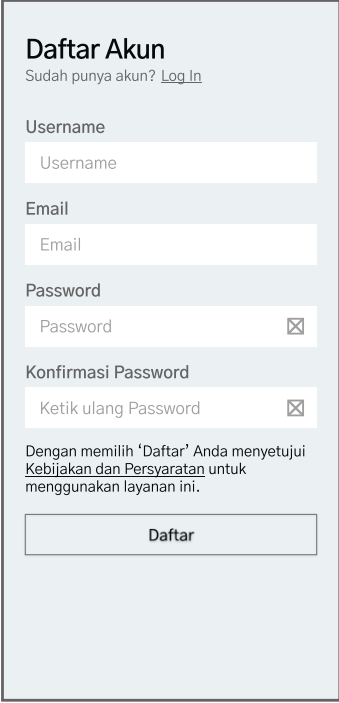
Dapat dilihat pada gambar diatas, setelah calon peserta kursus berada pada tampilan detail kelas kursus, pengguna dapat melihat daftar kelas kursus lain yang ditawarkan oleh instruktur atau lembaga kursus bersangkutan. Hal ini memungkinkan calon peserta untuk melakukan perbandingan antar kelas atau dengan instruktur lain sebelum melakukan pendaftaran kursus.





3.3.1.2. Sketsa Awal Aplikasi

73


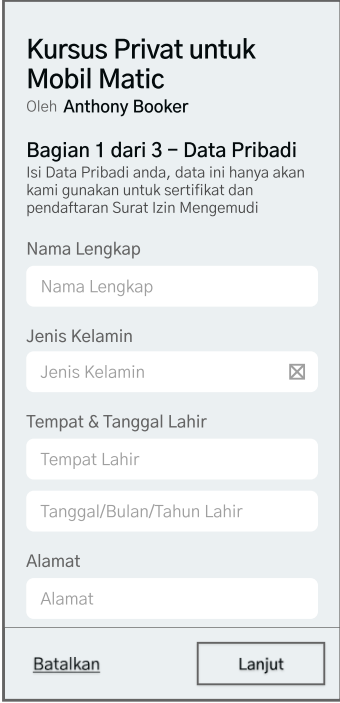
Tabel 3. 17 Sketsa Awal untuk Iterasi Kedua

No.	Sketsa Aplikasi	Keterangan
1	 <p>Tingkatkan Keahlian Mengemudi Anda Bersama Kami</p> <p>Jalanan tidak dapat diprediksi. Jadilah pengemudi yang bertanggung jawab, belajar dengan instruktur berpengalaman.</p> <p>Log In / Daftar</p> <p>Lewati Dulu</p>	<p>Sketsa awal untuk tampilan depan aplikasi tidak mengalami perubahan kecuali teks pada tombol yang dibuat lebih tebal</p>
2	 <p>Daftar Akun</p> <p>Sudah punya akun? Log In</p> <p>Username</p> <p>Email</p> <p>Password</p> <p>Konfirmasi Password</p> <p>Dengan memilih 'Daftar' Anda menyetujui <u>Kebijakan dan Persyaratan</u> untuk menggunakan layanan ini.</p> <p>Daftar</p>	<p>Sketsa awal untuk tampilan <i>form</i> pendaftaran akun disesuaikan dengan masukan yang diterima pada iterasi sebelumnya</p>


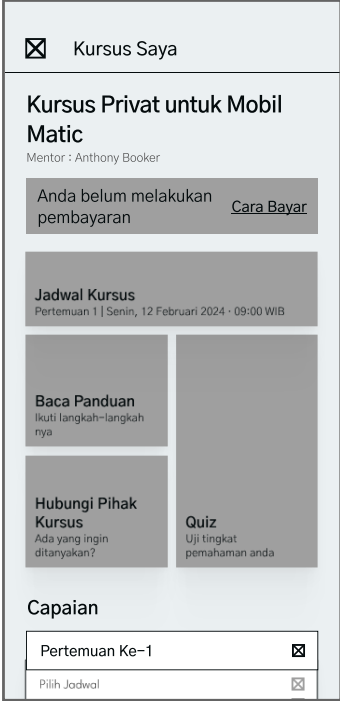
Tabel 3. 18 Sketsa Awal untuk Iterasi Kedua (Lanjutan-1)

No.	Sketsa Aplikasi	Keterangan
3		<p>Sketsa awal untuk tampilan beranda, kolom pencarian dipindah ke <i>navbar</i> dibagian atas aplikasi. Bagian kursus berlangsung diubah agar lebih mudah diproses</p>
4		<p>Sketsa awal untuk tampilan detail kelas kursus disesuaikan dengan masukan yang diterima. Ditambahkan juga fungsi untuk menampilkan semua kelas yang ditawarkan oleh instruktur atau lembaga kursus</p>

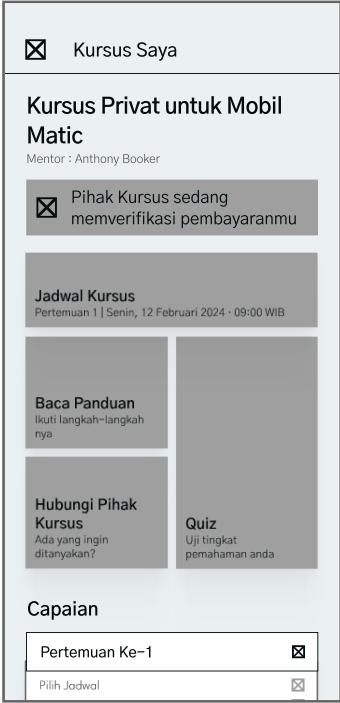

Tabel 3. 19 Sketsa Awal untuk Iterasi Kedua (Lanjutan-2)

No.	Sketsa Aplikasi	Keterangan
5		<p>Sketsa awal untuk tampilan daftar kelas kursus yang dimiliki lembaga kursus atau instruktur terkait</p>
6		<p>Sketsa awal untuk tampilan <i>form</i> pendaftaran kelas kursus sudah disesuaikan dengan masukan yang diterima dari iterasi sebelumnya</p>

Tabel 3. 20 Sketsa Awal untuk Iterasi Kedua (Lanjutan-3)

No.	Sketsa Aplikasi	Keterangan
7		<p>Sketsa awal untuk tampilan pemilihan jadwal kursus, yang sudah disesuaikan berdasarkan masukan yang diterima pada iterasi sebelumnya</p>
8		<p>Sketsa awal untuk tampilan detail kursus aktif yang menunjukkan status pembayaran</p>

Tabel 3. 21 Sketsa Awal untuk Iterasi Kedua (Lanjutan-4)

No.	Sketsa Aplikasi	Keterangan
9		<p>Sketsa awal untuk tampilan detail kursus aktif untuk status pembayaran yang berbeda</p>
10		<p>Sketsa awal untuk tampilan detail kursus aktif dengan asumsi peserta kursus sudah melakukan pembayaran dan sudah diverifikasi.</p>

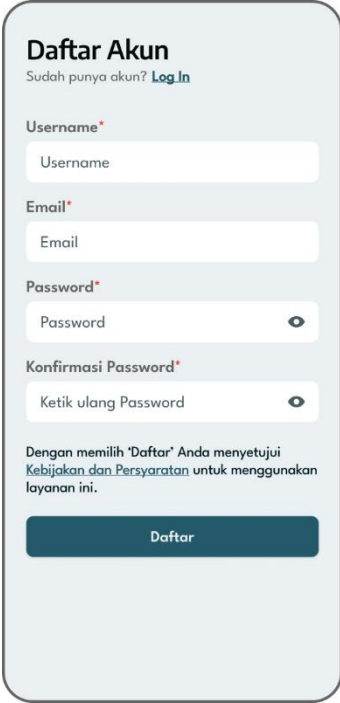

3.3.2. Built Prototype

Untuk *prototype* iterasi kedua, kami melanjutkan hasil dari proses desain singkat sebelumnya. Berikut adalah hasil pembangunan *prototype*.



Tabel 3. 22 Desain *Prototype* untuk Iterasi Kedua

No.	Desain <i>Prototype</i>	Keterangan
1		Tampilan depan aplikasi didasarkan pada sketsa awal pada proses sebelumnya

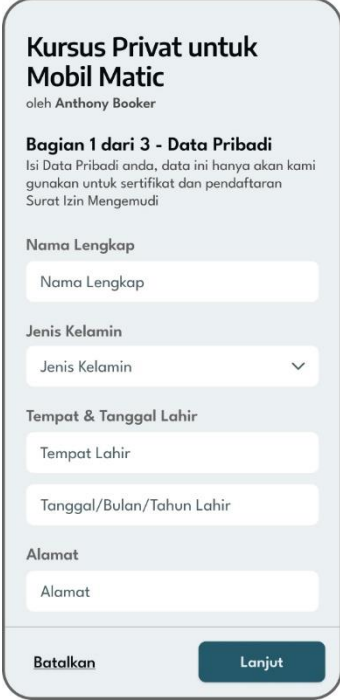

Tabel 3. 23 Desain *Prototype* untuk Iterasi Kedua (Lanjutan-1)

No.	Desain <i>Prototype</i>	Keterangan
2		Tampilan <i>form</i> pendaftaran didasarkan pada sketsa awal dan masukan yang disarankan pada iterasi sebelumnya
3		Tampilan beranda aplikasi dimana kolom pencarian diganti menjadi ikon pencarian dibagian <i>navbar</i>



Tabel 3. 24 Desain *Prototype* untuk Iterasi Kedua (Lanjutan-2)

No.	Desain <i>Prototype</i>	Keterangan
4		<p>Tampilan detail kelas kursus disesuaikan dengan masukan yang diterima pada iterasi sebelumnya</p>
5		<p>Tampilan daftar kelas kursus yang dimiliki lembaga kursus sesuai masukan yang didapatkan iterasi sebelumnya</p>


Tabel 3. 25 Desain *Prototype* untuk Iterasi Kedua (Lanjutan-3)

No.	Desain <i>Prototype</i>	Keterangan
6		Tampilan <i>form</i> pendaftaran kelas kursus berdasarkan hasil sketsa awal pada proses sebelumnya
7		Tampilan untuk pemilihan jadwal kursus, yang sudah disesuaikan berdasarkan masukan yang diterima pada iterasi sebelumnya

Tabel 3. 26 Desain *Prototype* untuk Iterasi Kedua (Lanjutan-4)

No.	Desain <i>Prototype</i>	Keterangan
8	 <p>The screenshot shows a mobile app interface for 'Kursus Saya'. At the top, there's a back arrow and the title 'Kursus Saya'. Below it, the course title 'Kursus Privat untuk Mobil Matic' is displayed, followed by the mentor's name 'Mentor : Anthony Booker'. A prominent red button labeled 'Cara Bayar' (How to Pay) is shown with the text 'Anda belum melakukan pembayaran' (You have not made a payment). Below this, there are several interactive cards: 'Jadwal Kursus' (Course Schedule) with the note 'Kamu belum memilih jadwal kursus' (You haven't selected a course schedule), 'Baca Panduan' (Read Guide) with 'Ikuti langkah-langkahnya' (Follow the steps), 'Hubungi Pihak Kursus' (Contact Course Provider) with 'Ada yang ingin ditanyakan?' (Anything you want to ask?), and a 'Quiz' section with 'Uji tingkat pemahaman anda' (Test your understanding level). At the bottom, there's a 'Capaian' (Achievement) section with a dropdown menu currently showing 'Pertemuan Ke-1' (Session 1) and a 'Pilih Jadwal' (Select Schedule) button.</p>	Tampilan detail kursus aktif yang menunjukkan status pembayaran
9	 <p>This screenshot shows the same 'Kursus Saya' interface as above, but with a different status. The red 'Cara Bayar' button is replaced by an orange banner that says 'Pihak Kursus sedang memverifikasi pembayaranmu' (The course provider is verifying your payment). The rest of the interface, including the course title, mentor name, 'Jadwal Kursus' card, 'Baca Panduan' card, 'Hubungi Pihak Kursus' card, 'Quiz' section, and 'Capaian' section, remains identical to the previous prototype.</p>	Tampilan detail kursus aktif untuk status pembayaran yang berbeda

Tabel 3. 27 Desain *Prototype* untuk Iterasi Kedua (Lanjutan-5)

No.	Desain <i>Prototype</i>	Keterangan
10		Tampilan detail kursus aktif akhir yang sudah disesuaikan dengan masukan yang diterima pada iterasi sebelumnya

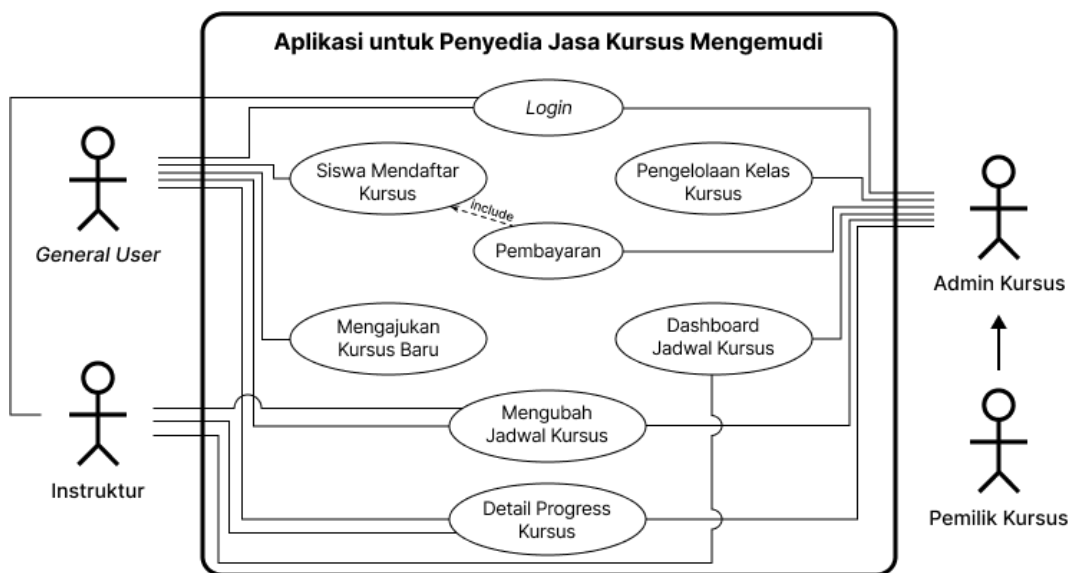
3.3.3. *Customer Evaluation*

Setelah proses pembangunan *prototype* selanjutnya kami melakukan pengujian terhadap hasil desain *prototype* dan alur pengguna yang dirancang. Pengujian ini dilakukan kepada pengguna *prototype* yang sama dengan yang menguji *prototype* iterasi pertama. Tujuannya adalah untuk mendapatkan masukan dan umpan balik dari pengguna terkait dengan desain dan alur pengguna yang baru. Masukan dan umpan balik ini kemudian akan digunakan untuk menyempurnakan *prototype* sebelum memasuki tahapan pengembangan selanjutnya. Namun, setelah kami kembali melakukan pengujian, pengguna *prototype* lebih menyukai hasil desain *prototype* iterasi kedua, dan *prototype* dirasa tidak membutuhkan perbaikan.

3.3.4. Review & Updation

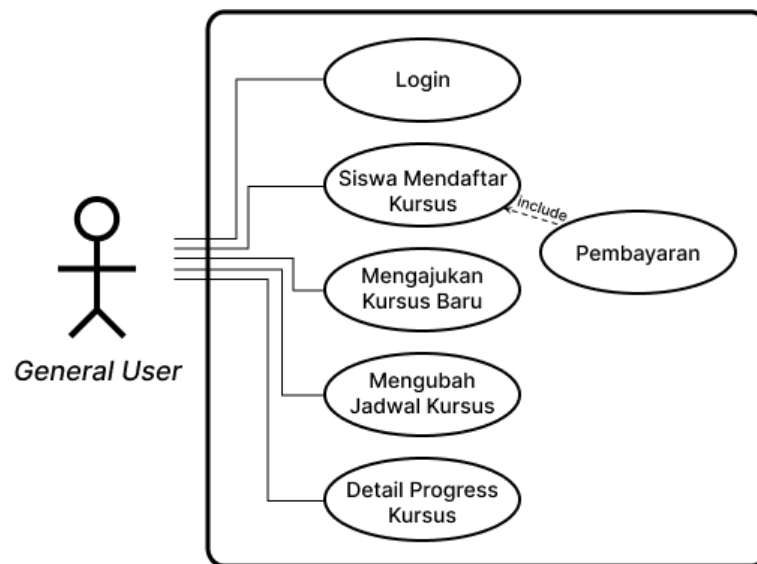
Dari hasil pengujian awal sebelumnya, pengguna *prototype* tidak memberikan masukan atau umpan balik terhadap *prototype* iterasi kedua. Hal ini menunjukkan bahwa pengguna *prototype* merasa puas dengan desain dan alur pengguna yang baru. Oleh karena itu, langkah selanjutnya adalah melakukan finalisasi desain untuk aplikasi yang kami rancang, tujuannya adalah agar desain yang sudah dibuat nantinya merupakan desain yang sama yang dihasilkan pada saat proses pengembangan, atau dalam istilah pengembangan perangkat lunak sering disebut dengan *handoff*.

3.4. Design



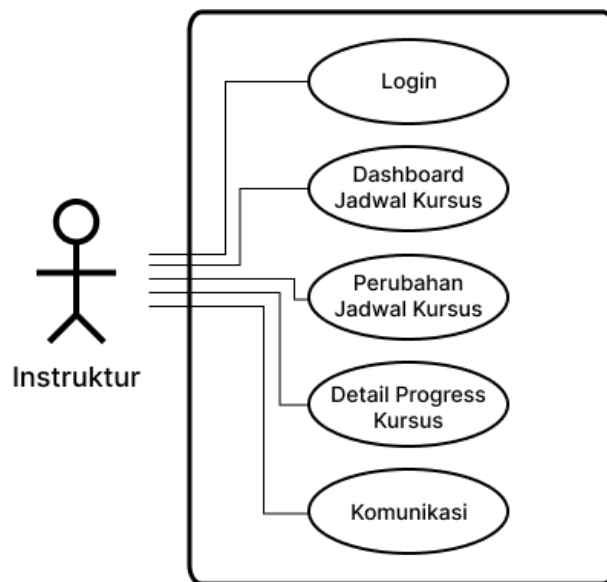
Gambar 3. 14 Use Case Diagram Aplikasi

Setelah melewati proses analisa kebutuhan dan desain *prototype* singkat diatas, langkah selanjutnya adalah menerjemahkannya ke diagram-diagram UML. Hal ini ditujukan untuk memberikan alur yang lebih jelas tentang interaksi-interaksi yang dilakukan user dengan sistem, serta sekali lagi memeriksa apakah semua fungsi yang disebutkan diatas sudah terakomodir semuanya. Untuk memecah kompleksitas dan menghindari kesalahpahaman, kami akan menjelaskan setiap *use case* dengan *Activity Diagram* dan *Use Case Scenario*.



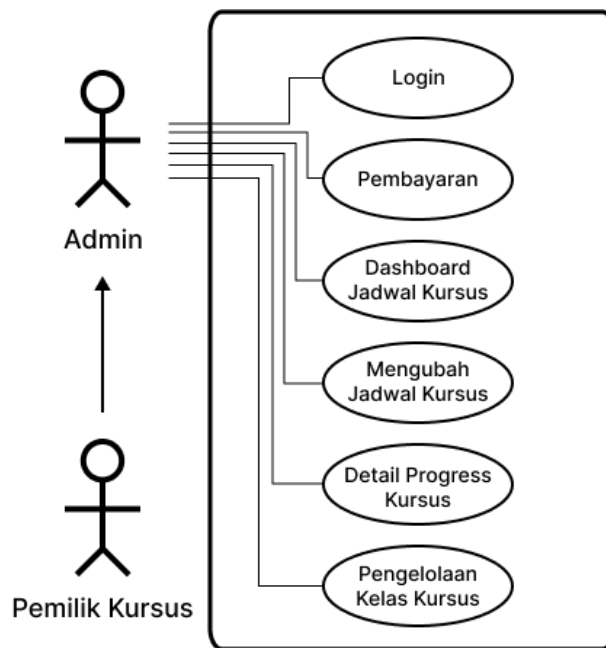
Gambar 3. 15 Use Case untuk pengguna *General User*

Kami mengelompokkan pengguna aplikasi kami menjadi 3 kelompok. Kelompok pertama adalah *General User*, yang mewakili masyarakat awam. Pengguna Umum dapat memanfaatkan aplikasi kami untuk mengikuti kursus mengemudi dan meningkatkan keterampilan berkendara mereka. Atau, mereka juga dapat menawarkan jasa mereka sebagai penyedia kursus mengemudi atau instruktur, membantu orang lain yang ingin belajar mengemudi. Pada gambar diatas, dapat dilihat bahwa *General User*, setidaknya dapat melakukan 7 operasi pada aplikasi kami nantinya, *Login* atau daftar sebagai pengguna aplikasi dan aktivitas pengelolaan akun yang lainnya, kemudian mengisi formulir pendaftaran kursus, melakukan pembayaran kursus, mendaftarkan diri sebagai penyedia jasa kursus baru, melakukan perubahan jadwal kursus yang diikuti, melihat detail progress kursus, dan yang terakhir melakukan komunikasi dengan pihak kursus, baik bagian Admin atau Instruktur kursus terkait.



Gambar 3. 16 *Use Case* untuk pengguna Instruktur

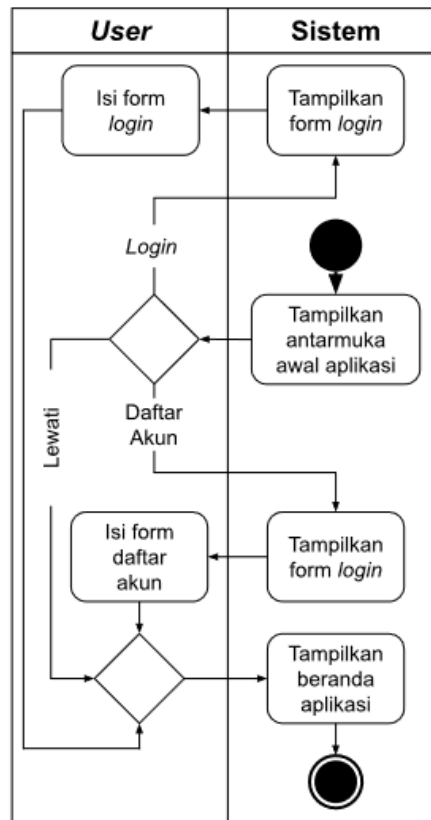
Kelompok pengguna selanjutnya adalah Instruktur, Instruktur mengemudi adalah pihak yang melakukan pertemuan langsung dengan siswa atau *General User* pada saat proses kursus mengemudi nantinya. Untuk Instruktur, operasi-operasi yang dapat dilakukan di dalam aplikasi kami adalah proses *Login* dan aktivitas pengelolaan akun yang lain, selanjutnya menampilkan jadwal kursus per hari, mengajukan perubahan jadwal kursus, melihat detail progress kursus siswa terkait, dan melakukan komunikasi antar pihak Admin atau Siswa yang berada di bawah tanggung jawabnya.



Gambar 3. 17 Use Case untuk pengguna Admin/Pemilik Kursus

General User dapat menjadi Pemilik Kursus atau Admin jasa kursus, apabila pengguna bersangkutan sudah menyelesaikan proses pendaftaran jasa kursus baru, dengan menjadi Pemilik Kursus atau Admin jasa kursus, pengguna dapat melakukan operasi-operasi seperti *Login* dan aktivitas-aktivitas pengelolaan akun lainnya, melihat formulir kursus yang diisi oleh calon siswa, mengubah status pembayaran, melihat jadwal kursus per hari, mengajukan perubahan jadwal kursus kepada Siswa dan Instruktur, melihat detail progress kursus siswa, melakukan aktivitas pengelolaan kelas kursus yang ditawarkan seperti mengubah harga, menetapkan kuota kelas, dan proses-proses lain, serta Pemilik/Admin Kursus dapat melakukan komunikasi dengan Instruktur atau Siswa kursus.

3.4.1. Login / Daftar Akun



Gambar 3. 18 Activity Diagram untuk proses Login

Untuk alur *Login*, sistem mulai dengan menanyakan apakah pengguna sudah punya akun? Jika ya, pengguna bisa langsung melakukan *Login*, jika tidak sistem mempunyai 2 opsi lanjutan, apakah pengguna ingin daftar akun sekarang atau pengguna bisa melakukan aksi ini nanti ketika sistem membutuhkan penyimpanan data untuk akun bersangkutan.

Tabel 3. 28 Use Case Scenario untuk proses Login

Kode Use Case	UC_login_untuk_semua_user
Nama Use Case	<i>Login</i>
Aktor	<i>General User, Pemilik / Admin, Instruktur</i>

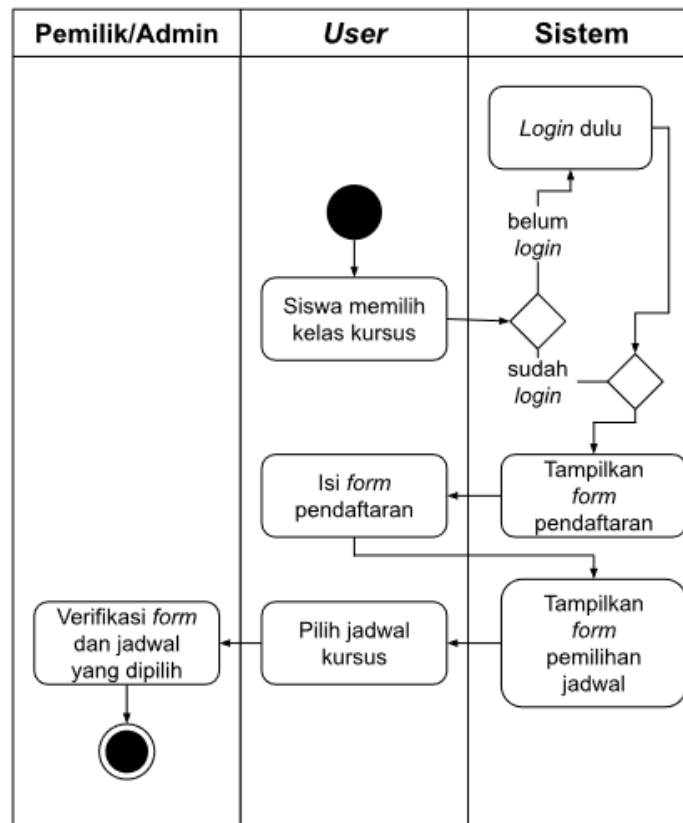
Tabel 3. 29 *Use Case Scenario* untuk proses *Login* (Lanjutan-1)

Deskripsi	Skenario untuk menyimpan data dari segala aktivitas yang dilakukan oleh Aktor	
Kondisi Awal	Sistem baru selesai memuat tampilan awal	
Kondisi Akhir	Aktor berada di halaman beranda/ <i>dashboard</i>	
Alur Kejadian Normal	Aktor	Sistem
		1. Sistem menampilkan tiga tombol (<i>Login</i> , <i>Daftar</i> , dan <i>Nanti</i>)
	2. Aktor memilih tombol <i>Login</i>	
		3. Sistem menampilkan <i>form Login</i>
	4. Aktor memasukkan data akun yang tersimpan di database	
		5. Sistem memverifikasi data yang diinputkan Aktor
		6. Sistem mengarahkan Aktor
	2a. Aktor memilih tombol “Daftar” 2b. Aktor memilih tombol “Nanti”	

Tabel 3. 30 *Use Case Scenario* untuk proses *Login* (Lanjutan-2)

	Aktor	Sistem
		6. Sistem mengarahkan Aktor
Alur Kejadian Alternatif	2a. Aktor memilih tombol “Daftar” 2b. Aktor memilih tombol “Nanti”	
		3a. Sistem menampilkan <i>form</i> daftar akun 3b. Sistem mengubah peran Aktor dari <i>General User</i> menjadi Siswa
	4a. Aktor memasukkan data akun baru 4c. Aktor memasukkan data yang tidak ada di database	4b. Sistem mengalihkan Aktor ke beranda/ <i>dashboard</i>
		5a. Sistem menyimpan data baru yang diinputkan Aktor 5c. Sistem tidak menemukan data yang diinputkan Aktor
		6a. Sistem mengalihkan Aktor ke beranda/ <i>dashboard</i> 6c. Sistem mengalihkan Aktor ke tampilan awal dan mengembalikan pesan <i>error</i> ke Aktor

3.4.2. Siswa Mendaftar Kursus



Gambar 3. 19 Activity Diagram untuk proses Siswa Mendaftar Kursus

Untuk alur mendaftar kursus, Siswa memilih kelas kursus yang dimaksud terlebih dahulu, selanjutnya sistem akan melakukan pengecekan apakah pengguna sudah melakukan proses *login* sebelumnya. Proses *login* diperlukan agar sistem mengetahui data yang akan di-entry oleh pengguna ini tersimpan bersama dengan data akun pengguna.

Tabel 3. 31 Use Case Scenario untuk proses Siswa Mendaftar Kursus

Kode Use Case	UC_siswa_daftar_kursus
Nama Use Case	Siswa Mendaftar Kursus
Aktor	Siswa & Pemilik/Admin

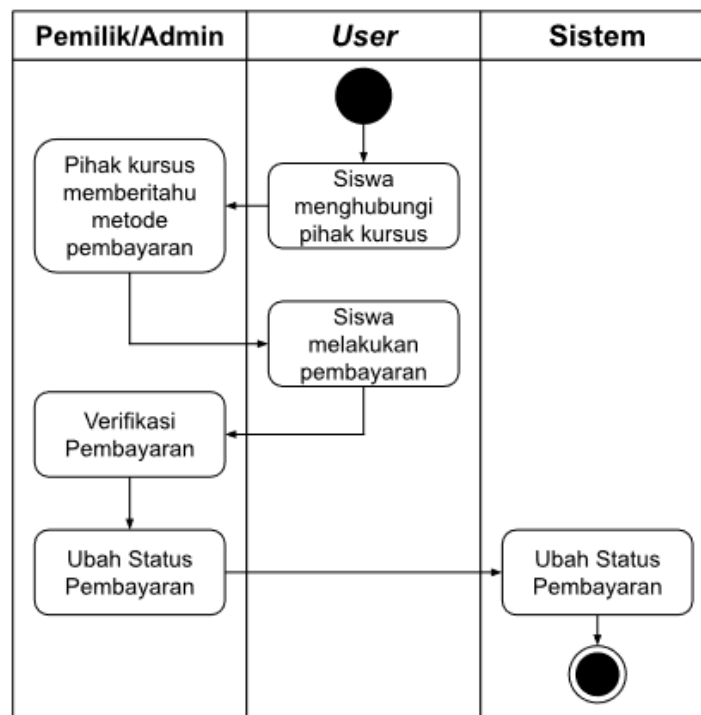
Tabel 3. 32 *Use Case Scenario* untuk proses Siswa Mendaftar Kursus (Lanjutan-1)

Deskripsi	Skenario untuk mendaftarkan diri mengikuti kursus mengemudi	
Kondisi Awal	Calon siswa sudah memilih kelas kursus yang diinginkan	
Kondisi Akhir	Calon siswa menyelesaikan pengisian formulir pendaftaran	
Alur Kejadian Normal	Aktor	Sistem
	1. Calon Siswa menekan tombol “Daftar Kelas”	
		2. Sistem mengalihkan Calon Siswa ke formulir pendaftaran kursus
	3. Calon Siswa mengisi formulir pendaftaran dengan lengkap	
		4. Sistem menyimpan <i>entry</i> yang dilakukan Calon Siswa
		5. Sistem mengirimkan pemberitahuan ke Pemilik/Admin
	6. Pemilik/Admin memverifikasi formulir yang diisi calon siswa	
Alur Kejadian Alternatif	3. Calon Siswa tidak jadi meneruskan pendaftaran	

Tabel 3. 33 *Use Case Scenario* untuk proses Siswa Mendaftar Kursus (Lanjutan-2)

	Aktor	Sistem
		4. Sistem mengalihkan Calon Siswa ke beranda/ <i>dashboard</i>

3.4.3. Pembayaran



Gambar 3. 20 *Activity Diagram* untuk proses Pembayaran

Pada sistem yang kami kembangkan proses pembayaran terjadi di luar sistem, sistem hanya memberitahukan kedua pihak status pembayaran saat ini.

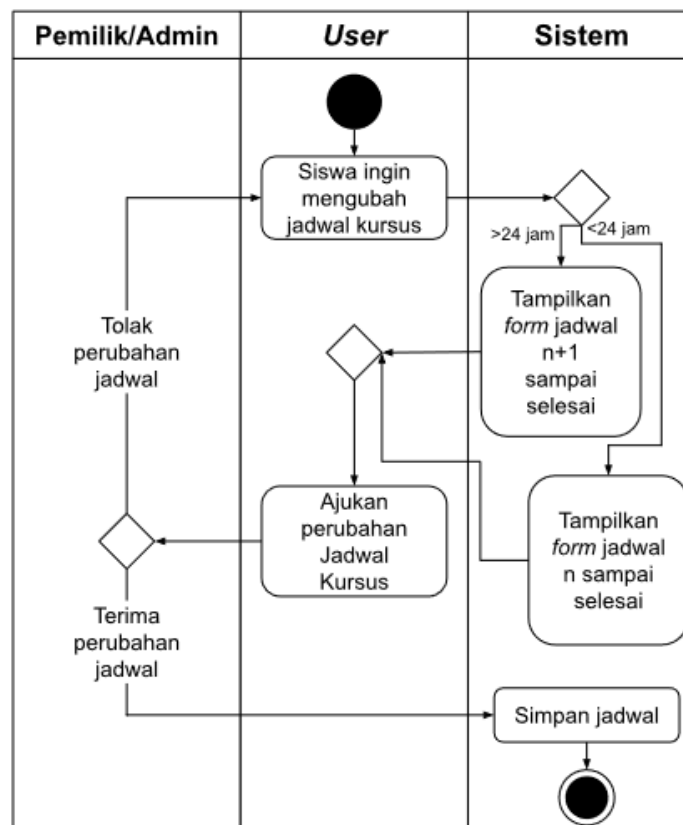
Tabel 3. 34 *Use Case Scenario* untuk proses Pembayaran

Kode Use Case	UC_pembayaran	
Nama Use Case	Pembayaran	
Aktor	Siswa & Pemilik/Admin	
Deskripsi	Skenario pemberitahuan status pembayaran	
Kondisi Awal	Calon siswa baru saja menyelesaikan <i>form</i> pendaftaran	
Kondisi Akhir	Calon siswa mendapatkan informasi status pembayaran	
Alur Kejadian Normal	Aktor	Sistem
		1. Sistem mengarahkan Calon siswa ke halaman detail kursus
	2. Siswa menghubungi pihak kursus menanyakan tentang proses pembayaran	
	3. Siswa melakukan proses pembayaran	
	4. Pemilik/Admin memverifikasi pembayaran	
	5. Pemilik/Admin mengubah status pembayaran	

Tabel 3. 35 *Use Case Scenario* untuk proses Pembayaran (Lanjutan-1)

Alur Kejadian Normal	Aktor	Sistem
		6. Sistem menampilkan status pembayaran yang dipilih Admin
Alur Kejadian Alternatif	-	-

3.4.4. Mengubah Jadwal Kursus



Gambar 3. 21 *Activity Diagram* untuk proses Mengubah Jadwal Kursus

Jika siswa ingin mengubah jadwal, sistem masih bisa melanjutkan proses apabila pertemuan selanjutnya lebih dari 24 jam, jadwal yang diubah nantinya akan mengubah pertemuan selanjutnya sampai pertemuan akhir. Sebagai contoh, siswa memilih kursus dengan 5 pertemuan, pada pertemuan ke-3 siswa tidak dapat hadir

sesuai dengan jadwal, jika siswa memilih untuk melakukan perubahan jadwal, maka siswa dapat mengubah jadwal kursus dari pertemuan ke-3 sampai pertemuan ke-5.

Tabel 3. 36 *Use Case Scenario* untuk proses Mengubah Jadwal Kursus

Kode Use Case	UC_mengubah_jadwal_kursus	
Nama Use Case	Mengubah Jadwal Kursus	
Aktor	Siswa & Pemilik/Admin	
Deskripsi	Skenario pengajuan perubahan jadwal	
Kondisi Awal	Siswa berada di halaman detail <i>progress</i> kursus	
Kondisi Akhir	Siswa berhasil mengubah rangkaian jadwal kursus	
Alur Kejadian Normal	Aktor	Sistem
	1. Aktor menekan tombol 'Jadwal Kursus'	
		2. Sistem mengalihkan Aktor ke halaman jadwal kursus siswa terkait
	3. Aktor menekan tombol 'Ubah Jadwal'	
		4. Sistem memeriksa apakah pertemuan selanjutnya < 24 jam

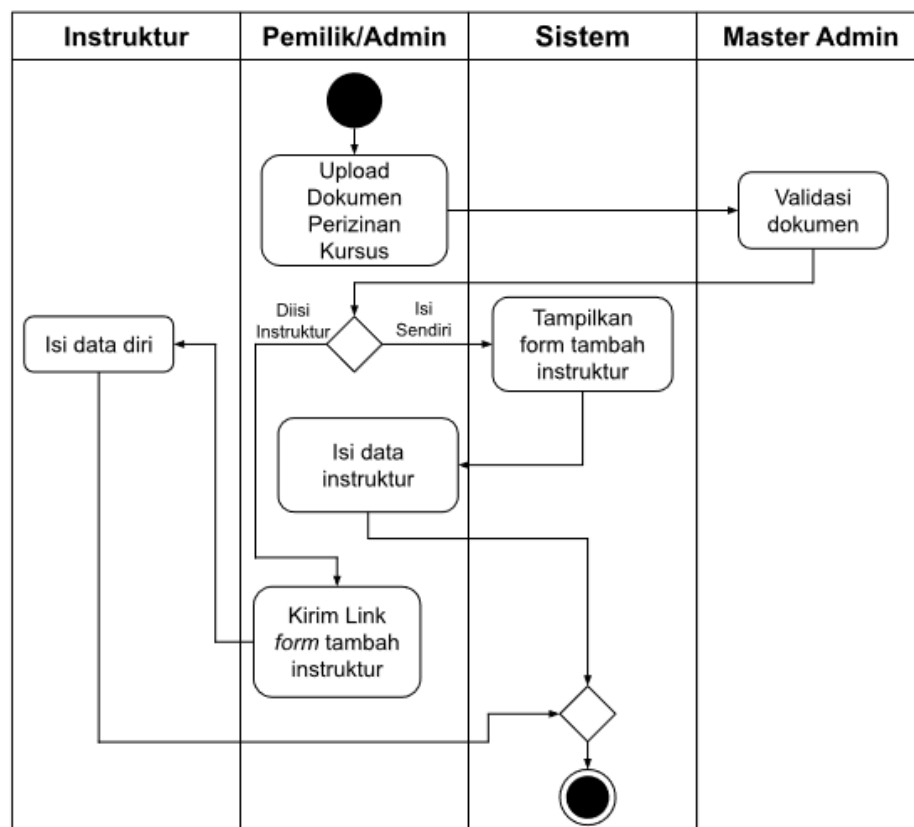
Tabel 3. 37 *Use Case Scenario* untuk proses Mengubah Jadwal Kursus (Lanjutan-1)

	Aktor	Sistem
		<p>5a. Jika pertemuan selanjutnya < 24 jam, jadwal yang dapat diubah dimulai dari pertemuan setelah pertemuan mendatang</p> <p>5b. Jika pertemuan selanjutnya > 24 jam, perubahan jadwal dapat dimulai dari pertemuan mendatang</p>
	6. Aktor menyesuaikan jadwal yang tersedia	
Alur Kejadian Alternatif	Aktor	Sistem
	7. Aktor menekan tombol 'Ajukan Perubahan Jadwal'	
		8. Sistem mengirimkan pemberitahuan ke Aktor lain untuk dikonfirmasi pengajuan jadwal yang dikirim
	9. Aktor lainnya mengkonfirmasi perubahan jadwal	

Tabel 3. 38 *Use Case Scenario* untuk proses Mengubah Jadwal Kursus (Lanjutan-2)

	Aktor	Sistem
		10. Sistem mengirimkan pemberitahuan penolakan pengajuan jadwal ke Aktor

3.4.5. Mengajukan Kursus Baru



Gambar 3. 22 *Activity Diagram* untuk proses Mengajukan Kursus Baru

Seperti proses-proses sebelumnya, karena proses ini memerlukan sistem untuk menyimpan data yang di-*entry* oleh pengguna, sebelum melakukan proses upload dokumen perizinan kursus, sistem akan memeriksa apakah pengguna sudah *login*? Jika sudah, proses akan berlanjut, jika belum, sistem akan mengarahkan pengguna ke halaman *login*.

Tabel 3. 39 *Use Case Scenario* untuk proses Mengajukan Kursus Baru

Kode Use Case	UC_mengajukan_kursus_baru	
Nama Use Case	Mengajukan Kursus Baru	
Aktor	Instruktur, <i>General User</i> (Calon Pemilik) & Master Admin	
Deskripsi	Skenario untuk mendaftarkan jasa kursus mengemudi baru	
Kondisi Awal	Calon Pemilik berada di halaman beranda/ <i>dashboard</i>	
Kondisi Akhir	Pemilik berada pada halaman beranda/ <i>dashboard</i>	
Alur Kejadian Normal	Aktor	Sistem
	1. Calon Pemilik berpindah ke halaman profil	
		2. Sistem menampilkan data dan tombol-tombol yang nantinya ditampilkan di halaman profil
	3. Calon Pemilik menekan menu 'Ingin jadi Instruktur?'	
		4. Sistem mengalihkan Aktor ke halaman <i>upload</i> dokumen perizinan

Tabel 3. 40 *Use Case Scenario* untuk proses Mengajukan Kursus Baru (Lanjutan-1)

	Aktor	Sistem
	5. Calon Pemilik mengupload dokumen perizinan penyelenggaraan kursus	
		6. Master Admin memverifikasi keaslian dokumen yang diupload oleh Calon Pemilik
		7. Sistem mengirim pesan berhasil ke Calon Pemilik kemudian mengarahkan Calon Pemilik untuk mengisi data Instruktur
		8. Sistem memberikan opsi kepada pengguna untuk mengisi data Instruktur sendiri atau diisi oleh Calon Instruktur bersangkutan
	9a. Calon Pemilik memilih mengisi data Instruktur sendiri 9b. Calon Pemilik memilih agar diisi oleh Calon Instruktur bersangkutan	

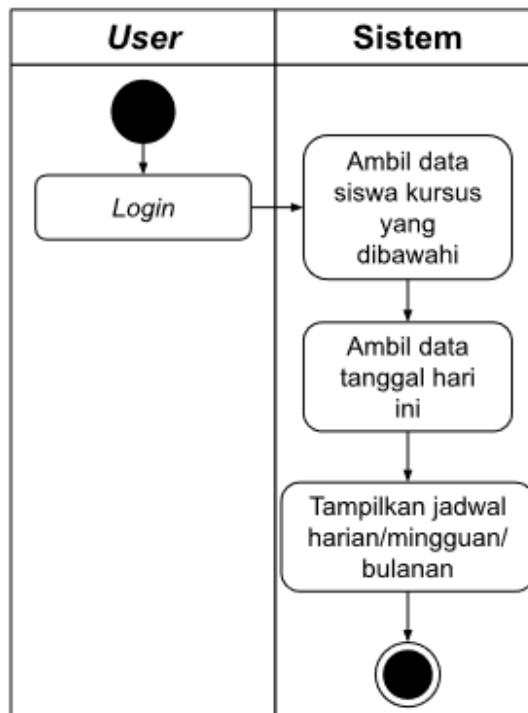
Tabel 3. 41 *Use Case Scenario* untuk proses Mengajukan Kursus Baru (Lanjutan-2)

	Aktor	Sistem
	10a. Calon Pemilik mengisi semua data Instruktur dengan lengkap 10b. Calon Pemilik mengirimkan link <i>form</i> penambahan Instruktur ke Calon Instruktur	
	11b. Calon Instruktur menerima <i>link</i> undangan mengajar	11a. Sistem menyimpan <i>entry</i> data dari Calon Pemilik, kemudian mengarahkan Calon Pemilik ke tampilan beranda/ <i>dashboard</i>
		12b. Sistem menampilkan halaman pendaftaran akun untuk diisi Calon Instruktur
	13b. Calon Instruktur mengisi <i>form</i> penambahan Instruktur	
		14b. Sistem menyimpan <i>entry</i> yang dilakukan Instruktur

Tabel 3. 42 *Use Case Scenario* untuk proses Mengajukan Kursus Baru (Lanjutan-3)

Alur Kejadian Alternatif	Aktor	Sistem
		6. <i>System</i> admin menolak keaslian dokumen atau dokumen yang di- <i>upload</i> dianggap kurang jelas
		7. Sistem mengirim pesan gagal ke Calon Pemilik dan mengarahkan untuk melakukan <i>upload</i> ulang
	10a. Calon Pemilik tidak mengisi <i>form</i> dengan lengkap atau membatalkan pengisian	
	13b. Instruktur tidak mengisi <i>form</i> dengan lengkap atau membatalkan pengisian	11a. Sistem mengalihkan Calon Pemilik ke halaman profil
		14b. Sistem mengalihkan Calon Instruktur ke halaman profil

3.4.6. Dashboard Jadwal Kursus



Gambar 3. 23 Activity Diagram untuk proses menampilkan *Dashboard* Jadwal Kursus

Untuk menampilkan *dashboard* pihak kursus (pemilik/admin kursus dan instruktur) harus menyelesaikan alur *Login* terlebih dulu, selanjutnya sistem akan menampilkan jadwal kursus yang dikelompokkan berdasarkan tanggal, dimulai dengan hari ini, esok hari, dan hari selanjutnya. Yang membedakan adalah jika instruktur hanya bisa melihat jadwal untuk dirinya sendiri, sedangkan pemilik/admin dapat melihat seluruh jadwal instruktur yang dibawahinya. Jumlah hari yang ditampilkan dapat diatur untuk menghindari pengguna kewalahan memproses informasi.

Tabel 3. 43 Use Case Scenario untuk proses menampilkan *Dashboard* Jadwal Kursus

Kode Use Case	UC_tampilkan_jadwal_kursus
Nama Use Case	Menampilkan <i>dashboard</i> jadwal kursus

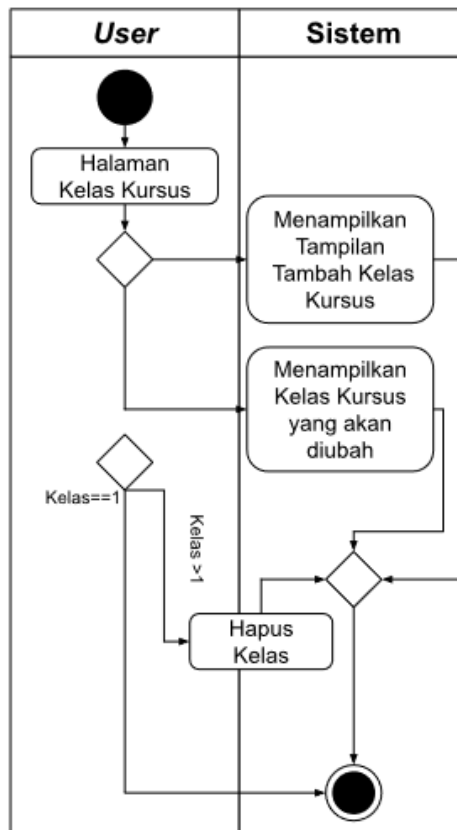
Tabel 3. 44 *Use Case Scenario* untuk proses menampilkan *Dashboard* Jadwal Kursus (Lanjutan-1)

Aktor	Pemilik / Admin, Instruktur	
Deskripsi	Skenario untuk menampilkan jadwal kursus per hari	
Kondisi Akhir	Sistem menampilkan semua jadwal sesuai batas hari yang ditentukan Aktor	
Alur Kejadian Normal	Aktor	Sistem
		1. Sistem mengumpulkan data siswa yang kursus dengan Aktor
		2. Sistem secara <i>default</i> , menampilkan jadwal untuk 3 hari (hari ini, esok hari, dan hari setelahnya)
		3. Sistem memberikan opsi lain untuk menampilkan jadwal dalam 15 hari dan 30 hari
Alur Kejadian Alternatif	Aktor	Sistem
	4a. Aktor memilih opsi untuk tampilkan jadwal dalam 15 hari	
		5a. Sistem menampilkan jadwal dimulai dengan hari ini sampai 14 hari kedepan

Tabel 3. 45 *Use Case Scenario* untuk proses menampilkan *Dashboard* Jadwal Kursus (Lanjutan-2)

	Aktor	Sistem
	4b. Aktor memilih opsi untuk tampilkan jadwal dalam 30 hari	
		5b. Sistem menampilkan jadwal dimulai dengan hari ini sampai 29 hari kedepan

3.4.7. Pengelolaan Kelas Kursus



Gambar 3. 24 Activity Diagram untuk proses Mengelola Kelas Kursus

Pemilik/Admin dari suatu kursus dapat melakukan penambahan atau menonaktifkan/hapus kelas kursus yang ada di halaman kursus mereka selama masih ada 1 kelas kursus tersisa. Selain itu, pemilik/admin ini dapat mengubah informasi-informasi yang berkaitan dengan kelas kursus, seperti deskripsi kursus, lama kursus, kategori kursus, benefit, harga, dan lain-lain.

Tabel 3. 46 Use Case Scenario untuk Mengelola Kelas Kursus

Kode Use Case	UC_pengelolaan_kelas_kursus
Nama Use Case	Pengelolaan Kelas Kursus
Aktor	Pemilik / Admin, Instruktur

Tabel 3. 47 *Use Case Scenario* untuk Mengelola Kelas Kursus (Lanjutan-1)

Deskripsi	Skenario untuk menambah, mengubah, dan menghapus/menonaktifkan kelas kursus yang ditawarkan	
Kondisi Awal	Aktor berada di halaman kursus	
Kondisi Akhir	Aktor berhasil menambah kelas kursus baru	
Alur Kejadian Normal	Aktor	Sistem
	1. Aktor menekan tombol “Tambah kelas” pada halaman kursus	
		2. Sistem mengalihkan Aktor ke <i>form</i> tambah kelas
	3. Aktor mengisi semua <i>form</i> dengan lengkap	3. Sistem memverifikasi bahwa semua <i>form</i> terisi
	4. Aktor menekan tombol “Tawarkan kelas”	
		5. Sistem menyimpan <i>entry</i> yang dilakukan Aktor
		6. Sistem mengarahkan Aktor ke halaman kursus dan menampilkan kelas yang baru ditambahkan pada halaman kursus

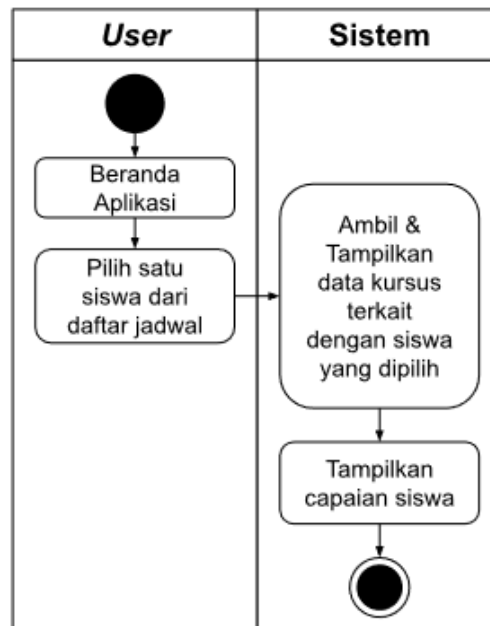
Tabel 3. 48 *Use Case Scenario* untuk Mengelola Kelas Kursus (Lanjutan-2)

Alur Kejadian Alternatif	Aktor	Sistem
		<p>1a. Sistem menampilkan tombol “Ubah kelas” di masing-masing kelas kursus</p> <p>1b. Sistem menampilkan tombol “Nonaktifkan kelas” jika kondisi terpenuhi</p>
	<p>2a. Aktor menekan tombol “Ubah kelas” di kelas Kursus yang dipilih</p> <p>2b. Aktor menekan tombol “Nonaktifkan kelas” di kelas Kursus yang dipilih</p>	
	3c. Aktor belum mengisi semua <i>form</i> dengan lengkap	<p>3a. Sistem mengalihkan Aktor ke <i>form</i> ubah kelas</p> <p>3b. Sistem memeriksa apakah kelas kursus yang ditawarkan > 1</p> <p>3c. Sistem membuat tombol “Tawarkan kelas” tidak bisa diklik</p>
	<p>4a. Aktor mengubah semua informasi yang ingin diubah</p> <p>4c. Aktor menekan tombol “Simpan Draft”</p>	<p>4b-0, jika kelas kursus hanya 1, sistem akan mengembalikan pesan error dan menyarankan apakah Aktor lebih memilih untuk</p>

Tabel 3. 49 *Use Case Scenario* untuk Mengelola Kelas Kursus (Lanjutan-3)

	Aktor	Sistem
		menutup kursus 4b-1, jika kelas kursus > 1, sistem akan menonaktifkan kelas yang dipilih
	5a. Aktor menekan tombol “Ubah kelas”	5c. Sistem menyimpan <i>entry</i> yang dilakukan Aktor
		6a. Sistem menyimpan <i>entry</i> yang dilakukan Aktor 6c. Sistem mengalihkan Aktor ke halaman kursus
		7a. Sistem mengalihkan Aktor ke halaman kursus

3.4.8. Menampilkan Detail Progress Kursus untuk Pihak Kursus



Gambar 3. 25 Activity Diagram untuk proses Menampilkan Detail Progress Kursus untuk Pihak Kursus

Pihak kursus dapat melihat detail dari kursus dari siswa yang dibawah. Halaman detail *progress* ini menunjukkan instruktur di pertemuan ke berapakah siswa saat ini, serta apakah siswa sudah membaca teori yang diberikan, apakah *quiz* yang diberikan sudah dikerjakan, dan capaian-capaian lain.

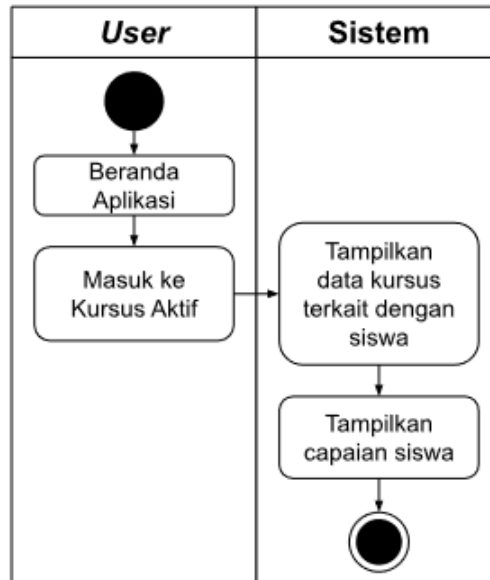
Tabel 3. 50 Use Case Scenario untuk proses Menampilkan Detail Progress Kursus untuk Pihak Kursus

Kode Use Case	UC_detail_progress_kursus_instruktur
Nama Use Case	Detail <i>progress</i> kursus
Aktor	Pemilik / Admin, Instruktur
Deskripsi	Skenario untuk menampilkan detail kursus aktif
Kondisi Awal	Aktor berada di <i>dashboard</i>

Tabel 3. 51 *Use Case Scenario* untuk proses Menampilkan Detail Progress Kursus untuk Pihak Kursus (Lanjutan-1)

Kondisi Akhir	Aktor memperoleh informasi detail kursus dari sistem	
Alur Kejadian Normal	Aktor	Sistem
	1. Aktor memilih salah satu siswa dari daftar jadwal yang ditampilkan di <i>dashboard</i>	
		2. Sistem menampilkan data kursus terkait dengan siswa yang dipilih Aktor
		3. Sistem menampilkan daftar capaian yang sudah dilakukan oleh siswa
Alur Kejadian Alternatif	-	-

3.4.9. Menampilkan Detail Progress Kursus untuk Siswa



Gambar 3. 26 Activity Diagram untuk proses Menampilkan Detail Progress Kursus untuk Siswa

Siswa dengan kursus aktif dapat melihat detail dari kursus yang diikuti. Halaman detail *progress* ini memberikan siswa kemungkinan untuk mempelajari teori mengemudi lebih lanjut dengan memilih ‘Baca Panduan’ dan mengerjakan *quiz* yang diberikan oleh sistem.

Tabel 3. 52 *Use Case Scenario* untuk proses Menampilkan Detail Progress Kursus untuk Pihak Kursus

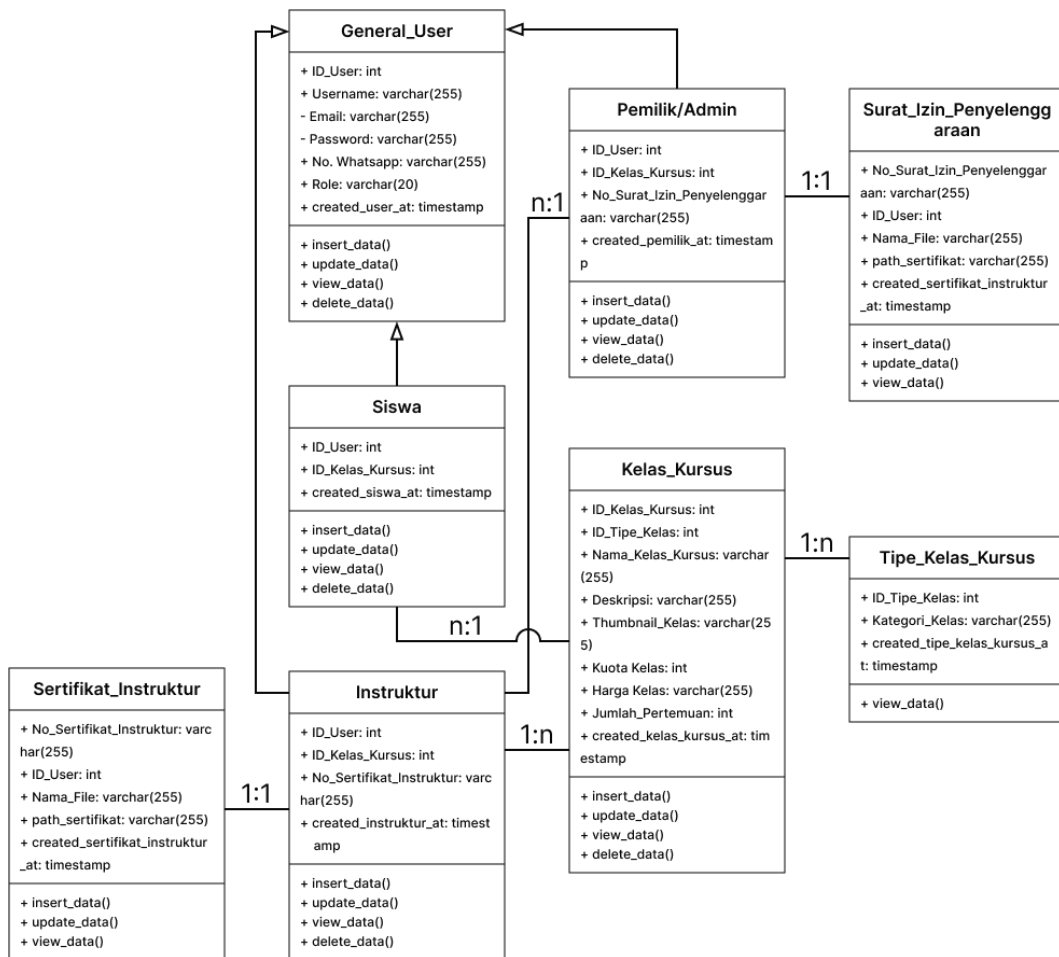
Kode Use Case	UC_detail_progress_kursus_siswa
Nama Use Case	Detail <i>progress</i> kursus
Aktor	<i>General User</i> (Siswa)
Deskripsi	Skenario untuk menampilkan detail kursus aktif
Kondisi Awal	Aktor berada di <i>dashboard</i>

Tabel 3. 53 *Use Case Scenario* untuk proses Menampilkan Detail Progress Kursus untuk Pihak Kursus (Lanjutan-1)

Kondisi Akhir	Aktor bisa memilih untuk perubahan jadwal, membaca panduan teori mengemudi, menyelesaikan Quiz, dan proses-proses lain	
Alur Kejadian Normal	1. Aktor menekan ‘Progress Kursus’ pada bagian atas <i>dashboard</i>	
		2. Sistem menampilkan informasi kursus yang dijalani siswa saat ini
		3. Sistem menampilkan capaian-capaian yang sudah dilakukan oleh siswa
Alur Kejadian Alternatif	1. Aktor berpindah ke halaman profil	
	2. Aktor menekan menu ‘Kursus Saya’	
		3. Sistem menampilkan informasi kursus yang dijalani siswa saat ini
		4. Sistem menampilkan capaian-capaian yang sudah dilakukan oleh siswa

3.4.10. *Class Diagram*

Untuk rancangan desain *database* dari aplikasi, kami menggambarkan struktur *database* dengan *class diagram*. Pada *class diagram* nantinya akan berisi atribut-atribut yang dimiliki masing-masing Aktor dan objek-objek lain, metode-metode yang dapat dieksekusi, dan relasi antar Aktor dan objek pada sistem nantinya. Berikut adalah rancangan *class diagram* untuk aplikasi penyedia jasa kursus mengemudi.



Gambar 3. 27 *Class Diagram* Aplikasi untuk Penyedia Jasa Kursus Mengemudi

Dapat dilihat pada gambar diatas, *database* aplikasi untuk penyedia jasa kursus mengemudi nantinya akan terdapat 8 tabel, 4 tabel utama yang terdiri dari tabel Siswa, Pemilik/Admin, Instruktur, dan Kelas Kursus. 1 tabel untuk menggeneralisasi tabel-tabel pengguna, tabel yang dimaksud adalah tabel Siswa, Instruktur, dan Pemilik/Admin. 2 tabel yang bergantung pada tabel lain, tabel Sertifikat Instruktur bergantung pada tabel Instruktur, tabel Sertifikat Instruktur sendiri dibuat untuk mengakses Sertifikat Instruktur yang sudah di-*upload* ke sistem. Sama halnya dengan tabel Surat Izin Penyelenggaraan, digunakan untuk mengakses Surat Izin Penyelenggaraan Kursus, yang digantungkan dengan tabel Pemilik/Admin, untuk sebuah akun dapat menjadi Pemilik/Admin, akun tersebut harus memiliki Surat Izin Penyelenggaraan yang sah. Dan yang terakhir 1 tabel agregasi untuk mengkategorikan kelas kursus, sebagai contoh, apakah kelas yang

ditawarkan untuk mobil dengan transmisi manual atau otomatis? atau apakah kelas yang ditawarkan sifatnya kursus kilat, kursus privat, atau kursus *reguler*? dan sebagainya.

3.5. *Development*

Pada tahap selanjutnya, tim pengembang akan mengkonversi hasil desain menjadi sistem yang dapat dioperasikan di berbagai jenis perangkat, terutama smartphone semua pengguna. Pengembangan ini akan dilakukan berbasis web, sehingga tim pengembang akan menggunakan teknologi HTML, CSS, Javascript, Framework Laravel, dan MySQL untuk sistem manajemen database. Proses konversi desain menjadi sistem web akan melalui beberapa langkah berikut:

- a) **Pemrograman *front-end*:** Tim pengembang akan menggunakan HTML, CSS, dan *Javascript* untuk membangun tampilan dan interaksi antarmuka pengguna.
- b) **Pemrograman *back-end*:** Tim pengembang akan menggunakan *Framework* Laravel untuk membangun logika dan fungsionalitas sistem.
- c) **Integrasi *database*:** Tim pengembang akan menggunakan MySQL untuk menyimpan dan mengelola data sistem.
- d) **Pengujian dan penyempurnaan:** Tim pengembang akan melakukan pengujian menyeluruh untuk memastikan sistem berjalan dengan baik dan sesuai dengan kebutuhan pengguna.

Detail lebih lanjut tentang proses pengembangan sistem web ini, termasuk teknologi yang digunakan, arsitektur sistem, dan hasil pengujian, akan dibahas pada Bab IV.

3.6. *Test*

Salah satu metode pengujian yang kami lakukan adalah *Black Box Testing*. *Black Box Testing* adalah pengujian yang dilakukan untuk mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak setelah

proses pengembangan selesai (Zidan, M. et al., 2022). Fokus dari pengujian ini adalah menguji kinerja fungsi dari sistem, apakah sudah bekerja sesuai dengan yang diharapkan. Berikut adalah penggalan tabel *black box testing* dari aplikasi kami nantinya.

Tabel 3. 54 Contoh Tabel *Black Box Testing*

Kode	Skenario Tes	Hasil yang Diharapkan	Sesuai
B01	<i>Form</i> Login/Daftar Akun tidak diisi apa-apa	Sistem tidak dapat mengambil inputan yang diberikan dan menampilkan pesan “Pengisian <i>form</i> tidak sesuai”	✓
B02	Pada <i>form login</i> , mengisi kolom <i>username</i> = “janedoe@gmail.com” dan <i>password</i> = “userlogin123”	Sistem mengambil inputan yang diberikan pengguna dan mengarahkan pengguna ke halaman beranda	✓
B03	Pada <i>form login</i> , mengisi kolom <i>username</i> = “janedoe@gmail.com” dan <i>password</i> = “user1234”	Sistem mengambil inputan yang diberikan pengguna dan mengembalikan pesan <i>error</i> “Password yang dimasukkan tidak sesuai”	✓
B04	Pada <i>form login</i> , mengisi kolom <i>username</i> = “abcabc” dan <i>password</i> = “userlogin123”	Sistem mengambil inputan yang diberikan pengguna dan mengembalikan pesan <i>error</i> “Data Pengguna tidak ditemukan”	✓

Tabel 3. 55 Contoh Tabel *Black Box Testing* (Lanjutan-1)

Kode	Skenario Tes	Hasil yang Diharapkan	Sesuai
Jumlah Pengujian			4
Sesuai			2
Tidak Sesuai			2
Tingkat Keberhasilan			50%

Selanjutnya, pada tahun 1996, John Brooke, mengembangkan sebuah skala untuk mengukur atau menilai kegunaan yang dirasakan peserta pengujian, yang dikenal dengan *System Usability Scale* (SUS). (Vlachogianni, P. et al., 2021) beranggapan mengapa SUS sangat populer bahkan sampai saat ini adalah karena SUS adalah alat pengukur psikologi yang handal dan tidak membutuhkan biaya dengan tingkat validasi tinggi dan dapat diandalkan. SUS terdiri dari 10 pernyataan yang secara bergantian menguji kesan positif dan negatif dari perasaan responden atau peserta pengujian terhadap sistem yang diujikan. Dibawah ini adalah 10 pertanyaan yang dirumuskan oleh John Brooke.

Tabel 3. 56 Contoh Tabel Pengujian *System Usability Scale*

No.	Pertanyaan	Sangat Tidak Setuju			Sangat Setuju	
		1	2	3	4	5
1	Saya berpikir akan menggunakan sistem ini lagi					

Tabel 3. 57 Contoh Tabel Pengujian *System Usability Scale* (Lanjutan-1)

No.	Pertanyaan	Sangat Tidak Setuju			Sangat Setuju	
		1	2	3	4	5
2	Saya merasa sistem ini rumit untuk digunakan					
3	Saya merasa sistem ini mudah untuk digunakan					
4	Saya membutuhkan bantuan dari orang lain atau teknisi dalam menggunakan sistem ini					
5	Saya merasa fitur-fitur sistem ini berjalan dengan semestinya					
6	Saya merasa ada banyak hal yang tidak konsisten (tidak serasi) pada sistem ini					
7	Saya merasa orang lain akan memahami cara menggunakan sistem ini dengan cepat					
8	Saya merasa sistem ini membingungkan					
9	Saya merasa tidak ada hambatan dalam menggunakan sistem ini					

Tabel 3. 58 Contoh Tabel Pengujian *System Usability Scale* (Lanjutan-2)

No.	Pertanyaan	Sangat Tidak Setuju			Sangat Setuju	
		1	2	3	4	5
10	Saya perlu membiasakan diri terlebih dahulu sebelum menggunakan sistem ini					

3.7. Maintenance

Tahapan ini dapat dilakukan apabila di masa mendatang, dengan adanya kebutuhan-kebutuhan tambahan, pihak kursus merasa ada beberapa fungsi atau fitur yang perlu ditambahkan. Tahap pemeliharaan sangat penting untuk memastikan bahwa sistem tetap relevan dan memenuhi kebutuhan pengguna. Pemeliharaan yang berkelanjutan dapat membantu memperpanjang umur sistem. Serta, penting untuk memiliki rencana pemeliharaan yang jelas untuk sistem. Biaya pemeliharaan harus diperhitungkan dalam anggaran keseluruhan untuk proyek. Tim pengembang harus selalu siap untuk melakukan pemeliharaan pada sistem.

Selain penambahan fitur-fitur sesuai kebutuhan, proses-proses pemeliharaan aplikasi dapat dikelompokkan menjadi 3 kelompok:

- **Pemeliharaan Fungsional** menjamin bahwa fungsionalitas-fungsionalitas utama dari aplikasi bekerja sesuai yang diinginkan. Beberapa aktivitas-aktivitas yang dilakukan adalah : Memperbaiki *bug*, Pembaruan keamanan, Optimalisasi kinerja, Pembaruan kompatibilitas, dan Pembaruan fitur serta aktivitas-aktivitas lain.
- **Pemeliharaan Non Fungsional** mencakup aktivitas-aktivitas pemeliharaan kualitas dan pengalaman pengguna dari aplikasi secara keseluruhan diluar fungsionalitas utama. Aktivitas-aktivitas yang dilakukan diantaranya :

Pembaruan konten, Pengujian kegunaan, Pengawasan kinerja, Pemeliharaan aksesibilitas, Pembaruan dokumentasi aplikasi, dan-lain-lain.

- **Pemeliharaan Infrastruktur** berfokus pada pemeliharaan-pemeliharaan infrastruktur yang mendukung aplikasi. Aktivitas-aktivitas yang tergolong pemeliharaan infrastruktur yaitu : Pemeliharaan *server*, Pemeliharaan jaringan, Pemeliharaan peluncuran, Pengelolaan skala aplikasi, dan aktivitas-aktivitas lain.

DAFTAR PUSTAKA

- Adhiva Kurnia, F. (2023). Driving Course And Driving License Service Information System Web-Based (Study Case Kurnia Jaya). In *Journal Of Computer Science And Big Data Journal Homepage:login* (Vol. 1, Issue 1). <http://jcosbida.com/index.php/index/http://jcosbida.com/index.php/index/loginirpi.or.id/index.php/malcom/article/view/89>
- Adrianto, S. (2021). Aplikasi Kenaikan Gaji Berkala Menggunakan Bahasa Pemrograman PHP pada Dinas Pendidikan dan Kebudayaan Kota Dumai. *Informatika*, 13(1), 32-39.
- Ahmad. (2022). *Cara Menulis Daftar Pustaka Dari Buku, Jurnal, Skripsi, Artikel, Website*. Diakses pada 26 Maret 2024, dari <https://www.gramedia.com/best-seller/cara-menulis-daftar-pustaka/>
- Arnowitz, J., Arent, M., & Berger, N. (2007). *Effective Prototyping for Software Makers*. Morgan Kaufmann Publishers.
- Bahar, Wibawa, B., & Situmorang, R. (n.d.). *Rekayasa Perangkat Lunak - Pendekatan Terstruktur & Berorientasi Objek*.
- Firdaus, A. (2022). *Pemodelan Proses Bisnis Konveksi di Tasikmalaya dengan Business Process Model and Notation (BPMN)*. *Jurnal Ekonomi dan Bisnis Digital*, 1(3), 133-142.
- GeeksforGeeks. *Activity Diagrams / Unified Modeling Language (UML)*. Diakses pada 29 Maret 2024, dari <https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/#2-activity-diagram-notations>
- GeeksforGeeks. *Class Diagram / Unified Modeling Language (UML)*. Diakses pada 29 Maret 2024, dari <https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/>
- Google. (2023). Bard (versi Desember 2023) [Model bahasa besar]

- Google. (2024). Gemini (versi Maret 2024) [Model bahasa besar]
- Hoang, H. (2022, 26 Desember). *Prototype model in software engineering: an 2023 overview*. Diakses pada 28 Maret 2024, dari <https://biplus.com.vn/prototype-model/>
- Jacobson, L., & Booch, J. R. G. (2021). *The unified modeling language reference manual*.
<http://debracollege.dspaces.org/bitstream/123456789/404/1/UML%20Reference%20Manual%20by%20James%20Rambaugh.pdf>
- KBBI. *Arti kata prototipe*. Diakses pada 26 Desember 2023, dari <https://kbbi.web.id/prototipe>
- Made, N., Elianti, D., Putra Githa, D., Ngurah, A. A., & Susila, H. (2022). Android-Based Driving Course Information System.
- Mahdy, N. R., Kasyrafurhman, G., Ramadhan, B., & Capah, D. A. H. (2021). Aplikasi Sistem Informasi Kursus Mengemudi Berbasis Web (Studi Kasus: Kursus Setir Mobil Santa). *JURNAL ILMIAH BETRIK: Besemah Teknologi Informasi dan Komputer*, 12(2), 178-185.
<https://scholar.archive.org/work/yjijjloibh45hhxsit6vblfey/access/wayback/https://ejournal.lppmsttpagaralam.ac.id/index.php/betrik/article/download/330/270>
- Martin, M. (diupdate 2024, 24 Februari). *Prototype Model in Software Engineering*. Diakses pada 28 Desember 2023, dari <https://www.guru99.com/software-engineering-prototyping-model.html>
- Miranda, R. A. (2023, November 29). *Apa itu PHP? – Pengertian, Fungsi, Sintaks, dan Kelebihannya*. Diakses pada 14 Maret 2024, dari <https://sekawanstudio.com/blog/php-adalah/#:~:text=Sejarah%20PHP,atau%20disingkat%20menjadi%20PHP%20tools>.

- Noviantoro, A. ., Silviana, A. B., Fitriani, R. R., & Permatasari, H. P. (2022). RANCANGAN DAN IMPLEMENTASI APLIKASI SEWA LAPANGAN BADMINTON WILAYAH DEPOK BERBASIS WEB. *Jurnal Teknik Dan Science*, 1(2), 88–103. <https://doi.org/10.56127/jts.v1i2.108>
- Nugroho, A. (2023). *Rancang Bangun Aplikasi Penyewaan Ruangan Untuk Penyelenggaraan Event Berbasis Web Dengan Menggunakan Metode Prototype*. (Proposal Skripsi, Institut Teknologi Adhi Tama Surabaya, 2023)
- Oktriwina, A.S. (2021). *Apa Itu Class Diagram dan Fungsinya dalam Pemrograman*. Glints. Diakses pada 29 Maret 2024 dari <https://glints.com/id/lowongan/class-diagram-adalah/>.
- OMG | Object Management Group. (2011). *Business Process Model and Notation* (BPMN). OMG.org. Diakses pada 14 Maret 2024 dari <http://www.omg.org/spec/BPMN/2.0>.
- Sari, I. P., Azzahrah, A., Qathrunada, I. F., Lubis, N., & Anggraini, T. (2022). Perancangan sistem absensi pegawai kantor secara online pada website berbasis HTML dan CSS. *Blend sains jurnal teknik*, 1(1), 8-15. <https://jurnal.ilmubersama.com/index.php/blendsains/article/view/66/23>
- Septiany, D.A. (2017). *Pengembangan dan Analisis Sistem Informasi Kemajuan Kelas Berbasis Website di SMK Muhammadiyah 1 Bantul*. (Skripsi, Universitas Negeri Yogyakarta, 2017)
- Setiawansyah, S., Lestari, D. T., & Megawaty, D. A. (2022). Sistem Informasi Pkk Berbasis Website Menggunakan Framework Codeigniter (Studi Kasus: Kampung Purworejo). *Jurnal Informatika dan Rekayasa Perangkat Lunak*, 3(2), 244-253. <https://jim.teknokrat.ac.id/index.php/informatika/article/view/2031/619>
- Simanullang, N. H., Siregar, A. W. B., & Masrizal, M. (2021). Sistem Informasi Pemesanan Menu Makanan Pada Rm Sedep Roso Rantauprapat Berbasis Web. *Journal of Student Development Informatics Management (JoSDIM)*,

1(1), 12-18.

<https://jurnal.ulb.ac.id/index.php/JoSDIM/article/view/2175/1946>

Somi, M. (2023). User Interface Development of a Modern Web Application (Doctoral dissertation, Politecnico di Torino). <https://webthesis.biblio.polito.it/secure/30076/1/tesi.pdf>

Subecz, Z. (2021). Web-development with Laravel framework. *Gradus*, 8(1), 211-218. https://real.mtak.hu/125616/1/2021_1_CSC_006_Subecz.pdf

Telkom University. (2023, 1 November). *Penulisan Daftar Pustaka dari Buku, Artikel Jurnal, Makalah, Media Online, hingga Video YouTube*. Diakses pada 26 Maret 2024, dari <https://telkomuniversity.ac.id/penulisan-daftar-pustaka-dari-buku-artikel-jurnal-makalah-media-online-hingga-video-youtube/>

tutorialspoint. (2023, November). *Software Engineering Overview*. Diakses pada 20 Desember 2023, dari https://www.tutorialspoint.com/software_engineering/software_engineering_overview.htm#:~:text=Definitions,as%20in%20the%20above%20statemen
[t](https://www.tutorialspoint.com/software_engineering/software_engineering_overview.htm#:~:text=Definitions,as%20in%20the%20above%20statemen)

Setiyawati, N., Bangkalang, D.H. (2022). The Comparison of Evaluation on User Experience and Usability of Mobile Banking Applications Using User Experience Questionnaire and System Usability Scale. *Proceedings* (2022): 82-87. <https://doi.org/10.3390/proceedings2022082087>

Vlachogianni, P., & Tselios, N. (2021). Perceived usability evaluation of educational technology using the System Usability Scale (SUS): A systematic review. *Journal of Research on Technology in Education*, 1–18. doi:10.1080/15391523.2020.1867938

Zidan, M., Nur'aini, S., & Wibowo, N. C. H., Ulinuha, M. A., (2022). Black Box Testing pada Aplikasi Single Sign On (SSO) di Diskominfostandi Menggunakan Teknik Equivalence Partitions. *Walisongo Journal of*

Information Technology, Vol. 4 No. 2 (2022): 127-137.

<https://doi.org/10.21580/wjit.2022.4.2.12135>