**RANCANG BANGUN SISTEM INFORMASI**

**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**

# LEMBAR PENGESAHAN

# DAFTAR ISI

[**HALAMAN JUDUL** i](#_Toc162351454)

[**LEMBAR PENGESAHAN ii**](#_Toc162351457)

[**DAFTAR ISI iii**](#_Toc162351458)

[**DAFTAR GAMBAR vii**](#_Toc162351459)

[**DAFTAR TABEL viii**](#_Toc162351460)

[**BAB I PENDAHULUAN 1**](#_Toc162351462)

[1.1. Latar Belakang 1](#_Toc162351463)

[1.2. Rumusan Masalah 5](#_Toc162351464)

[1.3. Tujuan Penelitian 5](#_Toc162351465)

[1.4. Batasan Masalah 6](#_Toc162351466)

[1.5. Sistematika Penulisan 6](#_Toc162351467)

[**BAB II LANDASAN TEORI 7**](#_Toc162351469)

[2.1. Kursus Mengemudi 7](#_Toc162351471)

[2.2. Rekayasa Perangkat Lunak 7](#_Toc162351472)

[2.3. Metode Rekayasa Perangkat Lunak *Prototyping* 9](#_Toc162351473)

[2.3.1. Apakah rancangannya berfungsi dengan baik? 11](#_Toc162351479)

[2.3.2. Apakah rancangannya bisa dikembangkan dengan ekonomis? 11](#_Toc162351480)

[2.3.3. Bagaimana tanggapan pengguna terhadap rancangannya? 12](#_Toc162351481)

[2.3.4. Pendekatan apa yang dapat diambil untuk beralih dari konsep ke produk (akhir)? 12](#_Toc162351482)

[2.3.5. Bagaimana pembuatan prototype dapat mendukung spesifikasi perancangan produk? 13](#_Toc162351483)

[2.3.6. Bagaimana pembuatan prototype berkontribusi untuk merencanakan durasi pengerjaan dan penghitungan anggaran supaya lebih baik? 13](#_Toc162351484)

[2.4. Sistem Informasi Berbasis Web 19](#_Toc162351485)

[2.4.1. *Waterfall* 22](#_Toc162351487)

[2.4.2. *Prototyping* 25](#_Toc162351488)

[2.4.3. *Agile Software Development* (ASD) 25](#_Toc162351489)

[2.5. *Business Process Model and Notation* (BPMN) 27](#_Toc162351490)

[2.6. *Use Case Diagram* 29](#_Toc162351491)

[2.7. *Unified Modeling Language* (UML) 30](#_Toc162351492)

[2.8. *Hyper Text Markup Language* (HTML) 31](#_Toc162351493)

[2.9. *Cascading Style Sheets* (CSS) 32](#_Toc162351494)

[2.10. *Javascript* 33](#_Toc162351495)

[2.11. *Framework PHP* Laravel 33](#_Toc162351496)

[2.11.1. *Routing* 34](#_Toc162351504)

[2.11.2. Arsitektur MVC 35](#_Toc162351505)

[2.11.3. *Views* dan *Templates* 36](#_Toc162351506)

[2.11.4. *Controller* 36](#_Toc162351507)

[2.11.5. Berinteraksi dengan *database* 37](#_Toc162351508)

[2.11.6. *Model* dan *Eloquent* 37](#_Toc162351509)

[2.11.7. *Authentication* dan *Authorization* 39](#_Toc162351510)

[2.11.8. *Middleware* 39](#_Toc162351511)

[2.11.9. *Validation* 39](#_Toc162351512)

[2.11.10. Pengamanan aplikasi 40](#_Toc162351513)

[**BAB III METODE PENELITIAN 41**](#_Toc162351515)

[3.1. Proses Bisnis 41](#_Toc162351517)

[3.2. Analisa Kebutuhan 47](#_Toc162351518)

[3.3. Use Case 49](#_Toc162351519)

[3.3.1. *Login* 51](#_Toc162351524)

[3.3.2. Menampilkan *Dashboard* Jadwal Kursus 53](#_Toc162351525)

[3.3.3. Pengelolaan kelas Kursus 55](#_Toc162351526)

[3.3.4. Menampilkan Detail *Progress* Kursus 58](#_Toc162351527)

[3.3.5. Komunikasi 60](#_Toc162351528)

[3.3.6. Siswa mendaftar kelas kursus 62](#_Toc162351529)

[3.3.7. Pembayaran 64](#_Toc162351530)

[3.3.8. Perubahan Jadwal 66](#_Toc162351531)

[3.3.9. Mendaftarkan Kursus Baru 69](#_Toc162351532)

[3.4. Tahapan *Prototyping* 73](#_Toc162351533)

[3.4.1. Verifikasi Kebutuhan 73](#_Toc162351535)

[3.4.2. Alur Tugas 74](#_Toc162351536)

[3.4.3. Penentuan Konten dan Tingkat Ketelitian *Prototype* 77](#_Toc162351537)

[3.4.4. Penentuan Karakteristik *Prototype* 78](#_Toc162351538)

[3.4.5. Memilih sebuah metode *prototyping* 81](#_Toc162351539)

[3.4.6. Memilih tools prototyping 85](#_Toc162351540)

[3.4.7. Menetapkan kriteria desain 86](#_Toc162351541)

[3.4.8. Membuat desain 88](#_Toc162351542)

[3.4.9. Meninjau desain 89](#_Toc162351543)

[3.4.10. Memvalidasi desain 90](#_Toc162351544)

[3.4.11. Men-*deploy* desain 91](#_Toc162351545)

[**DAFTAR PUSTAKA 94**](#_Toc162351546)

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB I

# PENDAHULUAN

## 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.



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 jembatani 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 sistematik 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 projek 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 SISTEM INFORMASI UNTUK PENYEDIA JASA KURSUS MENGEMUDI**”

## Rumusan Masalah

Dari Latar Belakang diatas, terdapat 1 poin penting yang dapat mewakili permasalahan-permasalahan lain : Dengan mengintegrasikan Sistem Informasi dan menerapkan Rekayasa Perangkat Lunak, apakah berdampak terhadap peningkatan kualitas layanan dari penyedia jasa kursus mengemudi ?

## Tujuan Penelitian

Tujuan dari penelitian ini yaitu untuk menjawab rumusan masalah yang sebelumnya disebutkan serta meluncurkan aplikasi untuk mewadahi para penyedia jasa kursus mengemudi untuk bisa menawarkan jasanya seperti mereka berada di sebuah *marketplace* yang berbasis web dan bisa diakses melalui telepon genggam mereka masing-masing.

## Batasan Masalah

Dari Latar Belakang dan Rumusan Masalah diatas dapat kami tetapkan bahwa masalah yang bisa kami atasi untuk penelitian ini adalah mengembangkan sistem informasi untuk penyedia jasa kursus mengemudi di Surabaya dan sekitarnya berbasis website.

## 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 tentang apa yang melandasi kami untuk 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 sebut dan sering kami gunakan dalam menyelesaikan penelitian kami.

**BAB III METODOLOGI PENELITIAN**

Untuk mengetahui langkah-langkah dalam melakukan persiapan penelitian dan proses awal dari penelitian seperti perencanaan, proses bisnis yang saat ini berjalan, kebutuhan pengguna, dan rancangan *prototype* yang ditawarkan, dapat dilihat pada bab ini..

# BAB II

# LANDASAN TEORI



## 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 bahwa 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.

## 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 sistematik, 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.

## 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) (Gemini, 2023).

*Prototyping* menurut Kamus Webster 1913 adalah (produk) asli atau model (sederhana) yang kemudian disalin; sehingga 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. *Prototype* memiliki dua tingkat ketelitian, *low-fidelity* atau ketelitian tingkat rendah yang sebatas konseptual, kasar, dan tidak mendetail. Kemudian, *high-fidelity* atau ketelitian tingkat tinggi yang sudah dapat kita lihat bentuk akhir dari tampilan perangkat lunak yang dirancang. Prototyping mampu menjawab pertanyaan-pertanyaan berikut ini :



### Apakah rancangannya berfungsi dengan baik?

Sebuah *prototype* dapat memvisualisasikan apakah perangkat lunak yang dikembangkan saat ini secara fungsionalitas sudah sesuai target. Sebuah *prototype* perangkat lunak dibangun berdasarkan pertanyaan-pertanyaan yang harus terjawab terlebih dahulu sebelum mengakhiri iterasi perancangan: Apa interaksi yang disediakan sudah lengkap? Apakah ada jalan buntu? Sudahkah kita mencakup tugas-tugas yang akan dieksekusi pengguna? Apakah fungsionalitas yang direpresentasikan di *prototype* berhasil? Dengan menggabungkan umpan balik yang didapat dari calon pengguna, desainer, ahli-ahli domain, *prototyping* secara berkala dapat memberikan sebuah gagasan baru untuk tahapan pengembangan berikutnya. Pada akhirnya, *prototyping* membantu para teknisi RPL untuk mencapai pengalaman pengguna (*user experience*) dengan baik, tidak hanya itu, *prototyping* juga menurunkan risiko rekayasa perangkat lunak.

### Apakah rancangannya bisa dikembangkan dengan ekonomis?

Ketika kita berdiskusi tentang *prototyping* untuk perangkat keras, biaya untuk setiap komponennya sudah jelas; (perangkat keras) itu sebuah elemen fisik yang memiliki harga tetap dan faktor ketersediaan. Sedangkan pada lingkup perangkat lunak, definisi “ekonomis” tergantung terhadap nilai-nilai organisasi. Ekonomis bisa jadi berarti mengembangkan perangkat lunak di negara yang biaya pengerjaannya murah, atau bisa jadi mengurangi jumlah *update* atau *patch,* untuk mengurangi biaya. “Biaya” bisa diartikan sebagai sebuah usaha yang dikeluarkan untuk menyelesaikan pengembangan, pemeliharaan, dan penunjangan sebuah perangkat lunak, atau bisa juga sebagai tarif untuk sebuah penambahan fitur khusus tanpa menambah biaya pemeliharaan. Sebuah *prototype* memungkinkan pemangku kepentingan untuk mengevaluasi seberapa tinggi biaya sebuah perangkat lunak nantinya. Intinya nilai ekonomi pada *prototyping* dapat dimanfaatkan untuk menurunkan tingkat risiko dan kebocoran anggaran dua hal tersebut dapat dikurangi dengan memberikan teknisi RPL, bagian pemasaran, dan manajer produk untuk mengestimasi biaya yang dibutuhkan nantinya.

### Bagaimana tanggapan pengguna terhadap rancangannya?

Manfaat *prototyping* selanjutnya adalah untuk mengevaluasi tanggapan pengguna dan/atau pemangku kepentingan lain terhadap konsep dari perangkat lunak. Membedakan tanggapan antara perangkat keras dan perangkat lunak dapat dilakukan sebagai berikut, tanggapan yang biasanya diberikan saat pengujian *prototype* untukperangkat keras adalah apakah perangkat yang digunakan nyaman digenggam atau tidak. Sedangkan untuk pengujian *prototype* perangkat lunak, tanggapan yang diberikan biasanya mencakup “tampilan dan nuansa”, dua hal tersebut adalah gabungan dari teknik-teknik interaksi dan desain visual. Dalam kata lain, sebuah prototype dapat membantu kita menentukan apakah sebuah perangkat lunak yang dirancang sesuai dengan pengguna. Sebuah *prototype* dapat memperlihatkan sebuah ide seakan seperti kenyataan, apabila dieksekusi dengan benar, *prototyping* dapat memberikan dampak yang baik kepada pemangku kepentingan

### Pendekatan apa yang dapat diambil untuk beralih dari konsep ke produk (akhir)?

Ide dalam proses *prototype* menghasilkan banyak hal baru untuk dijelajahi dan pemahaman-pemahaman baru tentang bagaimana masyarakat umum berinteraksi dengan perangkat lunak. Salah satu contoh dari *prototyping* untuk penelitian secara umum adalah model *prototype* kosong. Model *prototype* kosong dibuat berdasarkan eksplorasi interaktif dari bermacam ukuran dan bentuk perangkat genggam dan penempatan tombol di luarnya. Desain yang menerapkan keterlibatan sering dimanfaatkan untuk mendapatkan bermacam perspektif, dalam mengembangkan ide. Seorang desainer dan calon pengguna menyusun balok busa (*styrofoam*), tombol yang diberi selotip ganda, dan bahan kerajinan lainnya untuk mengeksplorasi bentuk perangkat keras dan penempatan tombol. Kemudian, peserta pengujian mendiskusikan apa yang mereka lakukan dan mereka berikan alasannya, sehingga, akan bermunculan berbagai ide. Tujuan dari praktek ini adalah untuk berfokus pada rancangan dan manipulasi fisik, dan pengetahuan yang didapat dari praktek ini digunakan untuk menjawab pertanyaan-pertanyaan penelitian.

### Bagaimana pembuatan prototype dapat mendukung spesifikasi perancangan produk?

Dalam lingkup formal, desainer menghasilkan spesifikasi desain menyeluruh yang telah dikualifikasikan pada pengujian tingkat kemudahan penggunaan, *prototyping*, eksplorasi, dan berbagai proses lain. Dalam lingkup lainnya, *prototype* yang dimanfaatkan untuk membantu pengembangan detail rancangan juga bisa dimanfaatkan sebagai spesifikasi rancangan. Sebuah *prototype* adalah spesifikasi desain yang jauh lebih konkrit dan dapat diaudit, dibandingkan menggunakan *use cases* atau definisi fungsional saja. Ketika *prototype* digunakan sebagai spesifikasi, *prototype* dengan ketelitian tingkat tinggi yang komprehensif dapat menjadi panduan pada saat proses implementasi. Walaupun demikian, *prototype* tidak bisa menggantikan dokumentasi. Pengetahuan yang didapat dari *prototype* sangat dianjurkan untuk didokumentasikan untuk melengkapi persyaratan atau dokumen kebutuhan (SKPL) yang kemudian pengetahuan dan hasil pemikiran rancangan tersebut dapat di bagikan ke projek yang lain.

### Bagaimana pembuatan prototype berkontribusi untuk merencanakan durasi pengerjaan dan penghitungan anggaran supaya lebih baik?

Menentukan lingkup dan sumber daya yang dibutuhkan adalah salah satu fase yang tersulit pada proses perencanaan proyek perangkat lunak. Tanpa representasi yang nyata dari sebuah produk atau fitur produk, perencanaan dilakukan secara abstrak, berdasarkan pengalaman sebelumnya, keahlian tim pengembang, dan familiaritas produk yang dikembangkan dengan kondisi pasar, produk yang sudah ada, dan teknologi yang dituju. Secara umum, perencanaan produk dan fitur perangkat lunak didasarkan pada informasi kompleks yang tidak sesuai seperti tren pasar, permintaan industri, permintaan fitur dari pengguna, inovasi dan keberhasilan kompetitor, rencana proyek sebelumnya, pengalaman anggota tim sebelumnya, dan biaya yang ada.

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. Kemudian, timbul pertanyaan, bagaimana melakukan *prototyping* dengan efektif? Menurut (Arnowitz et al., 2007) dalam bukunya yang berjudul *“Effective Prototyping for Software Makers”* mereka menjelaskan proses *prototyping* terbagi menjadi 4 fase, dan dalam setiap fase tersebut terdapat langkah-langkah yang dapat membantu kita dalam melakukan proses *prototyping*.

Fase pertama: Perencanaan, pada fase ini, kita akan menentukan kebutuhan dan rencana proses *prototyping*. Dengan mengenali, mengadaptasi, dan mematuhi rencana pembuatan *prototype*, kita dapat memahami dan mengidentifikasi dan kebutuhan dan asumsi dari *prototype*, membuat alur tugas dan skenario untuk menyesuaikan konteks, kemudian memutuskan bagaimana perpaduan antara konten dan tingkat ketelitian terkait dengan konteks tersebut.

Langkah pertama: Verifikasi kebutuhannya, proses awal dari fase ini dimulai dengan menentukan kebutuhan *prototyping*. Kebutuhan ini sama sekali berbeda dengan kebutuhan perangkat lunak (SKPL), lebih ke sebagian kecil dari kebutuhan tersebut berdasarkan peserta pengujian *prototype* nantinya, dan pada fase apa dalam proses pembuatan perangkat lunak. Dalam menentukan kebutuhan *prototyping,* kita harus berfokus kepada *prototype* yang mana yang berdampak terhadap alur tugas dan konten di dalam *prototype.*

Langkah kedua: Buat alur tugas/tampilannya, agar bisa melakukan *prototype* secara efektif, kita wajib memahami bagaimana seorang pengguna nantinya bernavigasi dari satu tampilan ke tampilan lainnya. Serta, penting juga untuk memahami apa yang terjadi ketika seorang pengguna mengklik salah satu *widget* (atau mengapa pengguna tersebut ingin mengklik *widget-*nya). Membuat sketsa alur tugas adalah aktivitas yang dapat diukur: bisa saja kita buat sketsa kecil atau bahkan besar dari keseluruhan sistem atau sketsa untuk mewakili satu orang dalam tim atau keseluruhan tim. Sebuah alur tugas kemudian dapat diubah menjadi rancangan dan *prototype* yang dikembangkan melalui proses iterasi. Terkadang, membuat sketsa alur tugas tidak semudah memahami alur tugasnya; melainkan kita juga harus paham konteks dari mengapa tugas tersebut bisa terjadi. Dan juga, bisa kita lengkapi alur tugas tersebut dengan skenario.

Langkah ketiga: Menentukan konten dan tingkat ketelitian, kebanyakan *prototype* dikelompokkan menjadi dua bagian, antara ketelitian tingkat rendah (*low-fidelity*) atau ketelitian tingkat tinggi (*high-fidelity*) dengan berbagai macam metode atau alat-alat yang dijadikan pada salah satu kelompok. Cara yang lebih komprehensif untuk mengelompokkan *prototype* adalah dengan mengidentifikasi konten didalamnya dan menyesuaikan informasi tersebut terhadap skala kemungkinan tingkat ketelitian. Selain karakteristik ketelitian diatas, *prototype* bisa memiliki tingkat ketelitian yang berbeda-berbeda untuk setiap jenis konten di dalamnya: desain informasi, desain interaksi dan model navigasi, desain visual, konten editorial, branding, dan kinerja/perilaku sistem.

Setelah tiga langkah diatas sudah ditentukan, maka kita bisa mulai masuk ke fase kedua: Spesifikasi, pada fase ini, proses *prototyping* mencakup hasil-hasil keputusan yang dibuat pada tiga langkah sebelumnya, dimana keputusan tersebut menentukan bagaimana langkah kita selanjutnya pada fase perencanaan. Pada fase ini pula kita melibatkan pengkhususan elemen-elemen *prototype* yang mewakili strategi *prototyping* secara keseluruhan. Pada fase perencanaan, kita menentukan apa yang di *prototype-*kan, pada fase ini, kita tentukan bagaimana melakukan *prototype*-nya. Hal ini dapat dilakukan dengan memecah isu yang nampak terlalu luas menjadi bagian-bagian kecil yang dapat dikelola sehingga mudah dipahami, dianalisa, dan dispesifikasikan, termasuk menspesifikasikan *prototype* itu sendiri dalam hal pengelompokkan karakteristik. Karakteristik tersebut memberikan kita pemahaman yang lebih dalam daripada hanya menyebutkan ketelitian tingkat rendah atau tingkat tinggi, dengan mendefinisikannya secara langsung apa yang kami inginkan dari *prototype* yang dibangun. Dengan demikian, kita bisa menentukan menggunakan metode apa nantinya kita membangun *prototype,* dan menggunakan alat apa, sehingga setelah kita mampu menyelesaikan setiap tahapan pada fase ini, pada fase selanjutnya kita bisa memfokuskan diri untuk melakukan desain dan perancangan.

Langkah selanjutnya, langkah keempat: Menentukan karakteristik *prototyping* yang tepat. Kesalahan ketika menentukan karakteristik *prototype* yang tepat adalah salah satu penyebab utama gagalnya melakukan *prototyping* dengan efektif. Seperti contoh, memberikan terlalu banyak atau terlalu sedikit informasi kepada target pengguna kita menyebabkan ketidakefektifan waktu kita, baik waktu untuk digunakan untuk membangun *prototype* atau waktu yang bisa digunakan untuk menguji *prototype* yang tidak mampu memberikan kita umpan balik yang diinginkan. Penting bagi kita untuk membedakan kepentingan antara pengguna akhir dan pemangku kepentingan yang membantu anda mengembangkan perangkat lunak. Selanjutnya, kita akan memetakan karakteristik dan konten dari *prototyping* terhadap metode yang paling tepat untuk kebutuhan kita.

Langkah kelima: Memilih metode *prototyping,* pada tahapan ini, kita akan berdiskusi dengan pihak-pihak yang terlibat untuk menentukan metode *prototyping* mana yang paling tepat dengan situasi saat ini. Akan kami tampilkan tabel untuk memilih metode *prototyping* mana yang paling tepat menurut kami*.*

Langkah keenam: Memilih alat *prototyping,* memilih alat yang sesuai dengan metode yang dipilih pada langkah sebelumnya. Saat ini sudah banyak sekali alat-alat untuk melakukan *prototyping* mulai dari yang tradisional seperti pensil dan kertas, dan perangkat lunak seperti Figma, Adobe XD, Invision Studio, Webflow, dan masih banyak yang lainnya.

Fase ketiga: Desain, setelah kita menentukan strategi *prototyping,* fase ketiga berfokus kepada proses eksekusi *prototype* melalui desain yang baik dan benar. Fase-fase sebelumnya membahas tentang perencanaan dan spesifikasi *prototype*, pada fase ini, kita menentukan bagaimana kita memanfaatkan informasi dan alat-alat yang ada sekaligus mengorganisir *prototype*. Fase ini membahas tentang bagaimana mengidentifikasi konten yang paling efektif, kaitannya dengan menciptakan desain konseptual dan skema navigasi untuk *prototype* kami. Pada fase ini kita akan mendiskusikan pemecahan tugas *prototyping*, menjadi bagian-bagian yang dapat dikelola untuk setiap iterasi *prototyping* dengan merumuskan kriteria desain. Kemudian kita akan satukan kebutuhan yang sudah dirumuskan juga sebelumnya dengan kriteria desain yang ada menjadi desain *prototype.*

Langkah ketujuh: Merumuskan kriteria desain, faktor utama dalam *prototyping* yang efektif adalah kemampuan untuk mempertahankan desain *prototype* kita. Desainer yang ahli dipastikan selalu mampu untuk menggabungkan panduan dalam mendesain dengan kebutuhan-kebutuhan teknis, kebutuhan-kebutuhan pengguna, dan kebutuhan-kebutuhan bisnis untuk menciptakan sebuah solusi terbaik. Pemikiran-pemikiran desain yang juga didasarkan pada panduan dalam mendesain dijamin akan mengeluarkan keputusan desain yang terbaik. Tetapi, setiap orang yang terlibat tidak semuanya seorang desainer, dan memang tidak harus semuanya memiliki dasar mendesain. Pada tahapan ini akan kami tampilkan praktik terbaik dalam mendesain dari berbagai sektor ilmu, mulai dari psikologi kognitif, desain grafis, dan desain informasi. Panduan-panduan desain ini memberikan jaminan kepada kita bahwa komposisi pada tampilan nantinya tidak dibuat secara sembarangan. Meskipun *prototype* yang dibuat nantinya bukan hasil desain yang terbaik, *prototype* tersebut bisa mudah dipahami oleh berbagai pihak, dan kita sebagai desainer harus mampu menjelaskan konsep pemikiran kita terhadap keputusan yang dibuat.

Langkah kedelapan: Bangun *prototype-*nya, pada langkah ini, kami akan menggabungkan panduan-panduan desain dan kebutuhan-kebutuhan perangkat lunak untuk menciptakan desain terbaik. Nantinya, kualitas *prototype* yang kita buat berdasarkan kualitas riset pengguna, definisi kebutuhan yang akurat, dan eksplorasi/iterasi dan analisa desain yang kita lakukan. Analisis yang kita lakukan hanya bisa dilakukan dengan baik dan menyeluruh apabila kita juga memahami panduan-panduan dan kebutuhan-kebutuhan yang dirumuskan sebelumnya, selain itu, pemahaman tentang kebutuhan pengguna juga dipertimbangkan.

Fase keempat: Hasil, pada fase pertama, kita berfokus tentang tahapan-tahapan untuk merencanakan proses *prototyping*. Dengan perencanaan yang matang, kita dapatkan pemahaman terbaik tentang apa yang akan kita bangun *prototype*-nya dan apa yang bisa kita tinggalkan untuk dibangun nantinya ketika kita ada proses pembuatan perangkat lunak. Pada fase kedua, kita menentukan spesifikasi yang diinginkan untuk menciptakan desain *prototype* dilanjutkan dengan fase ketiga yang menguraikan desain dari *prototype* dan bagaimana kita membangunnya dengan prinsip-prinsip desain. Fase keempat adalah saat dimana kita akan menyampaikan *prototype* yang kita bangun ke pihak-pihak internal, kemudian memvalidasikan *prototype* yang dibangun ke pihak-pihak eksternal, kemudian yang terakhir kita akan melanjutkan prosesnya dengan mengimplementasikannya ke dunia nyata.

Langkah kesembilan: Tinjau ulang *prototype*, langkah ini memaparkan tinjauan-tinjauan yang dilakukan dengan pemangku kepentingan dalam perusahaan dan cara-cara untuk menjamin bahwa proses *prototype* yang efektif tervalidasi dengan sah.

Langkah kesepuluh: Validasi desainnya, pada langkah ini akan kami tunjukkan cara-cara melakukan pengujian tingkat kemudahan penggunaan dan teknik-teknik validasi lainnya dengan pemangku kepentingan diluar perusahaan.

Langkah terakhir: Implementasi desainnya, setelah semua langkah dilalui, dan sudah kita lewati iterasi-iterasinya, langkah selanjutnya adalah menerjemahkan hasilnya menjadi produk atau dalam hal ini perangkat lunak yang nyata, yang selanjutnya bisa dirilis agar bisa digunakan oleh masyarakat luas. Pada langkah ini, implementasi melibatkan tindakan-tindakan yang memahami bahwa *prototype* yang dihasilkan sudah memenuhi tujuan dan sasaran dari rencana sebelumnya.

## 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 pengendalian, 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 projek pemrograman sistem informasi (PSI) berkisar antara 18 bulan sampai dengan 5 tahun, kemudian, pada laporan tersebut dikemukakan juga bahwa, 68% dari keseluruhan projek melebihi jadwal yang ditetapkan, kemudian 75% dari keseluruhan projek harus dirancang ulang setelah proses implementasi dan 35% perusahaan setidaknya memiliki satu projek yang sudah ditinggalkan. Dapat disimpulkan bahwa projek PSI ini merupakan sebuah projek 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 PSI. 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 projek 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 projek. 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*. Berikut akan kami uraikan dan bandingkan tiga metode pokok dalam pengembangan sistem informasi.



### *Waterfall*

Metode *Waterfall* membagi proses-proses pengembangan sistem informasi menjadi tahapan-tahapan formal dimana keluaran dari satu tahapan menjadi input tahapan selanjutnya. Keluaran ini selanjutnya turun terus menerus seperti air terjun dimana bagian akhir dari proses ini akan menghasilkan sistem perangkat lunak. Tahapan pertama dari metode ini adalah uji kelayakan, dimulai dengan menentukan lingkup dan tujuan dari sistem baru, mempertimbangkan jangkauan solusi alternatif dan menyelidiki dampak dari sistem baru nantinya pada perusahaan. Hasil keputusan dari uji kelayakan adalah untuk para manajer menyetujui solusi yang diinginkan.

Pada tahap kebutuhan, kebutuhan yang menjadi fokus akan dibentuk ke dalam sebuah form, sehingga, istilah yang dipahami oleh tim pengembangan adalah “apa” yang dibutuhkan bukan lagi “bagaimana” bisa kita atasi secara teknis. Spesifikasi kebutuhan ini harus disetujui oleh pelaku bisnis, ketika sebuah sistem sudah dibangun dan digunakan nantinya akan memenuhi kebutuhan-kebutuhan bisnis. Desain atau perancangan berfokus pada menerjemahkan kebutuhan yang dirumuskan menjadi sebuah spesifikasi sistem yang nantinya mampu diubah menjadi sebuah sistem yang bekerja oleh para pengembang sistem maksudnya pada saat proses *coding*. Pengujian secara menyeluruh dianggap penting dan kadang dilakukan pada tiga tingkatan. Tingkat pengujian pertama adalah pengujian yang dilakukan oleh *programmer* kepada modul-modul program. Jika para *programmer* merasa puas dengan masing-masing modul, kemudian seluruh modul tersebut dapat digabungkan menjadi satu dan dilakukan pengujian selanjutnya, yaitu pengujian sistem. Staf-staf pengembangan sistem akan merumuskan rencana pengujian dan data pengujian untuk diujikan kepada sistem sampai tahapan mendalam. Pada tahapan melibatkan pengujian kapasitas puncak untuk melihat seberapa banyak transaksi yang mampu dikelola oleh sistem secara bersamaan sebelum kinerja sistem menurun sampai dengan tingkat yang tidak dapat diterima atau sampai sistem benar-benar tidak mampu lagi.

Pengujian sistem tingkat tiga adalah uji penerimaan pengguna, ketika tim pengembang merasa puas dengan hasil uji sistem, langkah selanjutnya sistem akan diberikan kepada calon pengguna yang kemudian akan melakukan pengujian sistem itu sendiri. Setelah pengujian selesai, tahap selanjutnya adalah pengoperasian langsung, yang nantinya sistem akan tetap dipantau dan diawasi kesalahan-kesalahan sistemnya atau sering dikenal dengan *bug* dan akan dilakukan peningkatan kualitas sistem jika teridentifikasi *bug*.

Siklus metode *Waterfall* ini, berakar dari teknik pembuatan perangkat keras dan perangkat lunak dan meskipun sesuai untuk perancangan perangkat-perangkat digital, akan menjadi kurang sesuai jika dibandingkan dengan aktivitas manusia pada umumnya, dimana faktor-faktor manusia dan perusahaan lebih susah diidentifikasi. Meskipun demikian, bagi banyak projek-projek pengembangan sistem informasi yang ada siklus ini memiliki kekurangan-kekurangan diantaranya: mahal, memakan banyak waktu, dan kurang fleksibel. Selain itu, kekurangan yang paling signifikan pada siklus *Waterfall* ini adalah asumsi bahwa kebutuhan yang teridentifikasi diharap dan dianggap bisa memenuhi seluruh kebutuhan calon pengguna dan pemangku kepentingan bahkan sebelum proses perancangan dimulai. Karena terkadang, pengguna belum sepenuhnya tahu (atau belum bisa menyampaikannya dengan terus terang) kebutuhan mereka bahkan setelah sistem sudah terbangun. Tidak hanya itu, kebutuhan seharusnya tidak ditetapkan dan dibekukan, kebutuhan selalu berubah-ubah seiring dengan berjalannya waktu. Sehingga merumuskan sebuah spesifikasi kebutuhan yang komplit dan permanen bukan sebuah tujuan yang realistis.

Siklus waterfall memungkinkan komunikasi dua arah yang baik di antara pengguna dan pengembang (sebagai contoh: kesepakatan spesifikasi, kesepakatan uji penerimaan pengguna), tetapi tidak sepenuhnya mendukung proses pembelajaran dan koordinasi yang menguntungkan. Kemudian, kenapa metode ini masih digunakan sampai saat ini? Pendapat kami adalah *Waterfall* digunakan untuk memberikan *milestone* dan tahapan-tahapan yang jelas untuk membuat para manajer senang dan memberikan ilusi bahwa mereka mengendalikan proses PSI yang ada. Pendapat lain mengatakan bahwa metode ini mungkin dapat bekerja dengan baik pada situasi tertentu. Seperti ketika spesifikasi kebutuhan yang bisa dirumuskan dengan jelas, atau ketika sistem informasi yang ada akan diganti dengan tim pengembang yang berpengalaman pada domain aplikasi yang dibangun, seperti aplikasi penggajian atau inventaris, metode ini mungkin saja bisa sesuai.

### *Prototyping*

Metode ini dapat diimplementasikan dengan mengintegrasikan siklus *Waterfall*, kemudian dibuat sebuah *prototype* berdasarkan spesifikasi kebutuhan untuk menguji apakah sesuai dengan maksud calon pengguna atau pemangku kepentingan. Hal ini memungkinkan calon pengguna untuk menguji spesifikasi yang ditawarkan dan coba memahami permasalahan mereka sendiri. Sebab akan lebih baik jika kita menguji hasil terjemahan spesifikasi kebutuhan yang serupa dengan hasil akhir sistem nantinya. Pada pengembangan yang lebih jauh lagi, *prototype* berevolusi melalui rangkaian iterasi dan peningkatan sampai calon pengguna merasa puas dengan versi yang ditawarkan. Dalam metode RAD, *prototype* biasanya akan menjadi hasil akhir dari sistem atau setidaknya bagian dari hasil akhir sistem. Namun, bisa saja *prototype* yang dihasilkan tidak digunakan sama sekali karena sudah ada desain yang lebih baik dan yang mengadopsi sebagian fungsi *prototype*.

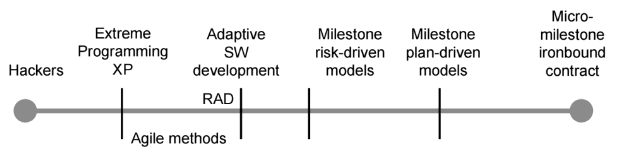
### *Agile Software Development* (ASD)

Metode ini meliputi pendekatan-pendekatan seperti *Extreme Programming* (XP), *Lean Development,* dan SCRUM. Sama halnya dengan RAD, dasar metode ini adalah perubahan tidak dapat dihindari. Melainkan sebuah masukan untuk mengatasi kebutuhan yang baru saja muncul ketika proses pengembangan dan dianggap lebih baik kita asumsikan bahwa spesifikasi kebutuhan pasti berubah dan akan lebih baik lagi apabila kita memfokuskan diri untuk mengurangi biaya pengerjaan ulang terkait. *Extreme Programming* (XP) memungkinkan tim pengembangan untuk:

* Menghasilkan versi pertama aplikasi dalam hitungan minggu, dengan memperoleh umpan balik secepat mungkin akan memberikan kesan bahwa pekerjaan yang dilakukan dapat terselesaikan dengan cepat.
* Mengatasi masalah dengan solusi paling simpel yang dapat diubah dengan mudah dan cepat.
* Meningkatkan kualitas desain sehingga versi selanjutnya akan lebih murah untuk dibuat dan diimplementasikan.
* Menjadikan proses pengujian aktivitas rutin sehingga kesalahan-kesalahan dapat terdeteksi secepat mungkin.

Metode *Agile* berfokus terhadap “buat senyata mungkin” melalui pembuatan kode pemrograman, karena pemrograman tidak kenal ampun (bisa saja bekerja dengan benar, bisa saja tidak) dibandingkan spesifikasi kebutuhan dan desain abstrak. Highsmith & Cockburn (2001) mengemukakan manifesto untuk metode ASD:

* Memprioritaskan interaksi satu sama lain dibandingkan proses dan alat-alat.
* Perangkat lunak yang bekerja dengan baik dibandingkan dokumentasi komprehensif.
* Kolaborasi dengan pelanggan dibandingkan negosiasi kontrak.
* Menanggapi perubahan dibandingkan mengikuti rencana.

Pendekatan metode ASD memprioritaskan fitur bukan rencana projek, dengan demikian, adaptasi dengan perubahan spesifikasi kebutuhan menjadi lebih mudah. Fokus terhadap pemrograman dan kecepatan ini membuat ASD dikritik sebagai ladang *hacker.* Pandangan lain berpendapat bahwa metode *Agile* ada dalam suatu spektrum yang mencakup proses *hacking* dan pengembangan berdasarkan *micro-milestone* yang terikat kontrak. Pilihan yang diambil untuk memposisikan sebuah proyek pada spektrum harus mempertimbangkan banyak faktor contohnya: budaya perusahaan, lingkungan eksternal, jenis aplikasi, dan tingkatan risiko.

## *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:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Notasi** | **Nama** | **Deskripsi** |
| 1 |  | ***Start Event*** | Sesuai dengan nama notasinya, notasi ini mengindikasikan dimana sebuah rangkaian proses dimulai. |
| 2 |  | ***Intermediate Event*** | Notasi ini digunakan ketika terjadi sebuah proses diantara rangkaian proses. |
| 3 |  | ***End Event*** | Sebagaimana nama dari notasi disamping, notasi ini mengindikasikan akhir dari rangkaian proses. |
| 4 |  | ***Task*** | 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 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Notasi** | | **Nama** | | | **Deskripsi** | |
|  |  | |  | | | lagi. | |
| 5 |  | | ***Choreograp-hy Task*** | | | Notasi ini merepresentasikan sebuah tugas yang melibatkan dua partisipan untuk menyelesaikannya. | |
| 6 |  | | ***Sub-process*** | | | Sebuah sub-proses adalah gabungan aktivitas yang dimasukkan pada sebuah proses. | |
| 7 |  | | ***Gateway*** | | | 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 |  | | ***Sequence Flow*** | | | Notasi ini mengindikasikan sebuah alur yang tidak dimulai dari *Intermediate Event* yang terhubung ke sebuah tugas / sub-proses. | |
| 9 |  | | ***Message Flow*** | | | Notasi ini digunakan untuk menunjukkan alur informasi antara dua partisipan yang bersiap untuk mengirim dan | |
| **No.** | | **Notasi** | | **Nama** | **Deskripsi** | |
|  | |  | |  | menerima informasi tersebut. | |
| 10 | |  | | ***Association*** | Notasi ini digunakan untuk menghubungkan informasi dan artifak terkait teks anotasi dan artifak lain yang terasosiasikan dengan elemen grafik lainnya. | |

## *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:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Notasi** | **Nama** | **Deskripsi** |
| 1 |  | ***Use Cases*** | Merepresentasikan fungsionalitas sebuah sistem, terkadang juga tujuan akhir dari aktor. Simbol ini selalu berada di dalam *Boundary Box*. |

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Notasi** | **Nama** | **Deskripsi** |
| 2 |  | ***Actors*** | Aktor yang berinteraksi dengan sistem biasanya memicu *use case*. Simbol ini selalu berada di luar *Boundary Box.* |
| 3 |  | ***Association*** | Arah panah digunakan untuk menandakan sebuah hubungan antara aktor dan *use case* atau antara dua *use case*.  Panah dengan anotasi <<extend>> menandakan bahwa sebuah *use case* mungkin mengadopsi perilaku dari *use case* lain.  Panah dengan anotasi <<include>> menandakan bahwa sebuah *use case* menggunakan fungsionalitas *use case* lain. |
| 4 |  | ***Boundary Box*** | Simbol ini mengindikasikan batas lingkup sistem. |

## *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.

## *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.

## *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.

## *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.

## *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:

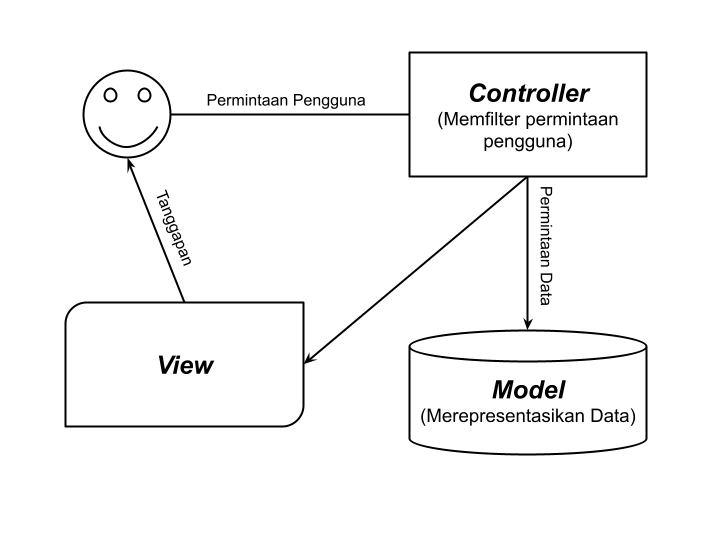


### *Routing*

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* projek yang berisi *file-file* rute tersebut.

### 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*.



### *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.

### *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*.

### 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.

### *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.



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*.

### *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*.

### *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.

### *Validation*

Kelas validatorLaravel 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.

### 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



## Proses Bisnis

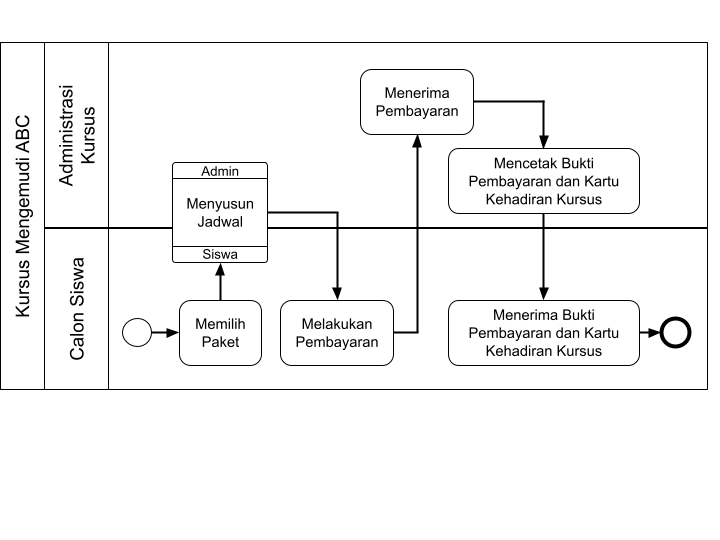
Seperti yang kita ketahui, berkendara dengan tertib di jalan raya merupakan tanggung jawab kita semua, namun, dalam menjaga ketertiban tersebut dibutuhkan pihak yang memiliki kewenangan untuk mengatur, mengarahkan, bahkan memberikan sanksi apabila terjadi pelanggaran di jalan raya, pihak tersebut tidak 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.

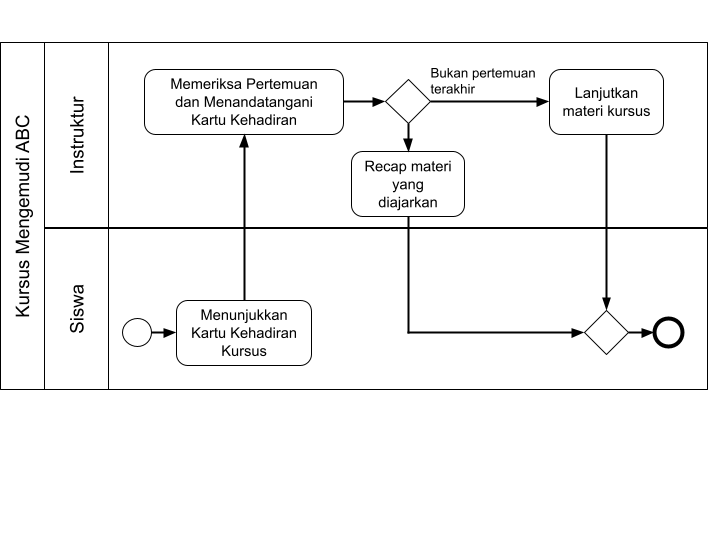
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Dokumen Persyaratan** | **Kursus ABC** | **Sie Bersaudara** | **“Hafiz”** |
| 1 | Akte pendirian perusahaan (Lembaga Kursus) atau KTP (Perorangan) | ✔ | ✔ | ✔ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Dokumen Persyaratan** | **Kursus ABC** | **Sie Bersaudara** | **“Hafiz”** |
| 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 |  |  | ✔ |

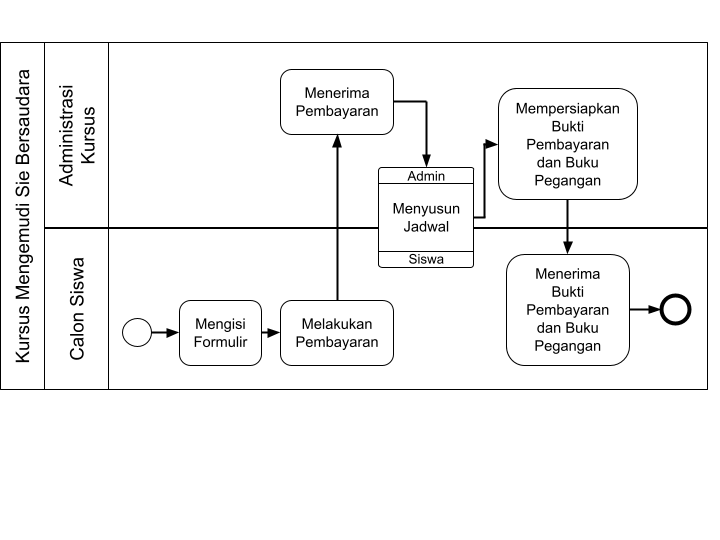
Pada saat kami melakukan penelitian ini belum ada standar atau peraturan yang mengatur atau mewajibkan para penyedia jasa kursus mengemudi untuk memiliki suatu proses tertentu. Sehingga langkah selanjutnya adalah kami menanyakan bagaimana proses bisnis yang ada pada ketiga kursus mengemudi tersebut. Agar lebih mudah dipahami, kami bagi proses bisnis menjadi 2 alur, Pendaftaran Kursus dan Proses Kursus Mengemudi.



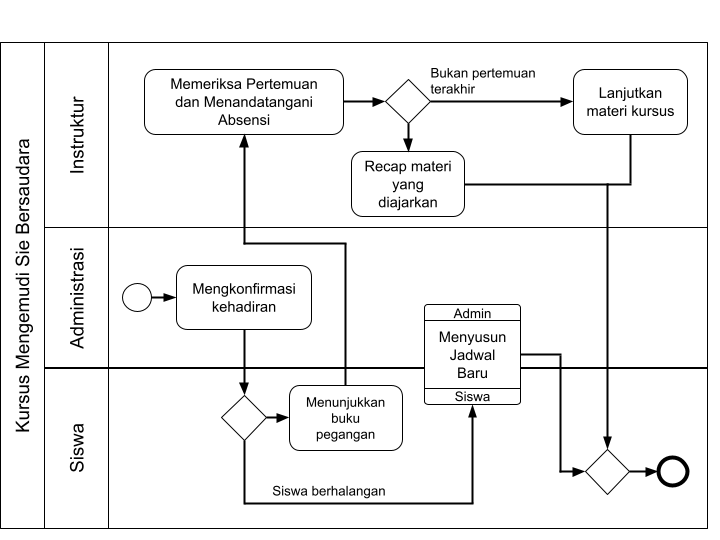
Untuk proses pendaftaran pada Kursus ABC dimulai dengan calon siswa memilih kelas kursus kemudian menyusun jadwal, jika calon siswa merasa sudah puas proses selanjutnya adalah pembayaran, dimana nominal uang yang dibayarkan harus diperiksa terlebih dahulu oleh pihak administrasi Kursus ABC, jika sudah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



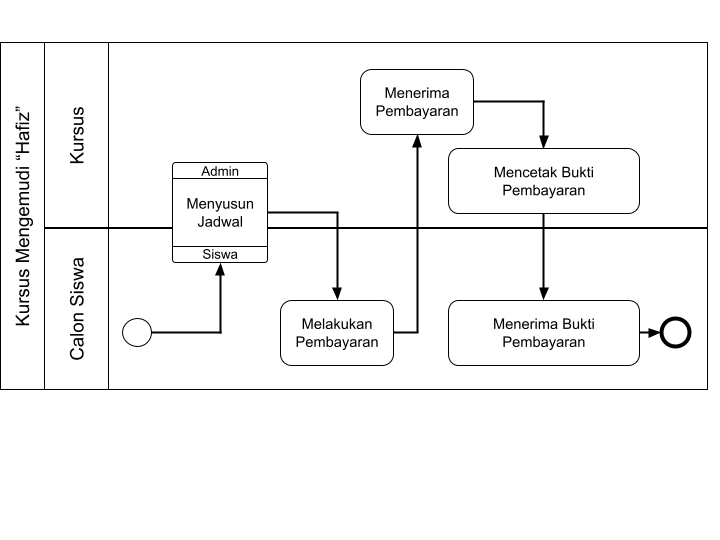
Proses kursus di kursus mengemudi ABC dimulai dengan siswa hadir ke tempat kursus atau siswa bisa memilih untuk dijemput oleh instruktur kursus di rumahnya. Sebelum memulai praktik, siswa menunjukkan kartu kehadiran kursus ke instruktur untuk ditandatangani kehadirannya bersamaan dengan instruktur memeriksa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



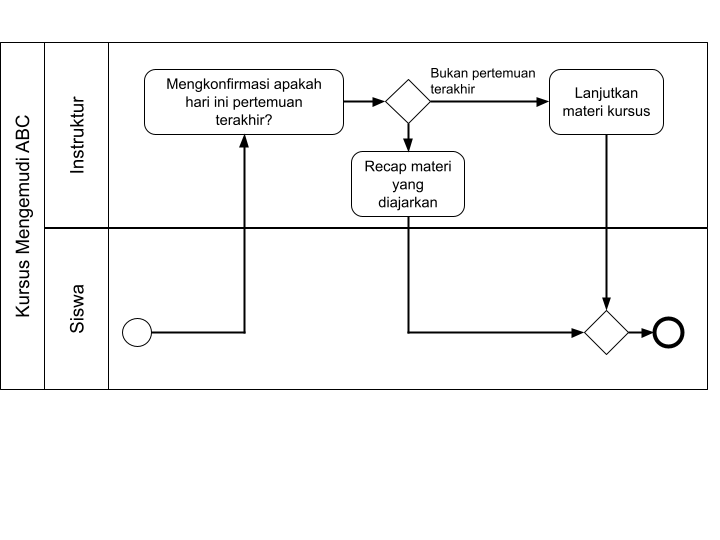
*Flowchart* di atas menunjukkan alur proses pendaftaran di Kursus Sie Bersaudara, dimulai dengan siswa mengisi formulir, di dalam formulir tersebut terdapat opsi kelas dan deskripsi dari masing-masing kelas, setelah mengisi formulir, siswa melakukan pembayaran, sama dengan kursus ABC, pihak administrasi 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.



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.



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.



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.

## 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.

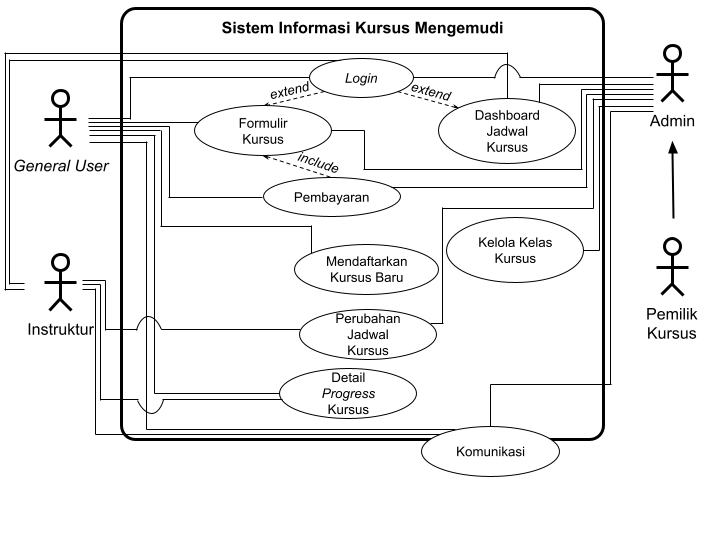
Kebutuhan bisnis/pemasaran memberikan pemahaman tentang kebutuhan bisnis atau kondisi *marketplace* yang ada. Kebutuhan fungsional memberikan pemahaman akan fitur-fitur yang dibutuhkan untuk mendukung kebutuhan bisnis atau pemasaran. Kebutuhan fungsional juga terkadang merupakan hasil dari hasil riset 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.

|  |  |  |
| --- | --- | --- |
| **No.** | **Kebutuhan / Persyaratan** | **Aspek** |
| Sisi Admin Kursus (Lembaga) / Pemilik (Perorangan & Lembaga) | | |
| 1 | Fungsi *Login* & *Logout* sebagai admin / pemilik | Fungsional |
| 2 | Dashboard yang berisi jadwal kursus setiap Instruktur per hari | Fungsional |
| 3 | Menambah kelas kursus beserta deskripsi, harga, jumlah pertemuan, benefit, dan kategori kelas kursus | Fungsional |
| 4 | Mengubah nama, deskripsi, harga, jumlah | Fungsional |
| **No.** | **Kebutuhan / Persyaratan** | **Aspek** |
|  | pertemuan, benefit, dan kategori kelas kursus |  |
| 5 | Menghapus / menonaktifkan kelas kursus | Fungsional |
| 6 | Menampilkan detail *progress* 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 *Login* & *Logout* sebagai Instruktur | Fungsional |
| 2 | Dashboard yang berisi jadwal kursus per hari | Fungsional |
| 3 | Menampilkan daftar siswa aktif | Fungsional |
| 4 | Menampilkan detail *progress* siswa | Fungsional |
| 5 | Mengajukan atau mengkonfirmasi perubahan jadwal kursus kepada siswa | Kegunaan |
| 6 | Menghubungi siswa | Fungsional |

|  |  |  |
| --- | --- | --- |
| **No.** | **Kebutuhan / Persyaratan** | **Aspek** |
| Sisi *General User* | | |
| 1 | Fungsi *Login* & *Logout* sebagai *General User* | Fungsional |
| 2 | Dashboard yang berisi rekomendasi kelas kursus dan *progress* kursus yang sedang aktif | Fungsional |
| 3 | Mendaftarkan diri ke kursus mengemudi | Fungsional |
| 4 | Menghubungi admin / pemilik kursus dan instruktur | Fungsional |
| 5 | Menampilkan detail *progress* 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 |
| Kebutuhan Teknis | | |
| 1 | Sistem mampu dijalankan melalui Smartphone semua pihak | Teknologi |
| 2 | Sistem mudah diakses di web/internet | Teknologi |

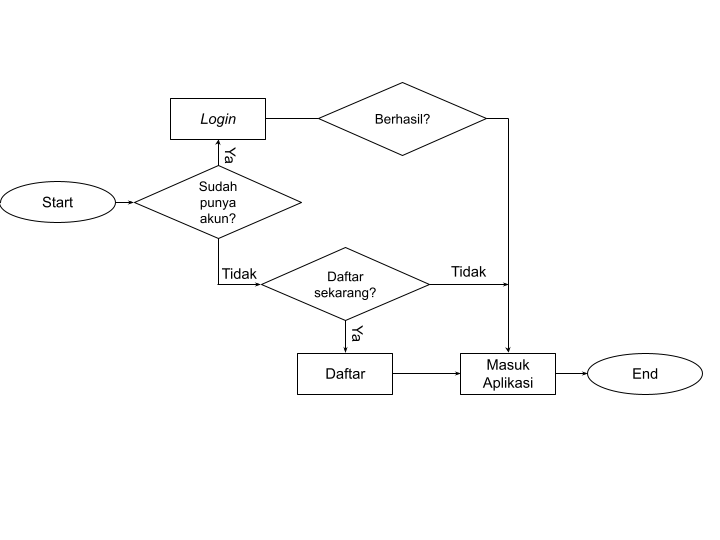
## Use Case

Setelah mengetahui analisa kebutuhan diatas, langkah selanjutnya adalah menerjemahkannya ke bentuk *Use Case Diagram*. Hal ini ditujukan untuk memberikan alur yang lebih jelas tentang interaksi-interaksi yang dilakukan user dengan sistem untuk masing-masing proses yang ada pada sistem, serta sekali lagi memeriksa apakah semua fungsi yang disebutkan diatas sudah terakomodir semuanya. Dari proses-proses yang sudah kami jelaskan diatas, langkah selanjutnya adalah merancang alur sistem baru yang dapat mengakomodasi perbedaan-perbedaan dari satu lembaga kursus dengan lembaga kursus lainnya. Untuk memecah kompleksitas dan menghindari kesalahpahaman, kami akan menjelaskan setiap *use case* dengan *Flowchart* dan *Use Case Scenario.*





### *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.

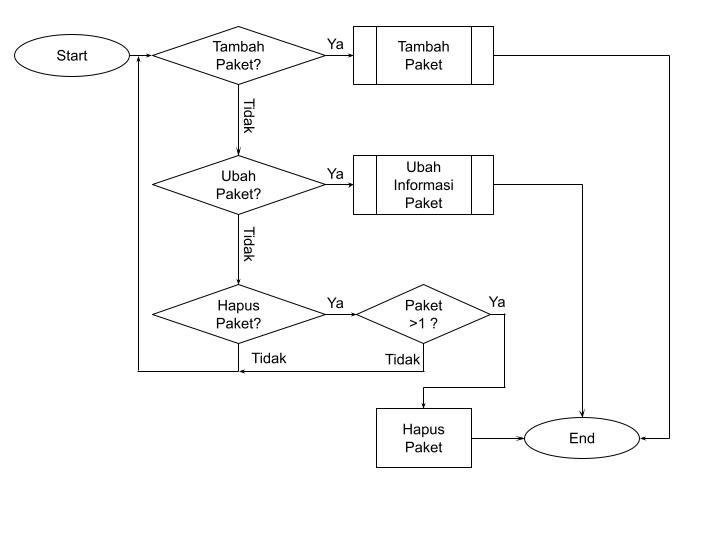
|  |  |  |
| --- | --- | --- |
| **Kode Use Case** | UC\_login\_untuk\_semua\_user | |
| **Nama Use Case** | *Login* | |
| **Aktor** | *General User*, Pemilik / Admin, Instruktur | |
| **Deskripsi** | Skenario untuk menyimpan data dari segala aktivitas yang dilakukan oleh Aktor | |
| **Kondisi Awal** | Sistem baru selesai memuat tampilan awal | |
| **Kondisi Akhir** | Aktor berada di halaman beranda/*dashboard* | |
| **Alur Kejadian Normal** | **Aktor** | **Sistem** |
|  | 1. Sistem menampilkan tiga tombol (*Login*, Daftar, dan |
|  | **Aktor** | **Sistem** |
|  | Nanti) |
| 2. Aktor memilih tombol *Login* |  |
|  | 3. Sistem menampilkan *form* *Login* |
| 4. Aktor memasukkan data akun yang tersimpan di database |  |
|  | 5. Sistem memverifikasi data yang diinputkan Aktor |
|  | 6. Sistem mengarahkan Aktor |
| **Alur Kejadian Alternatif** | 2a. Aktor memilih tombol “Daftar”  2b. Aktor memilih tombol “Nanti” |  |
|  | 3a. Sistem menampilkan *form* daftar akun  3b. Sistem menganggap Aktor sebagai *General User* |
| 4a. Aktor memasukkan data akun baru  4c. Aktor memasukkan data yang tidak ada di database | 4b. Sistem mengalihkan Aktor ke beranda/*dashboard* |
|  | **Aktor** | **Sistem** |
|  | 5a. Sistem menyimpan data baru yang diinputkan Aktor  5c. Sistem tidak menemukan data yang diinputkan Aktor |
|  | 6a. Sistem mengalihkan Aktor ke beranda/*dashboard*  6c. Sistem mengalihkan Aktor ke tampilan awal dan mengembalikan pesan *error* ke Aktor |

### 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.

|  |  |  |
| --- | --- | --- |
| **Kode Use Case** | UC\_tampilkan\_jadwal\_kursus | |
| **Nama Use Case** | Menampilkan *dashboard* jadwal kursus | |
| **Aktor** | Pemilik / Admin, Instruktur | |
| **Deskripsi** | Skenario untuk menampilkan jadwal kursus per hari | |
| **Kondisi Awal** | Sistem selesai memverifikasi data akun yang diinputkan Aktor pada proses login | |
| **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 *default*, 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 |
| 4b. Aktor memilih opsi untuk tampilkan jadwal dalam 30 |  |
|  | **Aktor** | **Sistem** |
| hari |  |
|  | 5b. Sistem menampilkan jadwal dimulai dengan hari ini sampai 29 hari kedepan |

### Pengelolaan 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.

|  |  |  |
| --- | --- | --- |
| **Kode Use Case** | UC\_pengelolaan\_kelas\_kursus | |
| **Nama Use Case** | Kelola kelas Kursus | |
| **Aktor** | Pemilik / Admin, Instruktur | |
| **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 *form* tambah kelas |
| 3. Aktor mengisi semua *form* dengan lengkap | 3. Sistem memverifikasi bahwa semua *form* terisi |
| 4. Aktor menekan tombol “Tawarkan kelas” |  |
|  | 5. Sistem menyimpan *entry* yang dilakukan Aktor |
|  | 6. Sistem mengarahkan Aktor ke halaman kursus dan menampilkan kelas yang baru ditambahkan pada |

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

|  |  |  |
| --- | --- | --- |
|  | **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 *entry* yang dilakukan Aktor |
|  | 6a. Sistem menyimpan *entry* yang dilakukan Aktor  6c. Sistem mengalihkan Aktor ke halaman kursus |
|  | 7a. Sistem mengalihkan Aktor ke halaman kursus |

### Menampilkan Detail *Progress* Kursus

Pihak kursus dan siswa dengan kursus aktif dapat melihat detail dari kursus yang diikuti. Halaman detail *progress* ini guna menunjukkan sedang berada di pertemuan ke berapakah siswa saat ini, didalamnya siswa juga dapat mempelajari teori mengemudi lebih lanjut dengan memilih ‘Baca Panduan’ dan mengerjakan ‘Quiz’ yang disediakan oleh Sistem.

|  |  |  |
| --- | --- | --- |
| **Kode Use Case** | UC\_detail\_progress\_kursus\_instruktur | |
| **Nama Use Case** | Menampilkan detail *progress* kursus | |
| **Aktor** | Pemilik / Admin, Instruktur | |
| **Deskripsi** | Skenario untuk menampilkan detail kursus aktif | |
| **Kondisi Awal** | Aktor berada di *dashboard* | |
| **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 *dashboard* |  |
|  | 2. Sistem menampilkan data kursus terkait dengan siswa yang dipilih Aktor |
|  | 3. Sistem menampilkan daftar capaian yang sudah dilakukan oleh siswa |
| **Alur Kejadian Alternatif** | - | - |

|  |  |  |
| --- | --- | --- |
| **Kode Use Case** | UC\_detail\_progress\_kursus\_siswa | |
| **Nama Use Case** | Menampilkan detail *progress* kursus | |
| **Aktor** | *General User* | |
| **Deskripsi** | Skenario untuk menampilkan detail kursus aktif | |
| **Kondisi Awal** | Aktor berada di *dashboard* | |
| **Kondisi Akhir** | Aktor bisa melakukan perubahan jadwal, membaca panduan teori mengemudi, menyelesaikan Quiz, dan proses-proses lain | |
| **Alur Kejadian Normal** | **Aktor** | **Sistem** |
| 1. Aktor menekan ‘Progress Kursus’ pada bagian atas *dashboard* |  |
|  | 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 |

### Komunikasi

Baik pihak kursus dan siswa dapat menghubungi satu sama lain dengan aplikasi Whatsapp diluar sistem, sistem sebatas menempelkan link pada tombol yang disediakan pada halaman ‘Detail Kursus’ yang selanjutnya bisa diklik oleh Aktor yang kemudian akan diarahkan Whatsapp untuk mengkonfirmasi apakah Aktor benar ingin melakukan komunikasi dengan Nomor Telepon bersangkutan.

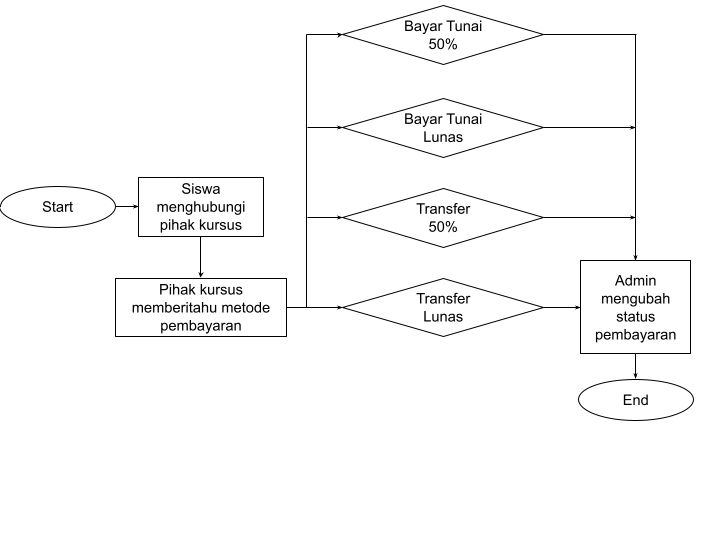
|  |  |  |
| --- | --- | --- |
| **Kode Use Case** | UC\_komunikasi | |
| **Nama Use Case** | Proses Komunikasi antar Aktor | |
| **Aktor** | *General User*, Instruktur, Pemilik/Admin | |
| **Deskripsi** | Skenario untuk berkomunikasi antar Aktor | |
| **Kondisi Awal** | Aktor berada di tampilan ‘Detail Kursus’ | |
| **Kondisi Akhir** | Aktor berada di tampilan percakapan Whatsapp dengan Aktor tujuan | |
| **Alur Kejadian Normal** | **Aktor** | **Sistem** |
| 1. Aktor menekan tombol ‘Hubungi Siswa/Instruktur/Admin’ |  |
|  | 2a. Untuk nomor Whatsapp Siswa, sistem mengambil nomor dari *Form* pendaftaran kursus  2b. Untuk nomor Whatsapp Instruktur, sistem mengambil nomor dari *Form* undangan mengajar yang dikirimkan Pemilik/Admin  2c. Untuk nomor Whatsapp Pemilik/Admin didapatkan dari *Form* pendaftaran kursus |
|  | **Aktor** | **Sistem** |
|  | baru yang dikirimkan Aktor |
|  | 3. Sistem mengarahkan Aktor ke halaman konfirmasi percakapan Whatsapp |
| 4. Aktor menekan tombol “Continue to Chat” pada halaman konfirmasi |  |
| **Alur Kejadian Alternatif** | - | - |

### Siswa mendaftar kelas kursus

Untuk alur mendaftar kursus, Aktor dengan peran *General User* memilih kelas kursus yang dimaksud terlebih dahulu, selanjutnya sistem akan melakukan pengecekkan 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.

|  |  |  |
| --- | --- | --- |
| **Kode Use Case** | UC\_daftar\_kelas\_kursus | |
| **Nama Use Case** | Mendaftar kelas kursus | |
| **Aktor** | *General User* & Pemilik/Admin | |
| **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 *entry* 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 |  |
|  | 4. Sistem mengalihkan Calon Siswa ke beranda/*dashboard* |

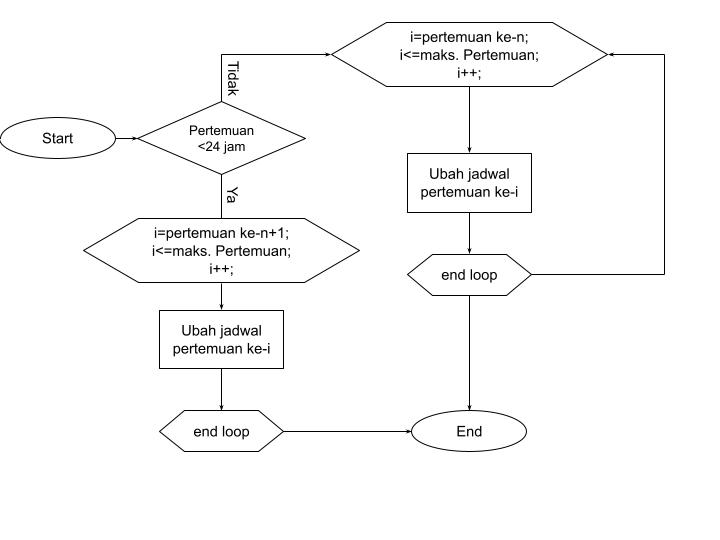
### Pembayaran



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

|  |  |  |
| --- | --- | --- |
| **Kode Use Case** | UC\_status\_pembayaran | |
| **Nama Use Case** | Status Pembayaran | |
| **Aktor** | *General User* & Pemilik/Admin | |
| **Deskripsi** | Skenario pemberitahuan status pembayaran | |
| **Kondisi Awal** | Calon siswa baru saja menyelesaikan *form* pendaftaran | |
| **Kondisi Akhir** | Calon siswa mendapatkan informasi status pembayaran | |
| **Alur Kejadian Normal** | **Aktor** | **Sistem** |
|  | 1. Sistem mengarahkan Calon siswa ke halaman detail kursus |
|  | **Aktor** | **Sistem** |
| 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 |  |
|  | 6. Sistem menampilkan status pembayaran yang dipilih Admin |
| **Alur Kejadian Alternatif** | - | - |

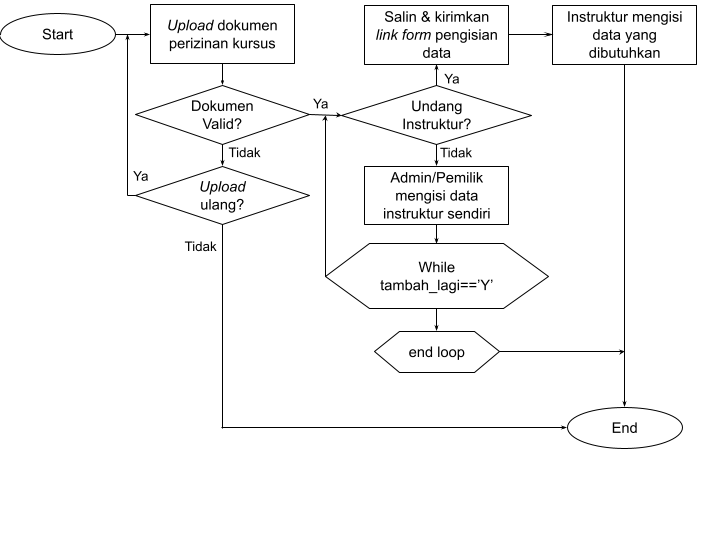
### Perubahan Jadwal



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.

|  |  |  |
| --- | --- | --- |
| **Kode Use Case** | UC\_perubahan\_jadwal | |
| **Nama Use Case** | Pengajuan perubahan jadwal | |
| **Aktor** | *General User* & Pemilik/Admin | |
| **Deskripsi** | Skenario pengajuan perubahan jadwal | |
| **Kondisi Awal** | Siswa berada di halaman detail *progress* 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 |
|  | 5a. Jika pertemuan selanjutnya < 24 jam, jadwal yang dapat diubah dimulai dari pertemuan setelah pertemuan mendatang  5b. Jika pertemuan selanjutnya > 24 jam, perubahan jadwal dapat dimulai dari pertemuan mendatang |
| 6. Aktor menyesuaikan jadwal yang tersedia |  |
| 7. Aktor menekan tombol ‘Ajukan Perubahan Jadwal’ |  |
|  | 8. Sistem mengirim |
|  | **Aktor** | **Sistem** |
|  | pemberitahuan ke Aktor lain untuk dikonfirmasi pengajuan jadwal yang dikirim |
| 9. Aktor lainnya mengkonfirmasi perubahan jadwal |  |
|  | 10. Sistem menyimpan perubahan jadwal |
| **Alur Kejadian Alternatif** | 6. Aktor membatalkan perubahan jadwal |  |
|  | 7. Sistem mengarahkan Aktor ke halaman detail *progress* kursus |
| 9. Aktor lainnya menolak perubahan jadwal |  |
|  | 10. Sistem mengirimkan pemberitahuan penolakan pengajuan jadwal ke Aktor |

### Mendaftarkan 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*.

|  |  |
| --- | --- |
| **Kode Use Case** | UC\_pendaftaran\_kursus\_baru |
| **Nama Use Case** | Pendaftaran Kursus Baru |
| **Aktor** | Instruktur, *General User* & *System* Admin |
| **Deskripsi** | Skenario untuk mendaftarkan jasa kursus mengemudi baru |
| **Kondisi Awal** | Calon Pemilik berada di halaman beranda/*dashboard* |
| **Kondisi Akhir** | Calon Pemilik berada pada halaman beranda/*dashboard* Pemilik/Admin |

|  |  |  |
| --- | --- | --- |
| **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 *upload* dokumen perizinan |
| 5. Calon Pemilik mengupload dokumen perizinan penyelenggaraan kursus |  |
|  | 6. *System* 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 |

|  |  |  |
| --- | --- | --- |
|  | **Aktor** | **Sistem** |
|  | 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 |  |
| 10a. Calon Pemilik mengisi semua data Instruktur dengan lengkap  10b. Calon Pemilik mengirimkan link *form* penambahan Instruktur ke Calon Instruktur bersangkutan |  |
| 11b. Calon Instruktur menerima *link* undangan mengajar | 11a. Sistem menyimpan *entry* data dari Calon Pemilik, kemudian mengarahkan Calon Pemilik ke tampilan beranda/*dashboard* |
|  | 12b. Sistem menampilkan |

|  |  |  |
| --- | --- | --- |
|  | **Aktor** | **Sistem** |
|  | halaman pendaftaran akun untuk diisi Calon Instruktur |
| 13b. Calon Instruktur mengisi *form* penambahan Instruktur |  |
|  | 14b. Sistem menyimpan *entry* yang dilakukan Instruktur |
| **Alur Kejadian Alternatif** |  | 6. *System* admin menolak keaslian dokumen atau dokumen yang di-*upload* dianggap kurang jelas |
|  | 7. Sistem mengirim pesan gagal ke Calon Pemilik kemudian mengarahkan Calon Pemilik untuk melakukan *upload* ulang |
| 10a. Calon Pemilik tidak mengisi *form* dengan lengkap atau membatalkan pengisian |  |
| 13b. Instruktur tidak mengisi *form* dengan lengkap atau membatalkan pengisian | 11a. Sistem mengalihkan Calon Pemilik ke halaman profil |
|  | 14b. Sistem mengalihkan Calon Instruktur ke halaman |
|  | **Aktor** | **Sistem** |
|  | profil |

## Tahapan *Prototyping*

Seperti yang disebutkan pada bab 2, metode *prototyping* dibagi menjadi 4 fase, fase perencanaan memiliki 3 tahapan: verifikasi kebutuhan; membuat alur tugas; dan menentukan konten dan tingkat ketelitian atau *fidelity* dari *prototype*. Fase spesifikasi terdiri dari 3 tahapan: menentukan karakteristik *prototyping*; memilih metode *prototyping*; memilih alat *prototyping*. Selanjutnya fase desain dimana fase ini memiliki 2 tahapan saja: merumuskan kriteria desain dan membangun *prototype* tersebut. Fase hasil setelah *prototype* dibangun tahapan selanjutnya yaitu meninjau ulang hasil desain, menguji/memvalidasi desain, dan melanjutkannya ke tahap pemrograman.



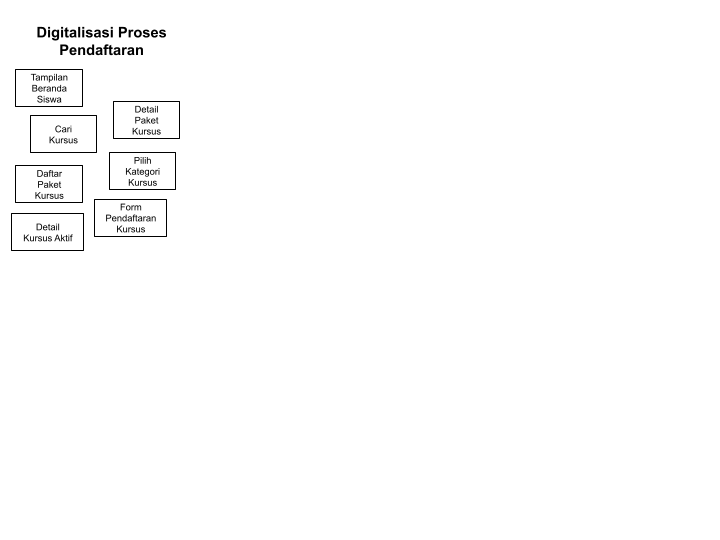
### Verifikasi Kebutuhan

Sebagaimana yang disebutkan pada bab 2, kebutuhan *prototype* merupakan sebagian kecil dari analisa kebutuhan dari keseluruhan perangkat lunak. Sebelumnya pada bab 1 kami juga menyebutkan beberapa masalah yang ingin kita tanggulangi dengan mengembangkan perangkat lunak ini, masalah pertama adalah kebanyakan masyarakat umum takut biaya yang disiapkan terlalu besar dan manfaat yang didapatkan dari kursus mengemudi tidak sepenuhnya disadari oleh masyarakat umum. Masalah selanjutnya sulitnya pelajar kursus untuk menerima informasi terbaru tentang kursus mengemudi yang diikuti juga seringkali dikeluhkan. Proses registrasi yang dianggap masih konvensional juga terkadang membutuhkan waktu lama. Pengelolaan jadwal kursus yang masih menggunakan media kertas juga terkadang menyulitkan pihak administrasi melacak jadwal-jadwal kursus yang padat. Dari penjelasan singkat ini dapat disimpulkan fitur-fitur inti dari sistem informasi untuk penyedia jasa kursus mengemudi adalah:

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Nama Kebutuhan** | **Aspek Kebutuhan** | **Prioritas** |
| 1. | Menampilkan biaya dan manfaat dari kelas kursus mengemudi | Bisnis | Tinggi |
| 2. | Menampilkan informasi terkait kelas kursus yang diikuti | Fungsional | Menengah |
| 3. | Digitalisasi proses pendaftaran | Fungsional | Menengah |
| 4. | Manajemen jadwal kursus | Fungsional | Tinggi |

### Alur Tugas

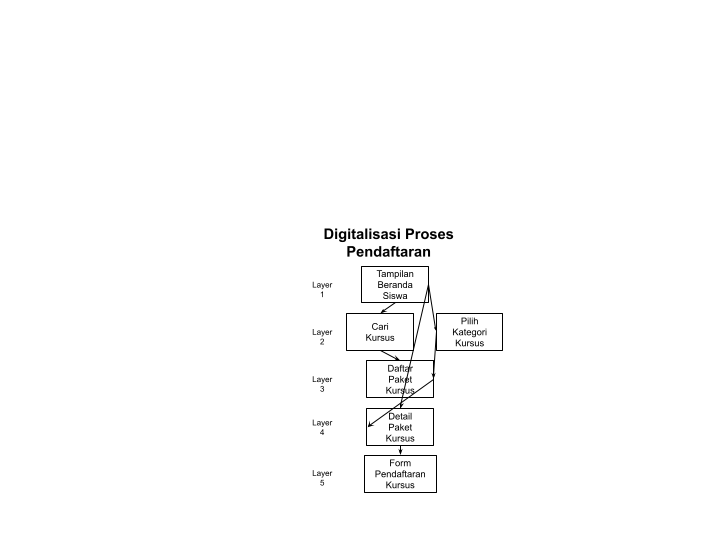
Berdasarkan buku *Effective Prototyping for Software Makers* (Arnowitz et al., 2007), pembuatan alur tugas memungkinkan kita untuk mendesain dan mengalokasikan tugas-tugas ke rangkaian tampilan/antarmuka sistem. Jonathan Arnowitz, Michael Arent, dan Nevin Berger lebih lanjut mengusulkan penggunaan *dependency diagram*. Diagram tersebut menggambarkan rangkaian tugas-tugas atau tahapan-tahapan dalam menyelesaikan sebuah tugas. Salah satu turunan dari *dependency diagram* adalah *task layer map* (peta lapisan tugas). Pembuatan peta lapisan tugas dibagi menjadi 4 tahapan.



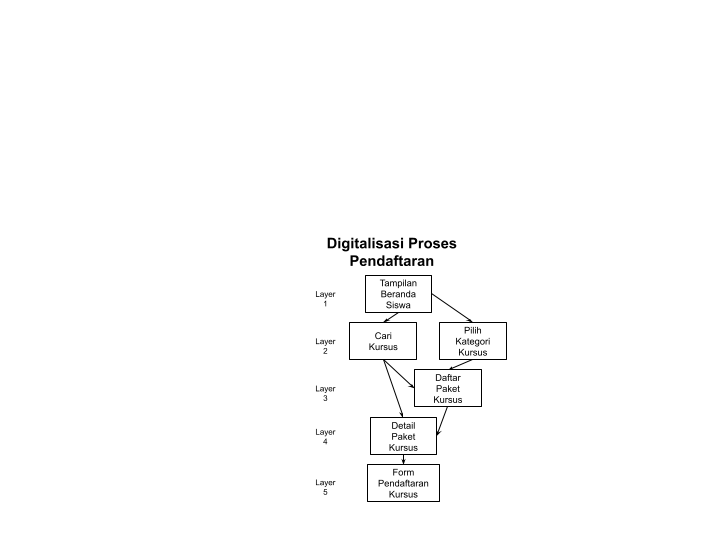
Tahapan pertama adalah membuat daftar tugas, sebelum membuat daftar tugas kita bisa mulai dengan melakukan *brainstorm* terhadap langkah-langkah yang dibutuhkan di perangkat lunak, kita bisa mendapatkan daftar tugas tersebut dari tugas serupa di kehidupan sehari-hari, *use case* yang di rancang, dan alur perpindahan data yang diasumsikan.



Tahapan kedua ialah mengidentifikasi hubungan antar tugas. Setelah kita buat daftar tugas pada perangkat lunak, kita tarik garis yang menghubungkan tugas-tugas yang bergantung satu sama lain. Untuk mengetahui arah garis, kita bisa membedakan titik awal dan titik akhir dengan menambahkan arah panah untuk garis yang menandakan titik akhir.



Tahapan ketiga pindahkan tugas-tugas berdasarkan urutan ketergantungannya. Untuk lapisan 1, lapisan paling atas, adalah tugas yang tidak bergantung pada tugas lain, atau tidak ada garis yang mengarah ke tugas tersebut. Lapisan 2 berisi tugas yang memiliki ketergantungan dengan tugas dari lapisan 1 dan seterusnya. Lapisan terakhir nantinya berisi tugas-tugas yang tidak di bergantungkan oleh tugas-tugas lain.



Tahapan keempat adalah Normalisasi garis hubungan antar tugas. Maksudnya hapus garis yang secara tidak langsung sudah diwakilkan oleh rangkaian tugas di lapisan sebelumnya.

### Penentuan Konten dan Tingkat Ketelitian *Prototype*

Tingkat ketelitian (*fidelity*) dari sebuah *prototype* dibagi menjadi dua, tingkat ketelitian rendah dan tingkat ketelitian tinggi. *Prototype* dengan tingkat ketelitian rendah menciptakan representasi awal dari konten yang akan ada di dalam perangkat lunak nantinya. Sedangkan *prototype* dengan ketelitian tinggi terlihat seperti perangkat lunak yang sudah *final*.

|  |  |
| --- | --- |
|  |  |

Untuk perancangan perangkat lunak yang kami lakukan, agar para pihak-pihak yang berkepentingan nantinya mudah memberikan masukan terhadap desain yang kami ajukan dan untuk menghindari kesalahpahaman kami memutuskan untuk memilih *prototype* dengan tingkat ketelitian tinggi.

|  |  |
| --- | --- |
|  |  |

Penentuan penampilan konten pada *prototype* juga dibagi menjadi 2, penampilan konten secara penuh dan penampilan konten-konten inti. Penampilan konten inti memungkinkan pelaku atau penanggungjawab *prototype* untuk menggunakan teks/gambar pengganti, contoh teks pengganti populer seperti *lorem ipsum* dll. Dengan alasan yang sama, kami memutuskan untuk memilih penampilan konten secara penuh.

### Penentuan Karakteristik *Prototype*

Penentuan karakteristik *prototype* ini wajib dilakukan sebelum proses pembuatan *prototype* dimulai, dan dianggap penting karena pelaku atau penanggung jawab *prototype* nantinya akan memahami seperti apa *prototype* yang dibutuhkan dan juga bagi para manajer produk bisa mengetahui apa saja yang akan dibuat *prototype*-nya. Dalam buku *Effective Prototyping for Software Makers* disebutkan 8 karakteristik *prototype*:

1. **Partisipan *prototype***: internal atau eksternal, partisipan yang akan melihat atau bereksperimen dengan *prototype*; tidak harus sama dengan pengguna akhir nantinya
2. **Tahapan**: tahap awal atau tahap menengah atau tahap akhir, tahapan proses pembuatan dan pengembangan perangkat lunak dimana proses *prototyping* dirasa optimal untuk dilakukan
3. **Kecepatan**: cepat atau mendetail, mana yang menjadi fokus/tujuan *prototype*, secepat mungkin atau sedetail mungkin
4. **Usia *prototype***: singkat atau biasa-biasa saja atau selama dibutuhkan, berapa lama sebuah *prototype* akan digunakan; apakah *prototype* digunakan sebatas untuk mengetahui konsep atau untuk dijadikan acuan bagi *programmer* selama proses pengembangan
5. **Ekspresi**: konseptual atau eksperimental, seberapa abstrak atau jelas desain visual dan interaksi yang dapat diekspresikan oleh sebuah *prototype*
6. **Metode**: naratif atau interaktif, metode partisipasi yang diharapkan dari seorang pengguna, apakah pasif (naratif) atau aktif (interaktif)
7. **Perantara**: fisik atau digital, perantara dimana *prototype* dihasilkan
8. **Tingkat Ketelitian**: rendah atau menengah atau tinggi, sebuah matriks tingkat ketelitian untuk setiap konten yang ada pada *prototype*

Dengan karakteristik yang disebutkan diatas, kita tidak bisa sekedar memilih 1 karakteristik dan tidak memilih karakteristik lain. Melainkan, kita bisa menggabungkan lebih dari 1 karakteristik yang dirasa paling tepat untuk proses *prototyping* kedepannya. Sebagai contoh, *prototype* bisa berbentuk fisik atau digital, atau *prototype* bisa saja hanya mengusung metode naratif atau khusus metode interaktif, namun kebanyakan *prototype* yang sering kita temui mengkombinasikan kedua-duanya. Dengan memahami konsep ini selanjutnya kita dapat menentukan karakteristik *prototype* perangkat lunak yang akan kami kembangkan sebagai berikut:

|  |  |  |
| --- | --- | --- |
| **Karakteristik *Prototype*** |  |  |
| **Partisipan** | Siapa yang akan menggunakan perangkat lunak? | Siapa yang akan melihat *prototype* ini? |
| **Target Kelompok** | Pengguna Akhir | Pihak Kursus dan Calon Pengguna |
| **Internal/Eksternal** | Eksternal | Eksternal |
| **Latar Belakang** | Bermacam-macam | Bermacam-macam |
| **Memahami proses kursus mengemudi?** | Beberapa | Beberapa |
| **Keahlian khusus dibutuhkan untuk menggunakan sistem?** | Tidak | Beberapa |
| **Syarat-syarat lain?** |  | |
| **Apakah partisipan dari *prototype* kami harus memahami situasi/kondisi yang mungkin dialami pengguna akhir?** | Tidak wajib | |
| **Apakah partisipan dari *prototype* secara langsung terlibat pada proses pengembangan perangkat lunak?** | Tidak | |
| **Tahapan proses pengembangan perangkat lunak saat ini** | Awal | |

|  |  |
| --- | --- |
| **Berapa lama waktu yang dibutuhkan untuk membuat *prototype*?** | 40 hari |
| **Kecepatan** | Mendetail |
| **Untuk berapa lama *programmer* merujuk *prototype* ini?** | Sampai perangkat lunak memasuki proses pengujian |
| **Berapa lama *prototype* akan digunakan, dilihat, atau dipresentasikan ke partisipan** | Sebelum proses pemrograman dilakukan |
| **Usia *prototype*** | Selama dibutuhkan |
| **Ekspresi** | Konseptual dan Eksperimental |
| **Metode** | Naratif dan Interaktif |
| **Perantara** | Digital |

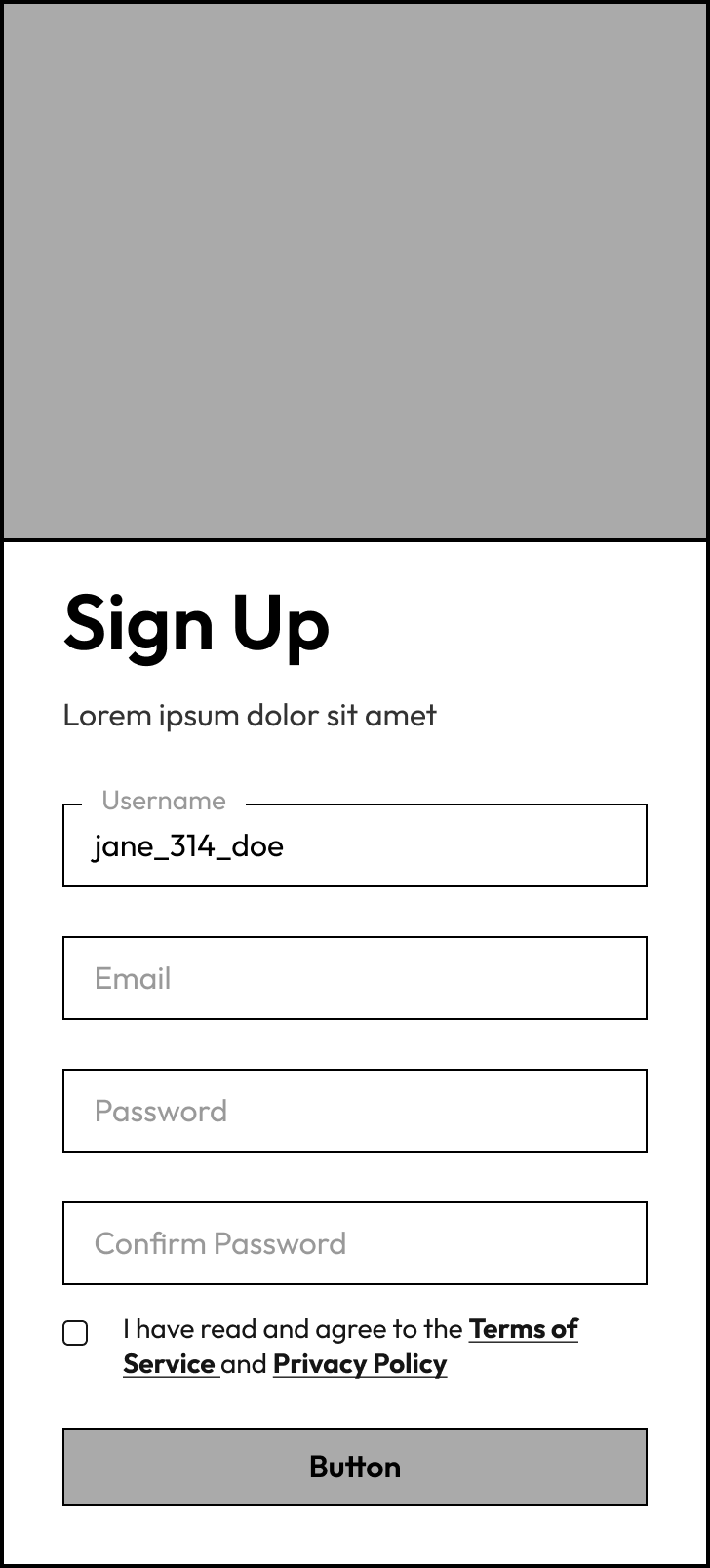
### Memilih sebuah metode *prototyping*

Saat ini sudah banyak sekali macam-macam metode *prototyping*, metode-metode ini diciptakan dengan ciri-cirinya masing-masing dan untuk tujuan yang berbeda-beda pula. Metode-metode *prototyping* yang populer diantaranya adalah:

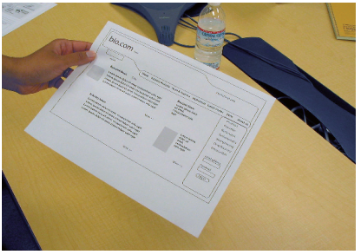
1. ***Card Sorting***: metode ini bersifat abstrak, interaktif, dan partisipatif yang biasanya dilakukan pada awal proses pengembangan perangkat lunak. Tujuannya adalah untuk menentukan bagaimana susunan struktur informasi dan navigasi yang paling baik. Partisipan dari hasil akhir metode ini adalah orang-orang yang tergabung dalam tim desain internal, yang diharapkan mampu memahami isu-isu desain informasi yang dihadapi dengan melakukan metode ini. Namun, sebelum mendapatkan hasil tersebut, tim desain informasi melakukannya bersama dengan target pengguna.



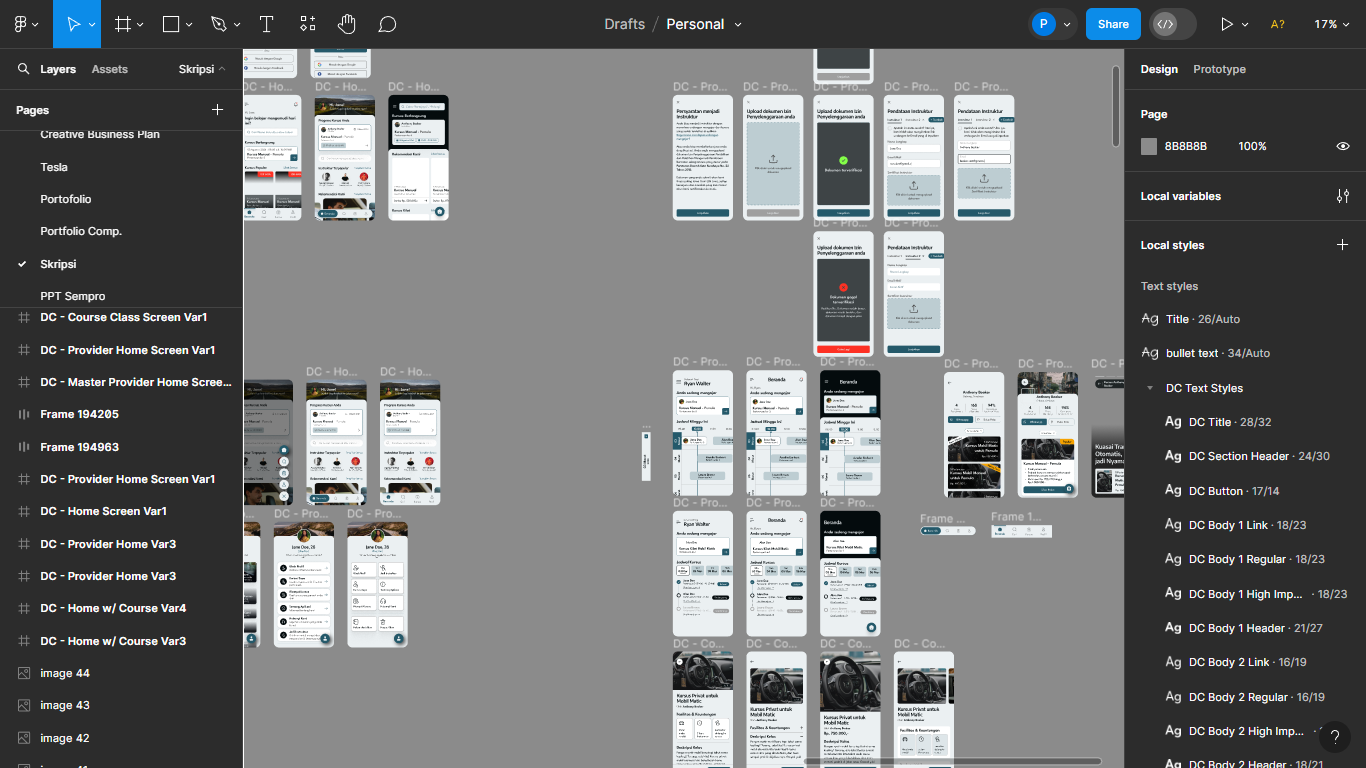
1. ***Wireframe Prototyping***: metode ini tergolong metode naratif, *wireframe* biasanya dibuat pada awal proses desain. Narasi yang dihasilkan biasanya berasal dari *use case* atau skenario, terkadang skenario tersebut digunakan sebagai *storyboard* juga. Metode ini menyuguhkan partisipan dengan sketsa tingkat tinggi, memvisualisasi asumsi konseptual tentang struktur produk dan interaksi secara umum. Tujuan utama dari metode ini yaitu menyepakati konsep dasar dan arah desain yang nantinya menjadi acuan bagi tim desain yang selanjutnya akan mengerjakan desain-desain yang lebih detail.



1. ***Paper Prototyping***: merupakan salah satu *prototype* interaktif yang berbentuk maket kertas dari tampilan/antarmuka pengguna. Tampilan ini biasanya sudah sepenuhnya fungsional, meskipun fungsionalitas nya dimaketkan diatas kertas. Metode ini memberikan kita kesempatan untuk menguji desain yang sudah dibuat ke pengguna sesungguhnya. Target partisipan metode ini, yang pertama adalah target pengguna sesungguhnya untuk menguji desain, dan yang kedua adalah *programmer* untuk menginformasikan kepada mereka bahwa desain ini merupakan desain *final*.



1. ***Digital Prototyping***: metode ini merupakan bentuk digital dari metode *paper prototype* diatas. Namun, *digital prototype* ini dapat berupa *prototype* dengan ketelitian rendah, tampilan naratif yang dapat diklik untuk memvisualisasikan sebuah konsep desain ke tampilan interaktif dengan ketelitian tinggi yang selanjutnya dapat digunakan sebagai spesifikasi tampilan pengguna. Tujuan dari *digital prototyping* hampir sama dengan *paper prototyping* dimana mereka bisa digunakan untuk memahami alur tugas dan konteks penggunaan, memvalidasi asumsi pada skenario, kebutuhan bisnis, dan gambaran pengguna, membantu penyusunan urutan tugas dan arah desain interaksi, meningkatkan detail *prototype* yang sebelumnya berupa sketsa menjadi hampir serupa dengan produk akhir, dan lain lain.



Untuk efisiensi, kemudahan akses dan demi menghindari kesalahpahaman terhadap partisipan *prototype*, kami memutuskan untuk mengimplementasikan metode *digital prototyping*. Alasan lain mengapa kami memilih metode ini adalah memudahkan *programmer* untuk membangun tampilan yang serupa, serta saat ini banyak *tools-tools* yang mendukung penggunaan metode *digital prototyping*, *tools-tools* tersebut antara lain: Figma, Adobe XD, Invision Studio, Adobe Photoshop, dan masih banyak *tools-tools* lain yang bahkan didukung oleh AI (*artificial intelligence*).

### Memilih tools prototyping

(Arnowitz et al., 2007) menjelaskan bahwa terdapat 6 langkah dalam memilih *tools* untuk melakukan proses *prototyping*. Langkah pertama adalah memilih metode *prototyping*, seperti yang disebutkan sebelumnya, kami memilih menggunakan metode *digital prototyping*. Langkah kedua adalah memetakan metode *prototyping* dengan *tools* yang sesuai. Berikut adalah hasil pemetaan tersebut. Selanjutnya kita membuat daftar *tools-tools* yang tersedia, dalam hal ini *tools* yang dimiliki, *tools* yang dikuasai, dan apakah *tools* tersebut dapat diakses oleh partisipan lainnya. Berikut pemetaan yang berhasil kami lakukan.

Kemudian, langkah keempat adalah menentukan *deadline* pembuatan *prototype* dan berapa lama *prototype* tersebut dibutuhkan. Selanjutnya menentukan kecocokannya dengan perkiraan waktu yang dibutuhkan dan tinjau ulang dengan karakteristik yang sebelumnya sudah diputuskan. Langkah keenam memutuskan secara *final*, *tools* apa yang paling tepat digunakan. Berikut adalah tabel penentuan *tools* untuk projek kami.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nama *Tools*** | **Prioritas** | **Keahlian Pembuat *Prototype*** | **Waktu yang dibutuhkan** | **Lama *prototype* dibutuhkan** | **Kesesuaian Karakteristik** | **Bisa diakses oleh partisipan lain?** |
| Adobe Photoshop | Rendah | Kurang baik | 50-80 hari | Lama | Cukup sesuai | Tidak |
| Adobe XD | Menengah | Cukup Baik | 40-60 hari | Lama | Sesuai | Bisa |
| Invision Studio | Tinggi | Baik | 35-50 hari | Lama | Sesuai | Bisa |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nama *Tools*** | **Prioritas** | **Keahlian Pembuat *Prototype*** | **Waktu yang dibutuhkan** | **Lama *prototype* dibutuhkan** | **Kesesuaian Karakteristik** | **Bisa diakses oleh partisipan lain?** |
| **Figma** | Sangat Tinggi | Sangat Baik | 30-40 hari | Lama | Sesuai | Bisa |

### Menetapkan kriteria desain

Tabel dibawah ini adalah tabel prioritas kriteria desain (Arnowitz et al., 2007), sebagai panduan dalam melakukan proses desain apabila pelaku/penanggung jawab *prototype* merasa masih kebingungan akan melakukan apa.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Panduan** | **Prioritas** | | | | | **Alasan** |
| **Panduan Desain Visual** | **1** | **2** | **3** | **4** | **5** |  |
| 1. Alur Informasi (Arahan) |  |  |  |  |  |  |
| 2. Susunan berbasis *grid* (Susunan) |  |  |  |  |  |  |
| 3. Ritme dan pola (Arahan) |  |  |  |  |  |  |
| 4. Kesatuan dan keragaman (Susunan) |  |  |  |  |  |  |
| 5. Struktur tipografi (Susunan) |  |  |  |  |  |  |
| 6. Keseimbangan (Arahan) |  |  |  |  |  |  |
| 7. Pengelompokkan logika (Susunan) |  |  |  |  |  |  |
| **Panduan** | **Prioritas** | | | | | **Alasan** |
| **Panduan Tampilan Pengguna** | **1** | **2** | **3** | **4** | **5** |  |
| 1. Penampilan progresif (Arahan) |  |  |  |  |  |  |
| 2. Efisiensi (Arahan) |  |  |  |  |  |  |
| 3. *Fitt’s Law* / Hukum Fitt (Arahan) |  |  |  |  |  |  |
| 4. Mudah dipelajari (Arahan) |  |  |  |  |  |  |
| 5. Berbicara dalam bahasa pengguna (Susunan) |  |  |  |  |  |  |
| 6. Secara eksplisit menampilkan tindakan dan kolom yang wajib diisi (Susunan) |  |  |  |  |  |  |
| 7. Sensitivitas internasional (Susunan) |  |  |  |  |  |  |
| 8. Aksesibilitas universal (Susunan) |  |  |  |  |  |  |
| 9. Pengguna harus merasa di dalam kontrol (Susunan) |  |  |  |  |  |  |
| 10. Meminimalisir beban kognitif (Susunan) |  |  |  |  |  |  |
| 11. Kepuasan (Susunan) |  |  |  |  |  |  |

### Membuat desain

Menurut (Arnowitz et al., 2007) pembuatan dan membangun sebuah desain *prototype* adalah titik temu dari semua hasil dari proses-proses yang sebelumnya kita siapkan. Sebagai contoh, mengidentifikasi tampilan dengan prioritas tinggi, hasil alami dari aktivitas kebutuhan. Kemudian kita bisa menggunakan kebutuhan-kebutuhan yang sudah dirumuskan dan kriteria-kriteria desain dengan menambahkan konten dan pemahaman partisipan untuk selanjutnya ditambahkan pada hasil desain. Semuanya dilakukan menggunakan metode dan *tools* yang dipilih sebelumnya. Pembuatan desain *prototype* dapat dilakukan dengan melalui 5 tahapan.

Tahapan pertama yaitu menentukan tampilan dengan prioritas paling tinggi, dari alur tugas yang sudah dipetakan dan skenario yang sudah direncanakan, menentukan tampilan dengan prioritas paling tinggi–tampilan yang wajib ada agar *prototype* dirasa masuk akal. Kemudian, kita tentukan tampilan yang paling kompleks dan menantang. Selanjutnya, kami membuat daftar yang berisi elemen-elemen yang akan ditambahkan pada tampilan ini. Kemudian, memilih daftar prioritas kriteria desain visual dan kriteria desain tampilan pengguna. Kriteria ini nantinya akan membantu kami dalam memblokir tampilan-tampilan *prototype*.

Tahapan kedua yaitu memblokir area-area yang terpenting. Dengan membuat area kosong di *tools prototyping* yang kita pilih, selanjutnya kita blokir secara kasar tampilan yang merupakan area fungsional logis, contohnya area untuk *branding*, area *header*, dan area menu navigasi. Selanjutnya kita bisa melakukan pemblokiran pada tampilan dengan prioritas tinggi lainnya menggunakan tampilan pertama sebagai acuan atau *template*. Dengan melakukan pemblokiran ini, kita bisa melihat pola desain yang mulai konsisten.

Tahapan ketiga, menyisipkan dan mulai menata elemen-elemen desain serta konten-konten yang lebih detail lagi ke dalam tampilan *prototype*. Pertama-tama kita buat sebuah elemen dengan visual atau fungsi yang paling penting menurut kriteria desain kita. Tampilan area fungsional yang ditujukan sebelumnya pada tahapan kedua bisa dijadikan sebagai pengganti dan perataan *grid* untuk elemen-elemen grafis dari tampilan pengguna yang lebih detail. Ketika kita sudah menambah detail-detail ke tampilan dengan prioritas tertinggi kita, tidak jarang kita akan melakukan sedikit mengutak-atik tata letak elemen-elemen desain nantinya.

Tahapan keempat yakni melakukan penataan elemen tampilan-tampilan dengan prioritas lebih rendah, selain itu, kita juga mengulangi tahapan ketiga untuk melakukan iterasi. Tidak jarang juga para pelaku / penanggung jawab *prototype* bolak-balik melakukan tahapan ketiga kemudian lanjut keempat, di tengah melakukan tahapan keempat, kita kembali lagi melakukan tahapan ketiga dan seterusnya. Biasanya, setelah pelaku / penanggung jawab *prototype* sudah menghasilkan 3 atau 4 atau lebih tampilan proses desain dan iterasi akan menjadi lebih mudah.

Tahapan kelima, sebelum kita melanjutkannya ke pemangku kepentingan lain, kita bisa mencatat alasan-alasan desain berdasarkan tujuan *prototype*: bisnis, fungsional, teknis dan kebutuhan kegunaan yang diaplikasikan *prototype* dan kriteria desain yang di prioritaskan. Dengan mencatat alasan-alasan ini, diyakini kita dapat mencapai tujuan dari desain dan bisa digunakan untuk memenuhi ekspektasi partisipan mengenai keputusan desain yang kita lakukan. Dengan melakukan hal tersebut, ketika ada seseorang menyarankan perubahan terhadap desain *prototype* kita, seperti warna, perlakuan tipografi, tombol-tombol interaksi, tata letak dan faktor-faktor lain, kita bisa mengemukakan alasan desain yang kita adopsi sebagai basis untuk pertimbangan dan negosiasi.

### Meninjau desain

Dalam meninjau ulang desain yang sudah dihasilkan kita bisa melakukan 5 tahapan untuk menjamin kualitas desain kita bersama dengan tim internal pengembangan perangkat lunak. Langkah pertama meninjau target partisipan, siapa yang perlu melihat *prototype*? Siapa yang menggunakan *prototype*? Siapa yang harus membuat keputusan? Langkah kedua menyesuaikan tujuan untuk masing-masing versi, pada tahap apa proses desain saat ini? Apa tingkat ketelitian yang dipilih atau seberapa jauh selesai? Seberapa lengkap konten didalam *prototype*?

Langkah ketiga menyesuaikan ekspektasi untuk tim peninjau, pada tahap apa proses pengembangan lunak saat ini? Seberapa jauh selesai? Kelengkapan konten di dalam *prototype*. Pada langkah ini, apabila kita adalah penanggung jawab *prototype*, kita harus menjelaskan kepada tim peninjau: 1. Proses pembuatan perangkat lunak; 2. Latar belakang *prototype*; 3. Karakteristik *prototype*; 4. Metode yang digunakan dan alasannya; 5. Hasil yang diharapkan dari *prototype*. Langkah keempat adalah cara kita mempresentasikan hasil pembuatan dan tinjauan yang sudah dilakukan, dengan menunjuk fasilitator terbaik, menyusun agenda, mempersiapkan partisipan, dan mengelola presentasi. Dan yang terakhir, merencanakan langkah selanjutnya, dari rangkaian proses peninjauan ini, apa langkah selanjutnya? Apakah melakukan tinjauan desain sekali lagi? Memvalidasi kegunaan perangkat lunak? Mengiterasi desain? Atau lanjut ke proses pemrograman?

### Memvalidasi desain

Jika tahapan sebelumnya kita melakukan peninjauan bersama dengan tim internal pengembangan perangkat lunak, maka, pada tahapan ini kita akan memvalidasi keputusan tersebut bersama pemangku-pemangku kepentingan diluar tim, seperti pelanggan, pengguna akhir, penanam modal, atau pemangku kepentingan lain yang merupakan target dari perangkat lunak ini. Salah satu cara untuk memvalidasi desain *prototype* adalah dengan melakukan pengujian kegunaan (*usability testing*). Langkah-langkah dalam melakukan pengujian kegunaan adalah:

1. Menyusun rencana pengujian
2. Menyiapkan kuesioner penyaring (definisi akurat dari pengguna) untuk mengkualifikasi partisipan pengujian
3. Menyusun daftar kontak partisipan pengujian yang potensial, termasuk partisipan uji coba
4. Menunjuk seorang perekrut partisipan pengujian internal atau sewa dari luar tim
5. Siapkan ruangan atau ruang lab selama durasi validasi dilakukan
6. Pastikan perlengkapan-perlengkapan untuk melaksanakan setiap sesi sudah disiapkan (alat untuk mencatat, perlengkapan untuk merekam audio / video, dan lain lain)
7. Menyiapkan jadwal sesi validasi–dimana partisipan bisa hadir pada hari apa dan jam berapa?
8. Membuat panduan bagi partisipan untuk memberikan pengguna konteks penggunaan dan tujuan-tujuan lain; termasuk data-data yang mungkin dapat dijadikan petunjuk bagi pengguna atau data yang mungkin harus di-*entry*-kan pengguna
9. Mempersiapkan stimulus pengujian–baik berupa *prototype* atau program perangkat lunak yang akan divalidasi
10. Melakukan gladi bersih bersama dengan partisipan yang bersedia
11. Melakukan sesi validasi yang normal-normal saja, tidak terlalu banyak, juga jangan tidak melakukannya sama sekali
12. Menggunakan catatan dan rekaman video yang diambil di setiap sesi, tinjau ulang dan analisa validasi data. Menafsirkan dan menyatukan keberhasilan-keberhasilan desain sebagai rekomendasi untuk meningkatkan kualitas desain
13. Menyiapkan sebuah presentasi dari hasil validasi dan keberhasilan-keberhasilan dan peningkatan kualitas desain

### Men-*deploy* desain

Langkah terakhir dari metode *prototyping* yaitu membawa *prototype* yang sudah didesain kemudian menerjemahkannya menjadi produk perangkat lunak yang nyata atau melakukan iterasi lagi. Sekarang kita dihadapkan oleh 3 tahapan terakhir dalam men-*deploy prototype* kita. Tahapan pertama yaitu menetapkan ekspektasi pada proses *handoff*. Proses *handoff* dan pekerjaan setelah pemrograman harus dilakukan secara kolaboratif dan dibutuhkan komunikasi terus menerus dengan pembuat *prototype*, tanpa memperhatikan apakah mereka dari tim internal atau konsultan. Kita menyaksikan banyak desain berakhir sia-sia ketika seorang atau tim *programmer* yang dipekerjakan malah tidak memedulikan *prototype* yang diberikan.

Tahapan kedua menyampaikan strategi *prototype*, menurut (Arnowitz et al., 2007), ada berbagai macam strategi dalam melakukan proses *handoff* *prototype*. Strategi *closed versus open* adalah strategi paling populer dan paling dikenal. Strategi *closed* maksudnya adalah *prototype* beserta dokumen yang melengkapi bersifat beku hanya dapat dilihat dan tidak dapat diubah. Strategi *open* maksudnya *prototype* beserta dokumen yang menyertainya bersifat terbuka terhadap semua perubahan yang mungkin muncul atau disarankan oleh *programmer* pada proses pemrograman.

Strategi *Narrative versus Interactive Format*, format naratif sama dengan *storyboard* ditujukan untuk mengenalkan dan menguji ide-ide baru yang memungkinkan kita untuk menjelaskan semua langkah-langkah dalam mencapai tujuan *prototype* menggunakan istilah yang mudah dimengerti. Sedangkan format interaktif, pada dasarnya adalah dokumentasi perilaku dan interaksi sistem dengan akurat.

Tahapan ketiga yaitu dokumentasi hasil *prototyping*, jika sebuah *prototype* sudah didesain dengan baik tapi tidak digunakan, apakah *prototype* tersebut dianggap ada? Mengingat sifatnya yang sementara, dokumentasi hasil proses *prototype* sangat penting apalagi jika akhirnya akan diadopsi ke proses perancangan perangkat lunak, khususnya tahapan implementasi/pemrograman. Untuk mendukung proses *handoff* ini dan memungkinkan transisi penyampaian dan implementasi yang lancar, sebuah dokumentasi resmi dan penyimpanan kebutuhan harus dimiliki oleh *prototype*. Penting rasanya untuk memberikan *prototype* beserta dokumen-dokumen yang menyertainya untuk memiliki kemudahan akses, baik melalui server pusat, sistem manajemen konten, atau didalam dokumen seperti panduan desain produk, panduan gaya, dan standar-standar dan panduan-panduan lain. Dokumentasi *prototype* layaknya bisa disisipkan ke dokumen-dokumen yang sudah digunakan oleh kebanyakan orang saat ini. Membuat dokumen-dokumen baru atau merevisi dokumen tersebut akan mengakibatkan dokumen yang kita tulis tidak akan pernah digunakan.

# 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.

Google. (2023). Bard (versi Desember 2023) [Model bahasa besar]

Google. (2024). Gemini (versi Maret 2024) [Model bahasa besar]

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/yjijjlloibh45hhxsit6vblfey/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)

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 kantoran 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>

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%20statement>

<https://dephub.go.id/post/read/%E2%80%8Btekan-angka-kecelakaan-lalu-lintas,-kemenhub-ajak-masyarakat-beralih-ke-transportasi-umum-dan-utamakan-keselamatan-berkendara>

Gambar grafik di bab 1: <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.kominfo.go.id%2Findex.php%2Fcontent%2Fdetail%2F10368%2Frata-rata-tiga-orang-meninggal-setiap-jam-akibat-kecelakaan-jalan%2F0%2Fartikel_gpr&psig=AOvVaw0Tutn4paJMxpPrPPvsjsOT&ust=1710032063225000&source=images&cd=vfe&opi=89978449&ved=0CBEQjRxqFwoTCJCvjYb85YQDFQAAAAAdAAAAABAD>