BAB IV

RANCANGAN SISTEM DAN PROGRAM USULAN

4.1. Analisa Kebutuhan Software

Sistem pemesanan print tekstil merupakan aplikasi berbasis *web* yang dikembangkan untuk mendukung digitalisasi proses bisnis layanan cetak dan pemesanan properti secara terintegrasi. Aplikasi ini hadir sebagai alternatif dari cara pencatatan manual yang kerap menimbulkan risiko kesalahan data, kurang efisien, dan menyulitkan dalam pemantauan riwayat pemesanan maupun status produksi.

Melalui implementasi sistem ini, seluruh tahapan operasional mulai dari pendaftaran pelanggan, pemilihan jenis kain dan motif, penjadwalan, penghitungan total biaya, hingga pemantauan progres pesanan dapat dikelola secara *real-time*. Petugas dapat melakukan pencatatan transaksi dan memperbarui status pesanan, sedangkan administrator memiliki kewenangan lebih luas untuk memantau aktivitas sistem, mengelola data layanan cetak, pengguna, serta menyusun laporan pemesanan.

Pada sisi *backend*, sistem ini dibangun menggunakan *Express.js*, yaitu kerangka kerja minimalis dan fleksibel berbasis Node.js yang mendukung pengembangan RESTful API dengan performa tinggi. Express dipilih karena kemudahan dalam pengelolaan *routing*, *middleware*, serta integrasi ke berbagai basis data modern. Sementara pada sisi *frontend*, aplikasi memanfaatkan Nuxt.js, framework Vue.js yang mendukung rendering sisi server (*server-side rendering*) sehingga menghasilkan antarmuka yang cepat dan optimal di berbagai perangkat.

Untuk mengatur alur proses pemesanan dan transisi status secara terstruktur, diterapkan pendekatan *Finite State Machine* (FSM). Dengan model ini, setiap pesanan akan melalui sejumlah keadaan (*state*) yang terdefinisi, misalnya Pesanan Diterima, Proses Cetak, Selesai, dan Diambil, sehingga pergerakan data lebih terkendali dan transparan.

Aplikasi ini dapat diakses melalui berbagai peramban *web* seperti Google Chrome, Mozilla Firefox, atau lainnya. Antarmuka pengguna dirancang agar intuitif, responsif, dan mudah digunakan oleh operator toko dengan latar belakang teknis yang beragam.

4.1.1. Kebutuhan Fungsional Sistem:

a. **Pencatatan dan Pemantauan Transaksi**: Sistem harus mampu mencatat dan memantau setiap tahapan pesanan transaksi, mulai dari pemesanan awal, konfirmasi pembayaran, proses produksi, hingga pengiriman atau pengambilan. Ini mencakup riwayat transaksi lengkap untuk memudahkan pemantauan dan audit.
b. **Manajemen Data Produk dan Harga**: Mengelola katalog produk transaksi (jenis kain, teknik cetak, ukuran, dll.) dan struktur harga yang fleksibel, termasuk diskon atau promosi. Informasi harga harus selalu terkini dan transparan.
c. ***Dashboard* Pemantauan**: Menyediakan *dashboard* interaktif yang menampilkan statistik pesanan, status produksi, ketersediaan bahan baku (jika relevan), dan insight penjualan secara *real-time* untuk membantu manajemen dan staf dalam pengambilan keputusan.
d. **Pengelolaan Data Pengguna**: Fitur manajemen pengguna yang komprehensif untuk mengelola data pelanggan dan admin, dengan hak akses yang berbeda sesuai peran masing-masing.
e. **Laporan dan Dokumentasi**: Menghasilkan laporan transaksi, pesanan, dan dokumen terkait lainnya secara otomatis untuk kebutuhan administrasi, pelaporan keuangan, dan analisis bisnis.
f. **Fitur Pemesanan *Online***: Memungkinkan pelanggan untuk melakukan pemesanan transaksi secara online melalui *website*, memilih spesifikasi produk, mengunggah desain, dan menyelesaikan proses *checkout* dengan mudah.

4.1.2. Kebutuhan Non-Fungsional

Untuk Sistem Pemesanan transaksi Mochi Studio, kebutuhan non-fungsional sangat krusial untuk memastikan pengalaman pengguna yang optimal dan operasional yang efisien:

a. **Aksesibilitas Berbasis Website**: Sistem ini akan dikembangkan sebagai aplikasi *web*, memungkinkan aksesibilitas kapan saja dan di mana saja melalui perangkat apa pun yang terhubung internet. Ini menjamin fleksibilitas operasional bagi Mochi Studio dan kemudahan bagi pelanggan.
b. **Keamanan Data dan Transaksi**: Keamanan adalah prioritas utama. Sistem akan menerapkan teknologi autentikasi yang kuat untuk melindungi data pelanggan, detail pesanan, dan transaksi. Ini termasuk enkripsi data, otorisasi peran pengguna (admin vs. pelanggan), dan perlindungan terhadap akses tidak sah untuk mencegah penyalahgunaan.
c. **Kemudahan Penggunaan** (*User-Friendly*): Antarmuka pengguna (UI) harus intuitif dan responsif, dirancang agar mudah digunakan oleh staf Mochi Studio maupun pelanggan tanpa memerlukan pelatihan teknis yang rumit. Tujuannya adalah meminimalkan hambatan adaptasi dan memaksimalkan efisiensi.
d. **Skalabilitas**: Sistem harus dirancang untuk dapat berkembang seiring dengan pertumbuhan Mochi Studio. Ini berarti sistem mampu menangani peningkatan volume pesanan, jumlah pelanggan, dan penambahan fitur baru di masa mendatang tanpa mengorbankan kinerja.
e. ***Responsive Design*** **:** Aplikasi dirancang dengan kemampuan responsif di berbagai ukuran layar sehingga membuat pelanggan mudah beradaptasi dalam aplikasi baik dengan gawai yang kecil maupun ukuran besar.
f. ***Database*** **:** Aplikasi menyimpan *database* dengan keamanan tinggi menggunakan jenis *database* MySQL

4.2. Desain

4.2.1 Desain Pemodelan Sistem

Sistem Pemesanan transaksi untuk Mochi Studio berbasis *website* dengan teknologi Node.js akan dirancang dengan pendekatan modern, mengadopsi praktik terbaik dalam pengembangan aplikasi *web* dan disesuaikan dengan alur bisnis transaksi.

1. Model Konseptual dan Arsitektur Sistem

a. **Arsitektur Berbasis *Web* dengan Node.js sebagai *Backend***: Node.js akan menjadi fondasi *backend*, menangani logika bisnis, API, dan pengelolaan database secara efisien dan *scalable*. Ini ideal untuk mendukung pemrosesan pesanan *real-time* dan pelacakan status melalui *Finite State Machine*.

b. ***Frontend* Berbasis Web Responsif:** Antarmuka pengguna akan dibangun menggunakan *framework* JavaScript modern NuxtJs. Ini akan memastikan sistem mudah diakses dari berbagai perangkat (desktop, tablet, smartphone) dengan tampilan yang adaptif dan pengalaman pengguna yang lancar.

c. ***Database* Relasional**: MySQL atau PostgreSQL akan digunakan untuk menyimpan data secara terstruktur dan terintegrasi. Ini mencakup data pengguna (pelanggan dan admin), katalog produk (jenis kain, metode cetak, ukuran, dll.), detail pesanan, status transaksi, dan riwayat produksi. Struktur database akan dirancang dengan *Physical Data Model* yang valid untuk memastikan integritas dan efisiensi *query*.

d. **RESTful API**: Komunikasi antara *frontend* dan *backend* akan diatur melalui RESTful API. Ini memungkinkan pertukaran data yang efisien dan modular, memudahkan pengembangan dan integrasi di masa depan.

e. **Keamanan Autentikasi dan Otorisasi**: Implementasi JSON Web Tokens (JWT) atau mekanisme autentikasi serupa akan digunakan untuk mengelola akses pengguna. Ini memastikan bahwa hanya

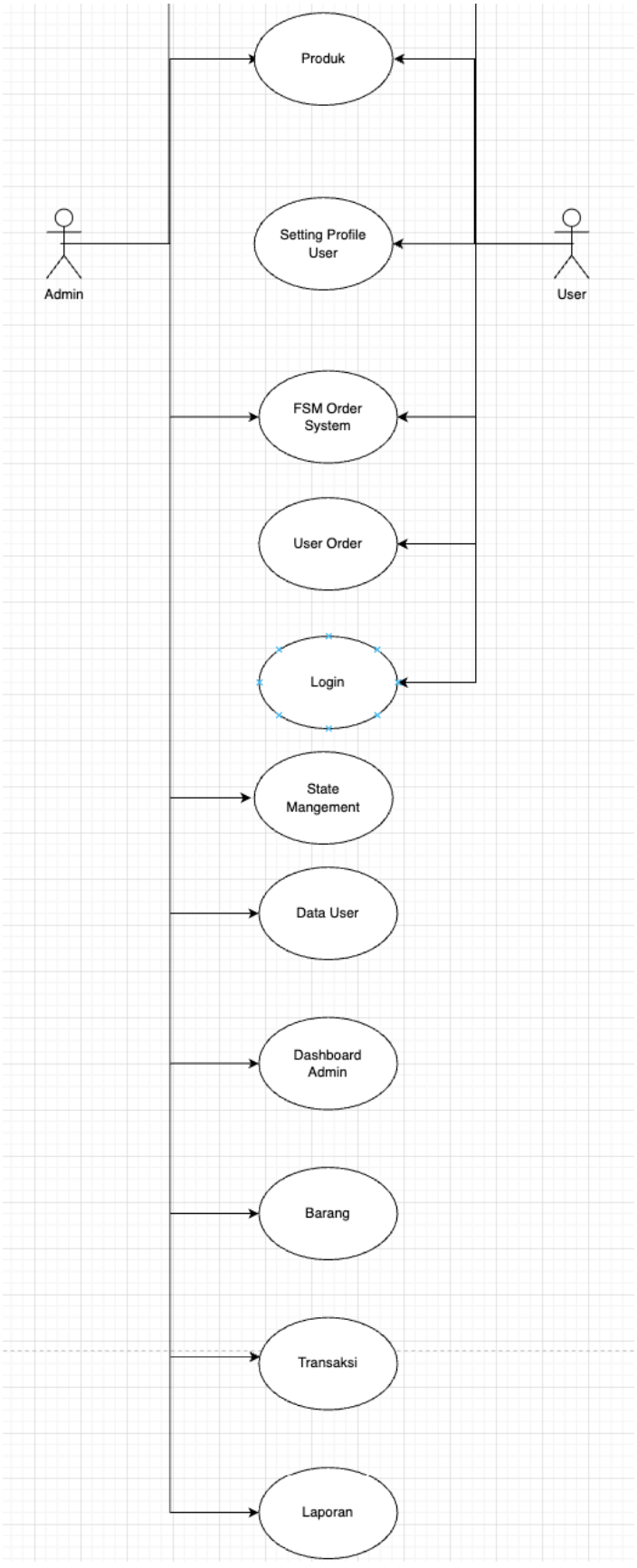
pengguna terdaftar (pelanggan) dan admin yang berwenang yang dapat mengakses fitur-fitur yang relevan.

2. Diagram Pemodelan Sistem (UML)

Desain sistem akan divisualisasikan menggunakan diagram UML (*Unified Modeling Language*) untuk memberikan gambaran yang jelas tentang fungsionalitas dan struktur:

a. *Use Case Diagram*:

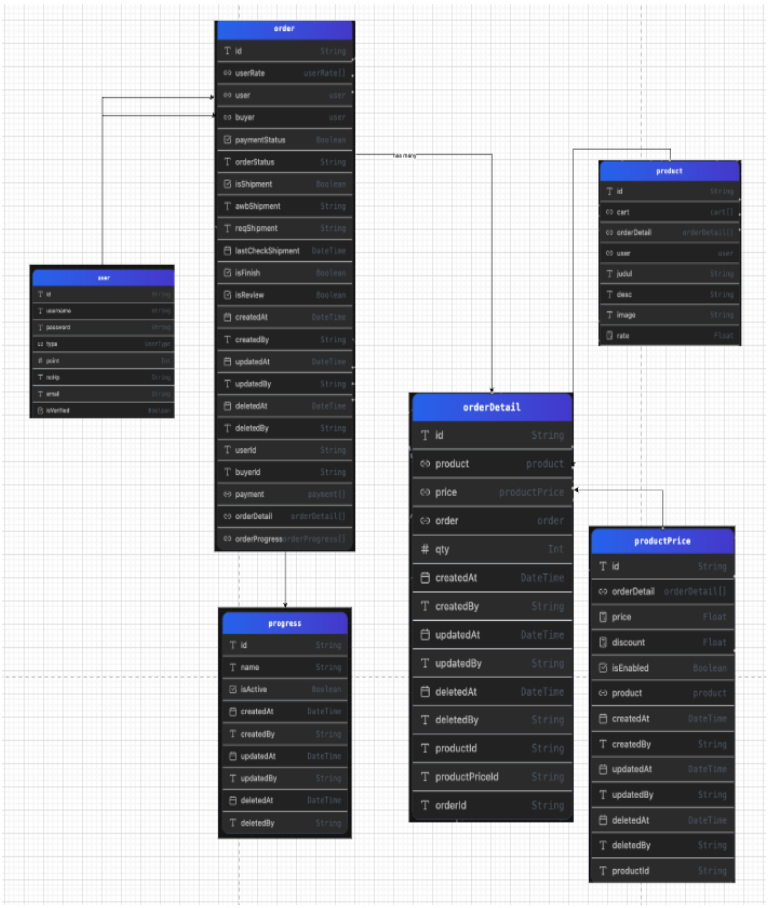
Use Case Diagram adalah gambaran interaksi dari pengguna sistem terhadap fitur yang tersedia. Dalam hal ini sistem yang dibangun mempunyai fungsi: Login, Manajemen Profil Pengguna, Pemesanan Barang (termasuk unggah desain, pemilihan spesifikasi), Pelacakan Status Pesanan, Konfirmasi Pembayaran, Pengelolaan Data Produk, Manajemen Pesanan, Pembaruan Status Produksi, dan Pelaporan.



Gambar 4. 1 Use Case Diagram

b. Class Diagram

Class Diagram adalah alat untuk membantu memvisualisasikan atau menggambarkan struktur kelas yang terdapat pada suatu sistem program. *Class* menggambarkan kelompok suatu objek bersama properti,operasi dan relasi yang sama[8].



Gambar 4. 2 Class Diagram

3. Komponen dan Modul

Sistem Pemesanan transaksi Mochi Studio akan memiliki beberapa modul utama yang bekerja sama untuk mempermudah seluruh proses, baik bagi pelanggan maupun tim internal.

a. **Manajemen Pengguna**. Modul ini memungkinkan pelanggan untuk mendaftar, masuk ke akun mereka, dan mengelola informasi profil pribadi. Admin juga bisa mengatur profil mereka sendiri. Yang

terpenting, modul ini memastikan setiap orang hanya memiliki akses ke fitur yang sesuai dengan peran mereka; pelanggan dapat memesan dan melihat riwayat pesanan, sementara admin bisa mengelola semua data dan transaksi.

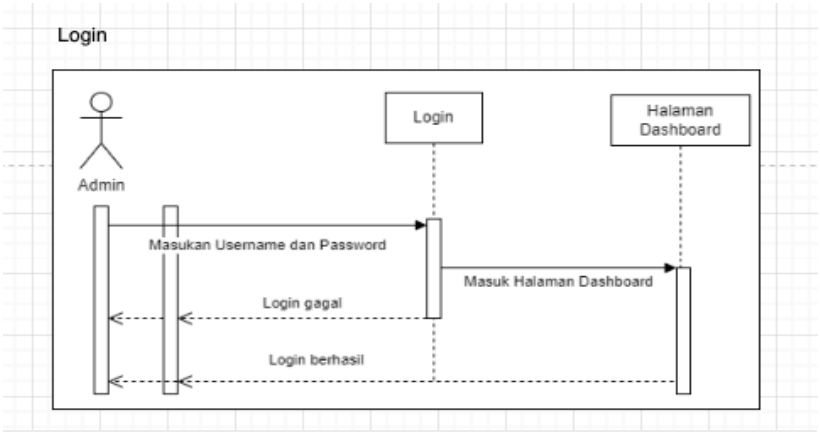
b. **Manajemen Produk** akan menangani semua informasi tentang layanan transaksi Mochi Studio. Ini termasuk memasukkan dan memperbarui detail seperti jenis kain yang tersedia, teknik cetak, pilihan ukuran, dan struktur harga yang fleksibel. Modul ini memastikan Mochi Studio dapat dengan mudah menambahkan atau mengubah spesifikasi produk sesuai kebutuhan pasar.
c. **Pemesanan**. Di sini, pelanggan dapat melakukan pemesanan secara online, mengunggah desain mereka, memilih spesifikasi produk yang diinginkan, dan melihat perkiraan biaya. Sistem secara otomatis akan memvalidasi kelengkapan pesanan sebelum diserahkan.
d. **Pelacakan Pesanan** (FSM). Modul ini memungkinkan pelacakan status pesanan secara real-time menggunakan konsep *Finite State Machine*. Bayangkan status pesanan bergerak dari "Menunggu Pembayaran", lalu ke "Desain Diverifikasi", kemudian "Dalam Produksi", hingga "Siap Kirim/Ambil", dan akhirnya "Selesai". Admin akan memiliki kemampuan untuk memperbarui status ini sesuai dengan kemajuan pesanan.
e. **Pengelolaan Transaksi**. Modul ini memungkinkan admin untuk mengelola semua aspek pesanan, mulai dari mengonfirmasi pembayaran, memperbarui status produksi, hingga menangani masalah yang mungkin timbul. Admin juga bisa memberikan otorisasi atau bahkan membatalkan pesanan jika diperlukan.

Untuk membantu pengambilan keputusan, ada modul Laporan & Statistik. Ini akan menghasilkan berbagai laporan digital, seperti laporan penjualan, daftar pesanan, dan statistik produksi. Data ini sangat berharga bagi Mochi Studio untuk menganalisis kinerja bisnis mereka.

4. *Sequence Diagram*

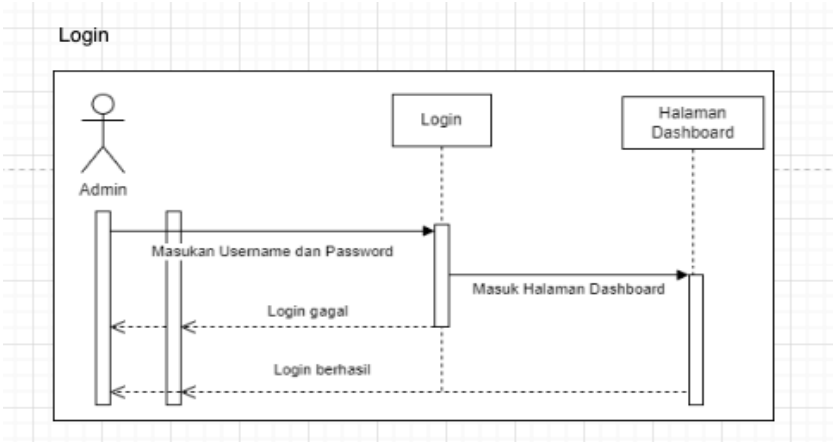
Sequence Diagram digunakan untuk menggambarkan alur komunikasi antar objek pada satu use case tertentu secara berurutan. Diagram ini menunjukkan bagaimana alur kerja dari *login* dimulai dari user hingga sistem melakukan autentikasi dan memberikan respon, baik sukses maupun gagal.

a. *Sequence Diagram* Pendaftaran



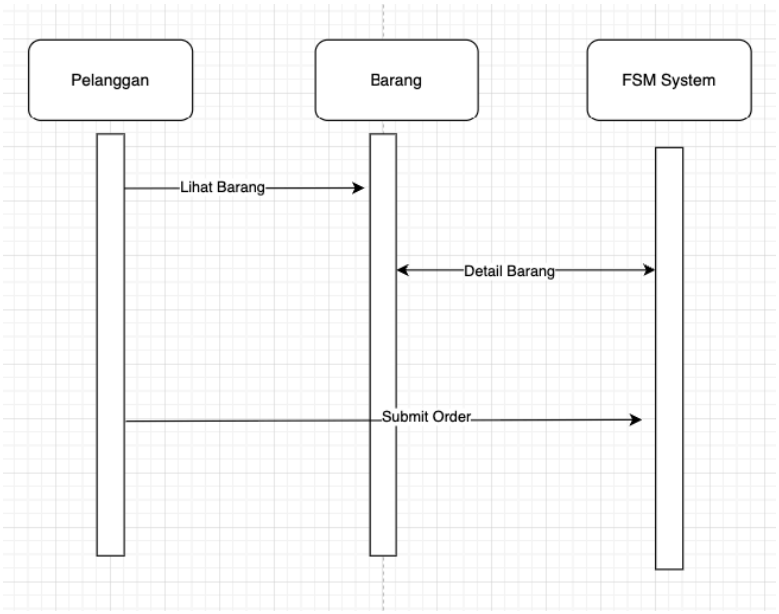
Gambar 4. 3 Sequence Diagram Pendaftaran

b. Sequence Diagram Login



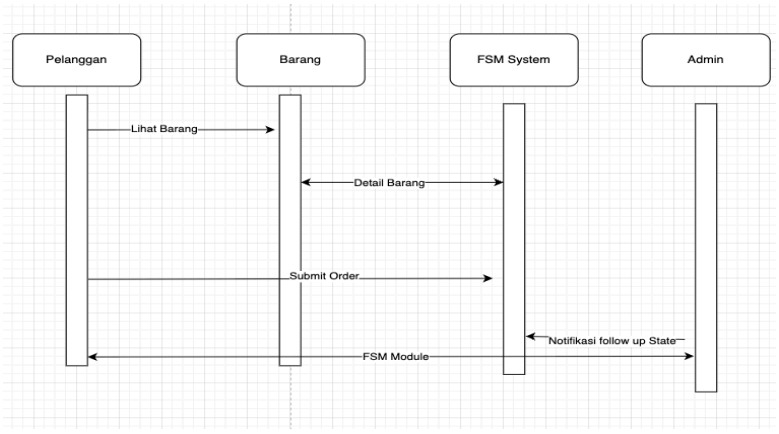
Gambar 4. 4 Sequence Diagram Login

c. *Sequence Diagram Checkout*



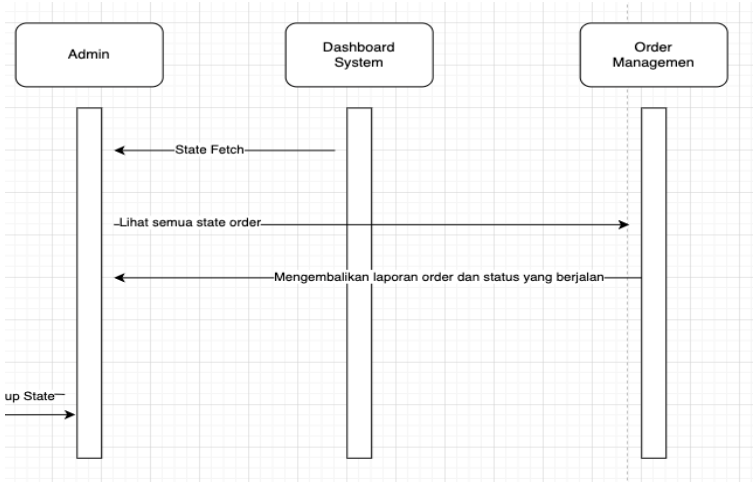
Gambar 4. 5 Sequence Diagram Checkout

d. *Sequence Diagram* Pemesanan dengan FSM Model



Gambar 4. 6 Sequence Diagram Pemesanan

e. *Sequence Diagram* Laporan dan Pelacakan Pesanan



Gambar 4. 7 Sequence Diagram Laporan Pesanan

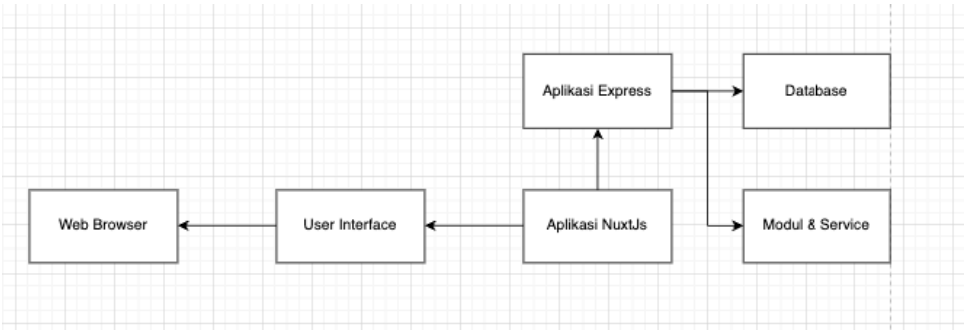
5. Teknologi Pendukung

Untuk memastikan kinerja optimal dan skalabilitas, sistem ini akan dibangun dengan teknologi modern dan terbukti:

a. ***Backend***: Node.js dengan *framework* Express.js untuk membangun API yang efisien dan scalable dalam menangani logika bisnis dan interaksi database.
b. ***Database***: MySQL atau PostgreSQL akan digunakan sebagai *database* relasional untuk menyimpan data pengguna, produk, pesanan, dan status transaksi secara terstruktur.
c. ***Frontend***: Nuxt.js (berbasis Vue.js) akan digunakan untuk membangun antarmuka pengguna yang responsif dan performa tinggi, mendukung *Server-Side Rendering* (SSR) untuk SEO dan pengalaman pengguna yang lebih baik.
d. **Autentikasi**: JSON Web Tokens (JWT) akan diimplementasikan untuk mengelola sesi pengguna dan mengamankan akses ke berbagai modul sistem.
e. ***Deployment***: Sistem akan diluncurkan pada layanan *cloud hosting* (misal: AWS, DigitalOcean, Heroku) untuk memastikan aksesibilitas luas dan ketersediaan tinggi dan bagian *front end* akan diluncurkan dengan layanan netlify.

6. *Component Diagram*

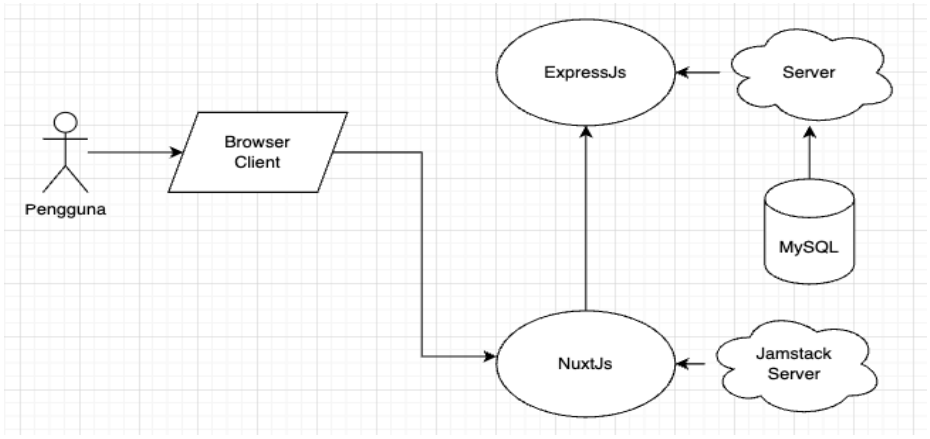
Component diagram menggambarkan bagaimana perangkat lunak berinteraksi untuk membangun sebuah sistem yang nantinya digunakan oleh pengguna. Berikut gambaran bagaimana sistem pemesanan ini dibangun :



Gambar 4. 8 Component Diagram

7. *Deployment Diagram*

Dalam *deployment* diagram digambarkan bagaimana nanti sistem yang telah dikembangkan dijalankan dalam konfigurasi fisik baik secara lokal maupun *live*. Dalam diagram ada beberapa node yang saling terhubung antara aplikasi pada sisi server dan aplikasi yang diakses pelanggan. Berikut gambaran dari deployment yang dilakukan oleh sistem :



Gambar 4. 9 Deployment Diagram

4.2.2 Desain Pemodelan Data

Desain pemodelan data untuk aplikasi pemesanan produk kustom Mochi Studio berbasis website menggunakan Node.js secara umum melibatkan beberapa komponen utama. Komponen-komponen ini dirancang untuk merefleksikan proses bisnis inti Mochi Studio, meliputi manajemen produk kustom, data pelanggan, transaksi pemesanan, dan pengelolaan progres pengerjaan produk.

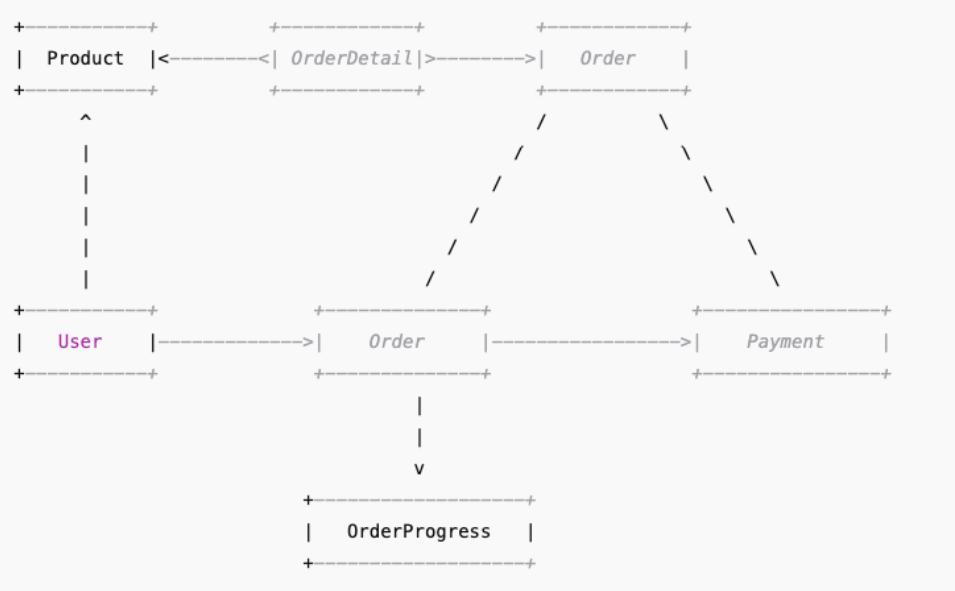
Desain ini mengadaptasi praktik terbaik dalam pengembangan sistem berbasis website, dengan penyesuaian pada teknologi Node.js sebagai backend modern dan penggunaan Prisma ORM untuk interaksi database yang efisien dan terstruktur.

1. Entitas dan Relasi Utama

Berdasarkan skema Prisma yang saya gunakan untuk pengembangan sistem ini, berikut adalah entitas utama beserta relasinya:

a. Product (1) — (N) OrderDetail: Satu jenis produk kustom Mochi Studio (misalnya, desain mug tertentu, pola kaos) dapat disertakan dalam banyak detail transaksi pemesanan. Relasi ini menunjukkan bahwa satu item produk master bisa dipesan berkali-kali oleh pelanggan yang berbeda.
b. User (Pelanggan) (1) — (N) Order: Satu pelanggan dapat melakukan banyak transaksi pemesanan. Entitas User di sini berperan sebagai pembeli (*buyer*).
c. Order (1) — (N) OrderDetail: Satu transaksi pemesanan (Order) dapat terdiri dari banyak item produk Mochi Studio yang berbeda (OrderDetail), misalnya satu pesanan bisa mencakup mug kustom dan kaos kustom.
d. Order (1) — (N) Payment: Satu order dapat memiliki banyak *record* pembayaran. Ini mengakomodasi skenario seperti pembayaran uang muka (DP) dan pelunasan, atau riwayat upaya pembayaran yang gagal.

e. Order (1) — (N) OrderProgress: Setiap order akan memiliki banyak *record progress* yang secara spesifik menunjukkan tahapan pengerjaan kustom yang telah atau sedang dijalani. Misalnya, untuk satu order mug kustom, akan ada *progress* "Desain Disetujui", "Proses Cetak Dimulai", dan seterusnya.



Gambar 4. 10 Ilustrasi Logika *Entitas*

2. Teknologi dan Pendekatan

a. Backend: Saya menggunakan Node.js dengan *framework* Express.js untuk membangun API. Ini memungkinkan sistem untuk berinteraksi secara efisien antara frontend dan database.
b. Database: Saya memanfaatkan database relasional MySQL/PostgreSQL yang diakses melalui Prisma ORM. Prisma dipilih karena kemampuannya dalam mengelola skema database secara type-safe dan mempermudah operasi CRUD (Create, Read, Update, Delete).
c. Desain Sistem: Saya menerapkan metode *Waterfall* untuk pendekatan pengembangan sistem. Pemodelan dilakukan dengan UML (Unified Modeling Language), mencakup Use Case Diagram untuk mendefinisikan fungsionalitas sistem dari sudut pandang pengguna, Class Diagram untuk memvisualisasikan struktur data

dan relasi antar entitas, serta *Sequence Diagram* untuk menggambarkan alur interaksi antarkomponen sistem secara jelas.

d. Fitur Tambahan: Sistem ini juga akan dilengkapi dengan fitur notifikasi (misalnya melalui email atau WhatsApp *Gateway*) untuk memberitahukan status order atau progres pengerjaan kepada pelanggan.

3. Contoh Struktur Koleksi/Tabel (Prisma/SQL)

Berikut adalah representasi struktur tabel atau koleksi utama dalam database, disesuaikan dari skema Prisma yang saya gunakan untuk Mochi Studio:

```
// Model Product (Produk Mochi Studio)
{
  "id": "String (UUID)",
  "judul": "String",
  "desc": "String",
  "image": "String (URL gambar)",
  "qty": "Int (Stok)",
  "type": "String (Kategori produk, misal: 'Mug Custom', 'Kaos Custom')",
  "isPo": "Boolean (Indikator Pre-Order)",
  "process": "String (Deskripsi alur pengerjaan kustom)",
  "userId": "String (FK ke User - pemilik produk)",
  "createdAt": "DateTime",
  "updatedAt": "DateTime"
}
```

```
// Model User (Pelanggan/Admin)
{
  "id": "String (UUID)",
  "username": "String (Unique)",
  "email": "String (Unique)",
  "password": "String",
  "noHp": "String (Nomor Telepon)",
  "displayName": "String",
  "type": "Enum (UserType - User/Admin)",
  "createdAt": "DateTime",
  "updatedAt": "DateTime"
}

// Model Order (Pemesanan)
{
  "id": "String (UUID)",
  "userId": "String (FK ke User - pemilik produk/Mochi Studio)",
  "buyerId": "String (FK ke User - pelanggan)",
  "paymentStatus": "Boolean (True jika sudah dibayar penuh)",
  "orderStatus": "String (Dikendalikan FSM: PENDING, PAID, PROCESSING, SHIPPED, etc.)",
  "isShipment": "Boolean (Indikator siap kirim)",
  "awbShipment": "String (Nomor Resi Pengiriman)",
  "isFinish": "Boolean (Indikator order selesai)",
```

```
  "isReview": "Boolean (Indikator sudah di-review)",
  "createdAt": "DateTime",
  "updatedAt": "DateTime"
}

// Model OrderDetail (Detail Item Pemesanan)
{
  "id": "String (UUID)",
  "orderId": "String (FK ke Order)",
  "productId": "String (FK ke Product)",
  "productPriceId": "String (FK ke ProductPrice - harga spesifik yang dipilih)",
  "qty": "Int (Jumlah produk)",
  "createdAt": "DateTime",
  "updatedAt": "DateTime"
}

// Model Progress (Master Data Tahapan Pengerjaan Kustom)
{
  "id": "String (UUID)",
  "name": "String (Nama tahapan, misal: 'Desain Disetujui', 'Produksi Dimulai')",
  "isActive": "Boolean",
  "createdAt": "DateTime",
  "updatedAt": "DateTime"
```

```
}

// Model OrderProgress (Progres Pengerjaan Spesifik per Order)
{
  "id": "String (UUID)",
  "orderId": "String (FK ke Order)",
  "progressId": "String (FK ke Progress)",
  "status": "String (Status tahapan ini, misal: 'completed', 'pending')",
  "timestamp": "DateTime (Waktu update progres)",
  "createdAt": "DateTime",
  "updatedAt": "DateTime"
}
```

4.Alur Sistem

Alur sistem pemesanan Mochi Studio berpusat pada pengelolaan status order yang dinamis, tanpa adanya pelacakan kiriman secara eksternal melainkan hanya pencatatan AWB.

a. Melakukan Pemesanan: Pelanggan memilih produk Mochi Studio, termasuk opsi kustomisasi jika ada, dan menyelesaikan proses pemesanan melalui website. Sistem kemudian mencatat order baru dengan status awal (orderStatus) "Pending" dan detail produk (OrderDetail).
b. Pembayaran: Setelah pemesanan, pelanggan melakukan pembayaran. Sistem akan memverifikasi pembayaran dan memperbarui paymentStatus pada order terkait. Jika pembayaran berhasil, orderStatus akan bertransisi ke "Paid".
c. Proses Pengerjaan (Mochi Studio): Setelah order berstatus "Paid", tim Mochi Studio akan memulai proses pengerjaan

produk kustom. Setiap tahapan pengerjaan (misalnya, persetujuan desain, produksi, quality control) akan dicatat secara detail dalam OrderProgress, memberikan visibilitas kepada pelanggan mengenai kemajuan order mereka. orderStatus utama akan berada di "Processing" selama tahapan ini.

d. Penyelesaian Pengerjaan & Siap Kirim: Setelah semua tahapan pengerjaan selesai, orderStatus akan bertransisi ke "Ready for Shipment".
e. Pengiriman: Tim Mochi Studio akan menyiapkan produk untuk pengiriman. Nomor resi pengiriman (*airwaybill*) akan dicatat pada entitas Order, dan orderStatus akan diperbarui menjadi "Shipped".
f. Konfirmasi Penerimaan: Setelah produk diterima oleh pelanggan, mereka dapat melakukan konfirmasi penerimaan di sistem. orderStatus kemudian akan berubah menjadi "Delivered".
g. Penyelesaian Order: Setelah produk diterima dan mungkin review diberikan oleh pelanggan, orderStatus akan bertransisi menjadi "Completed", menandakan berakhirnya siklus order.
h. Admin Mengelola Operasi: Admin Mochi Studio memiliki peran sentral dalam mengelola data produk (product), memantau dan memperbarui status setiap order (orderStatus), mengelola progres pengerjaan (OrderProgress), dan mengelola data pelanggan (User). Sistem juga akan menyediakan laporan penjualan yang relevan untuk analisis bisnis.

Desain ini memastikan setiap langkah dalam proses pemesanan Mochi Studio terekam dengan jelas, dari awal hingga akhir, dengan *feedback* yang relevan kepada pelanggan dan alat manajemen yang komprehensif untuk Mochi Studio.

4.2.3 Desain *User Interface*

Dalam perancangan sistem pemesanan untuk Mochi Studio, saya mengutamakan desain *User Interface* (UI) yang mampu menyediakan kemudahan penggunaan, kejelasan informasi, dan responsivitas. Tujuan utamanya adalah agar pelanggan dapat melakukan pemesanan produk kustom Mochi Studio dengan cepat, nyaman, dan terpandu dengan baik. Mengingat karakteristik produk Mochi Studio yang memungkinkan personalisasi dan sistem pre-order, UI yang dirancang harus mampu memfasilitasi interaksi yang dinamis dan memberikan *feedback progress* yang transparan kepada pengguna.

1. Pendekatan Desain

Saya mengadopsi dua pendekatan utama dalam perancangan UI sistem ini:

a. **Metode *Design Thinking***: Saya menerapkan tahapan *Design Thinking* yang berorientasi pada pengguna, meliputi *Empathize, Define, Ideate, Prototype,* dan *Testing*. Pendekatan ini esensial untuk memastikan desain UI benar-benar memahami kebutuhan unik pelanggan Mochi Studio, terutama terkait proses personalisasi produk dan alur pre-order yang mungkin kompleks. Melalui tahapan ini, saya dapat mendefinisikan masalah pengguna secara akurat, menghasilkan ide-ide solusi yang kreatif, membangun prototipe untuk visualisasi, dan melakukan pengujian langsung dengan calon pengguna untuk memvalidasi efektivitas dan kemudahan penggunaan desain.

b. ***Human-Centered Design* (HCD)**: Fokus saya pada *Human-Centered Design* (HCD) bertujuan untuk menciptakan tampilan yang intuitif dan mempermudah pelanggan di setiap langkah proses pemesanan. Mulai dari pemilihan produk, konfigurasi kustomisasi, hingga pelacakan progres pengerjaan dan pengiriman, setiap interaksi dirancang agar sealami dan seefisien mungkin bagi pengguna.

2. Komponen Utama UI

Berikut adalah deskripsi komponen UI utama yang saya rancang untuk *website* pemesanan Mochi Studio, disesuaikan dengan alur bisnis dan struktur data yang telah dibahas:

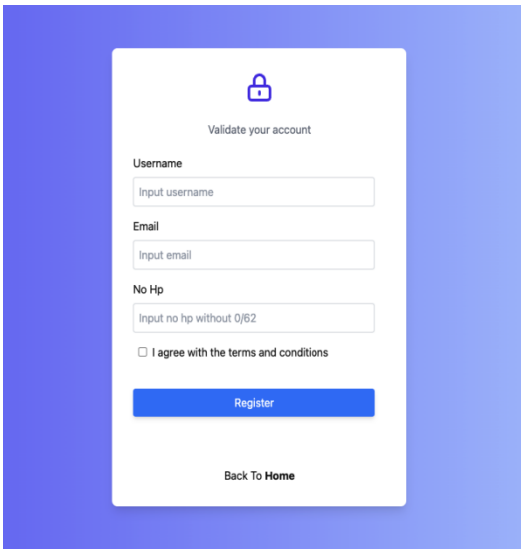
Tabel IV. 1 Tabel Halaman Website

Halaman / Komponen	Deskripsi Fungsionalitas	Contoh Fitur Utama
Halaman *Login*/Registrasi	Berfungsi sebagai gerbang autentikasi bagi pengguna (pelanggan) untuk mengakses fitur-fitur pemesanan.	Menyediakan *form input username/*email dan *password*, tombol *login*, opsi "Lupa Password", serta tautan untuk registrasi akun baru. Pada *form* registrasi, pengguna mengisi data pribadi seperti nama, email, nomor HP, dan *display name*.
Halaman Beranda	Halaman ini dirancang untuk menyajikan sorotan produk, promosi terkini, atau koleksi produk unggulan Mochi Studio.	Mencakup banner promosi, *carousel* produk-produk pilihan, serta navigasi kategori produk untuk memudahkan eksplorasi.

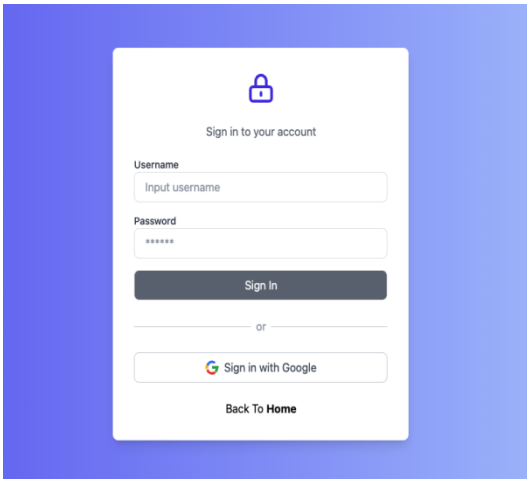
Halaman Produk & Detail	Halaman ini menampilkan daftar produk yang tersedia di Mochi Studio. Saat pengguna memilih produk, halaman detail akan menampilkan informasi yang lebih komprehensif, termasuk opsi kustomisasi atau spesifikasi pre-order.	Pada Daftar Produk, ditampilkan foto, judul, harga, dan tag relevan (isPo, best seller). Detail Produk mencakup galeri gambar, deskripsi (*desc*), rata-rata rating (rate), ulasan pelanggan (productComment), opsi kustomisasi (jika data produk mendukung), pilihan productPrice (harga dan diskon), indikator ketersediaan stok (qty), tombol "Tambah ke Keranjang", dan informasi *process* (alur kerja kustom) untuk produk personalisasi.
Halaman *Checkout*	Merupakan tahap final dalam proses pemesanan, di mana pengguna melengkapi data pengiriman, meninjau ringkasan order, dan memilih metode pembayaran.	Meliputi input alamat pengiriman (userAddress), pilihan jasa pengiriman, ringkasan OrderDetail yang mencakup product, productPrice, qty, informasi paymentStatus, dan tombol "Konfirmasi Pesanan".
Halaman Status Pesanan	Ini adalah komponen krusial yang menyediakan informasi terkini mengenai status setiap order pelanggan, termasuk progres pengerjaan dan pelacakan pengiriman. Halaman ini mengintegrasikan konsep *Finite State Machine* (FSM) yang telah saya rancang.	Daftar Order menampilkan orderStatus untuk setiap pesanan. Pada Detail Order, pengguna dapat melihat state FSM (orderStatus seperti PENDING, PROCESSING, SHIPPED), detail orderProgress (misalnya "Desain Disetujui", "Produksi Dimulai"), nomor resi (awbShipment), *tracking* pengiriman, riwayat transaksi (payment), tombol konfirmasi penerimaan (confirmDelivery), serta opsi untuk memberikan *review* (isReview).

Admin Dashboard/Panel	*Interface* ini dirancang untuk manajemen internal Mochi Studio, mencakup pengelolaan produk, stok, transaksi, pengguna, serta pemantauan detail *progress order*.	Manajemen Produk (CRUD untuk product, productPrice, productGallery). Manajemen Order (melihat daftar order, mengubah orderStatus yang memicu FSM, update orderProgress—misalnya menandai "Desain Selesai", input awbShipment). Manajemen Pengguna (CRUD user). Laporan seperti statistik penjualan dan order berdasarkan status.

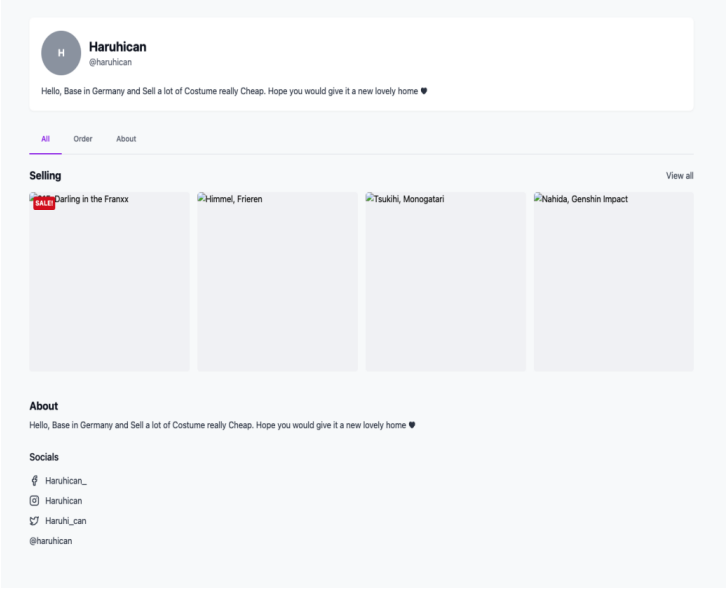
Berikut *User Interface* yang ada pada sistem :



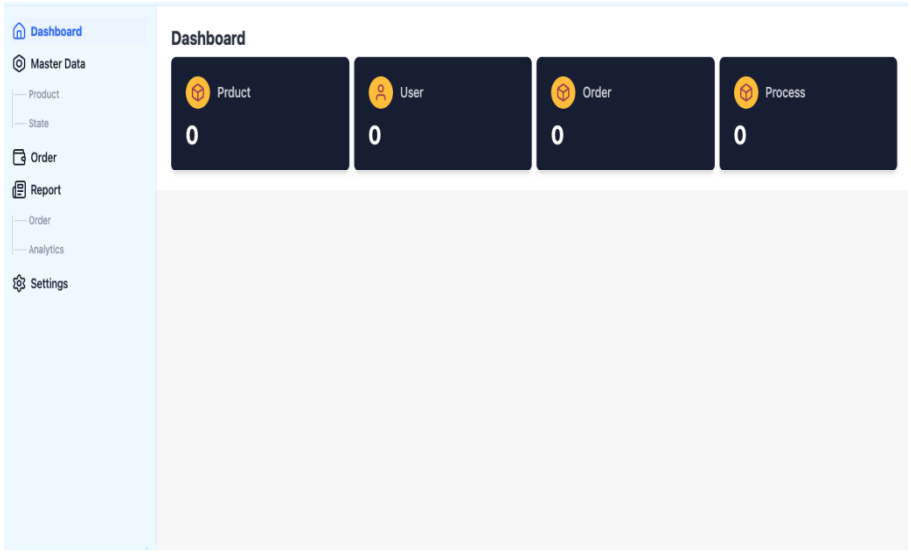
Gambar 4. 11 Halaman Registrasi



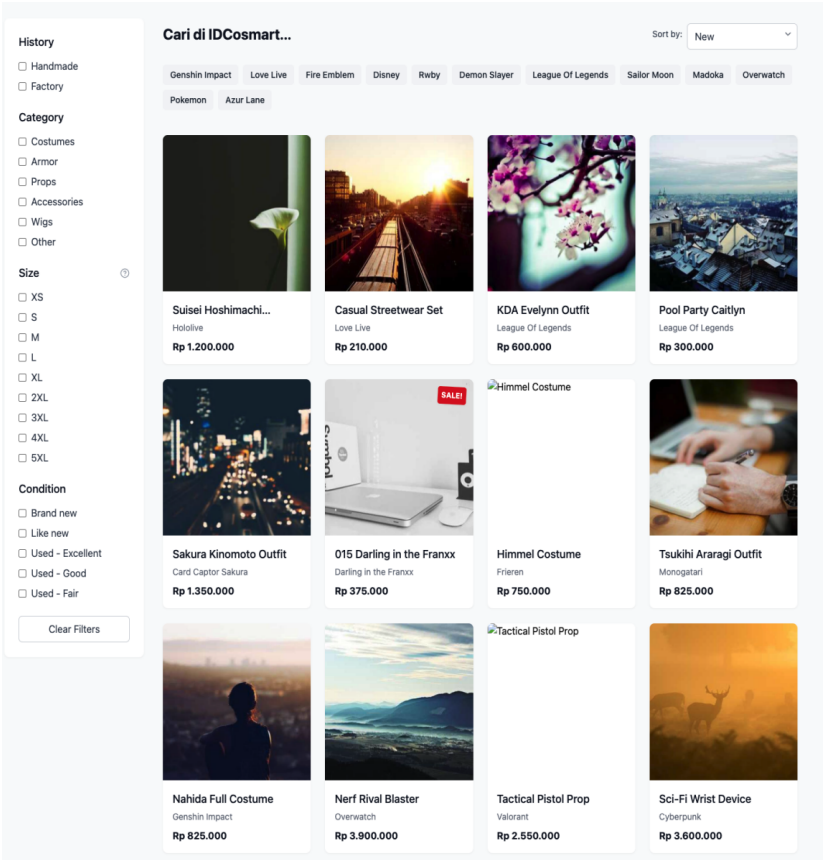
Gambar 4. 12 Halaman Login



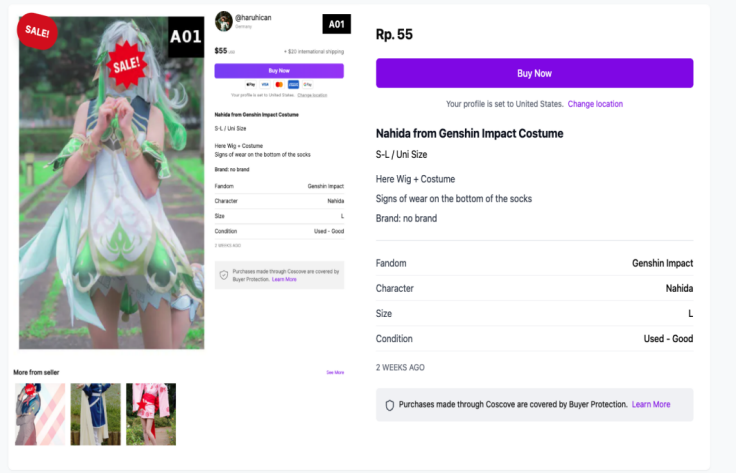
Gambar 4. 13 Halaman Beranda Pelanggan



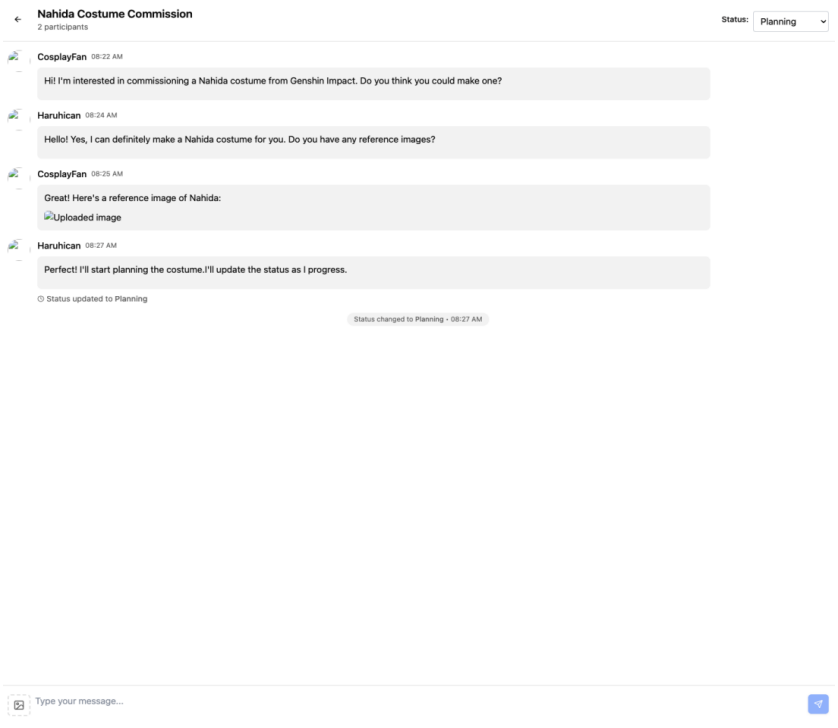
Gambar 4. 14 Halaman Beranda Admin



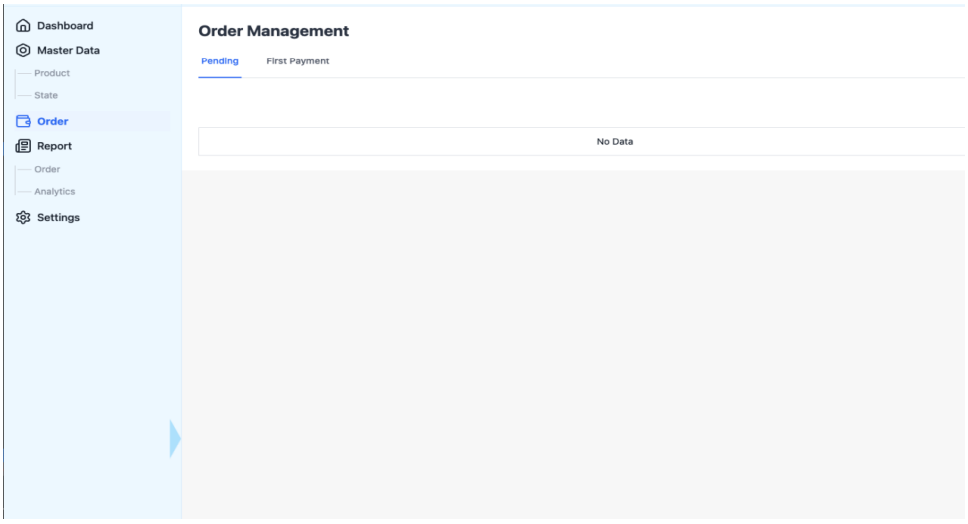
Gambar 4. 15 Halaman Produk



Gambar 4. 16 Halaman Detail Produk



Gambar 4. 17 Halaman Pemesanan



Gambar 4. 18 Halaman Laporan Pemesanan Admin

3. Desain Visual dan UX

Saya memberikan perhatian khusus pada aspek visual dan pengalaman pengguna:

a. Desain Bersih & Minimalis: Saya memilih desain yang bersih dan minimalis untuk menonjolkan visual produk Mochi Studio yang artistik. Palet warna akan disesuaikan dengan branding Mochi Studio, kemungkinan kombinasi warna pastel atau cerah yang lembut, untuk menciptakan suasana yang *playful* namun tetap profesional dan elegan.
b. Navigasi Intuitif: Navigasi dirancang agar sederhana dan mudah diakses. Saya akan menggunakan elemen seperti *sticky header* pada desktop dan menu *hamburger* pada perangkat *mobile* untuk mengarahkan pengguna dengan cepat ke halaman-halaman penting seperti Produk, Keranjang, Pesanan Saya, dan Akun.
c. Responsivitas: Desain sistem ini akan sepenuhnya responsif, memastikan tampilan dan fungsionalitas optimal di berbagai jenis perangkat, mulai dari desktop, tablet, hingga smartphone. Hal ini menjamin pengalaman pengguna yang konsisten di mana pun mereka mengakses website.
d. *Feedback* Interaktif: Saya mengimplementasikan berbagai bentuk *feedback* interaktif yang jelas kepada pengguna:

e. Indikator *Loading*: Digunakan saat memuat data atau memproses transaksi, memberikan informasi visual bahwa sistem sedang bekerja.
f. Validasi *Form*: Pesan kesalahan akan ditampilkan secara inline dan jelas saat pengguna mengisi form dengan data yang tidak valid, memandu mereka untuk koreksi.
g. Visualisasi *Progress*: Untuk melacak orderProgress, saya akan menggunakan *progress bar* atau *timeline* visual pada halaman status pesanan. Ini memberikan representasi *real-time* tentang tahapan pengerjaan order, sesuai dengan state FSM yang sedang berjalan.

4. *Tools* dan Implementasi

Dalam proses perancangan dan pengembangan UI:

a. Desain UI/UX: Saya menggunakan *tools* seperti Figma untuk membuat *wireframe*, *mockup*, dan prototipe interaktif. Penggunaan *tool* ini memungkinkan saya untuk melakukan user *testing* awal dan iterasi desain secara efisien sebelum masuk ke tahap pengembangan kode.
b. *Frontend Framework*: Implementasi *frontend* akan menggunakan *framework* JavaScript populer seperti React.js. *Framework* ini dipilih karena kemampuannya dalam menciptakan *declarative* UI dan pengelolaan *state* yang efisien, yang sangat cocok untuk berinteraksi dengan *backend* Node.js yang saya kembangkan.
c. Komponen UI *Library*: Untuk mempercepat pengembangan dan menjaga konsistensi tampilan, saya akan memanfaatkan UI component library seperti Material-UI atau mengadopsi customizable CSS *framework* seperti Tailwind CSS. Ini menyediakan komponen siap pakai yang sudah teruji dan responsif.

5. Contoh Alur Pengguna (dengan FSM Implisit)

Berikut adalah alur pengguna dalam sistem pemesanan Mochi Studio, dengan implikasi FSM pada setiap tahapnya.

1. Pada sisi pelanggan :

a. Pengguna Membuka *Website*: Pengguna akan melihat *splash screen* Mochi Studio, kemudian secara otomatis diarahkan ke halaman beranda.

b. Login atau Registrasi: Pengguna akan masuk ke akun yang sudah ada atau mendaftar akun baru (User dibuat/diverifikasi).

c. Melihat Produk: Pengguna menjelajahi daftar product Mochi Studio, melihat detail produk, productPrice, dan opsi kustomisasi yang tersedia.

d. Memilih & Memesan Produk: Pengguna memilih produk yang diinginkan, menentukan jumlah (qty), dan menambahkannya ke *cart*.

e. Setelah selesai memilih, pengguna melanjutkan ke halaman *checkout*.

f. Proses *Checkout*: Pengguna mengkonfirmasi atau memilih alamat pengiriman (userAddress), memilih jasa pengiriman, dan memilih metode pembayaran.

g. Sistem kemudian membuat entitas order baru dengan orderStatus awal PENDING. Ini secara implisit memicu transition CREATE_ORDER dalam FSM.

h. Pengguna melanjutkan proses pembayaran dan bisa melalui *send image* melalui *chat system*. Setelah pembayaran berhasil diverifikasi (paymentStatus: true dan entri payment baru tercatat), FSM akan memicu *event* paymentDpReceived atau paymentFullReceived, yang kemudian akan mengubah orderStatus menjadi DP_PAID atau PAID.

i. Ketika orderStatus telah mencapai PAID atau DP_PAID, admin Mochi Studio akan memicu FSM startProcessing. Status order di sistem akan berubah menjadi PROCESSING.
j. Pada halaman ini, pengguna akan melihat timeline orderProgress yang diperbarui secara berkala oleh admin (misalnya "Desain Dikerjakan", "Produksi Berlangsung", "Pengecekan Kualitas").
k. Setelah semua tahapan pengerjaan diselesaikan, admin akan memicu FSM markReadyForShipment, mengubah orderStatus menjadi READY_FOR_SHIPMENT.
l. Ketika admin menginput awbShipment dan memicu FSM shipOrder, orderStatus akan berubah menjadi SHIPPED. Pengguna kemudian dapat melacak status pengiriman melalui nomor resi yang diberikan.
m. Setelah pesanan sampai di tangan pelanggan dan dikonfirmasi (baik oleh pelanggan atau melalui sistem pelacakan otomatis yang memicu FSM confirmDelivery), orderStatus akan berubah menjadi DELIVERED.
n. Setelah periode tertentu (misalnya, 3 hari) atau setelah pengguna memberikan review produk (isReview), FSM completeOrderProcess akan dipicu, menandai orderStatus sebagai COMPLETED.

2. Manajemen Admin (Melalui Admin Dashboard/Panel):

Admin memiliki akses untuk melihat seluruh daftar order, mengubah orderStatus (secara langsung memicu transisi FSM), menginput awbShipment, dan memperbarui detail orderProgress untuk setiap pesanan. Selain itu, admin bertanggung jawab dalam pengelolaan product, productPrice, dan informasi *user* di dalam sistem.

Desain UI yang saya rancang, dengan integrasi erat bersama *Finite State Machine*, bertujuan tidak hanya untuk meningkatkan kenyamanan pelanggan dalam memesan produk Mochi Studio, tetapi juga untuk memberikan transparansi yang tinggi mengenai *progres* pesanan. Hal ini secara signifikan akan memudahkan manajemen operasional bagi pihak Mochi Studio, menciptakan sistem yang efisien dan *user-friendly*.

4.3. *Code Generation*

Dalam pengembangan aplikasi Mochi Studio yang menggunakan Node.js (Express.js) untuk backend dan Nuxt.js untuk *frontend*, *code generation* dapat secara signifikan mempercepat proses pengembangan, mengurangi kesalahan manual, dan memastikan konsistensi kode.

Berikut adalah gambaran dan rekomendasi terkait *code generation* untuk aplikasi Mochi Studio:

4.3.1. *Code Generation* untuk *Backend* (Express.js)

Untuk *backend* Mochi Studio, code generation dapat diterapkan pada berbagai lapisan:

a. **Struktur Proyek Awal**

Menggunakan *tools* seperti express-generator (atau custom script yang dibuat sendiri) untuk membuat struktur folder dasar proyek Express.js, termasuk direktori untuk routes, controllers, models, middlewares, dan konfigurasi.

b. **API *Endpoints***

Membuat *script* atau menggunakan *library* yang dapat meregenerasi *boilerplate* untuk *routes* (misalnya, GET /api/items, POST /api/items, PUT /api/items/:id, DELETE /api/items/:id). Secara otomatis membuat controller yang sesuai dengan fungsi CRUD (Create, Read, Update, Delete) untuk setiap entitas (misalnya, itemController.js, orderController.js, userController.js). Ini dapat mencakup template dasar untuk validasi *input* dan penanganan respons.

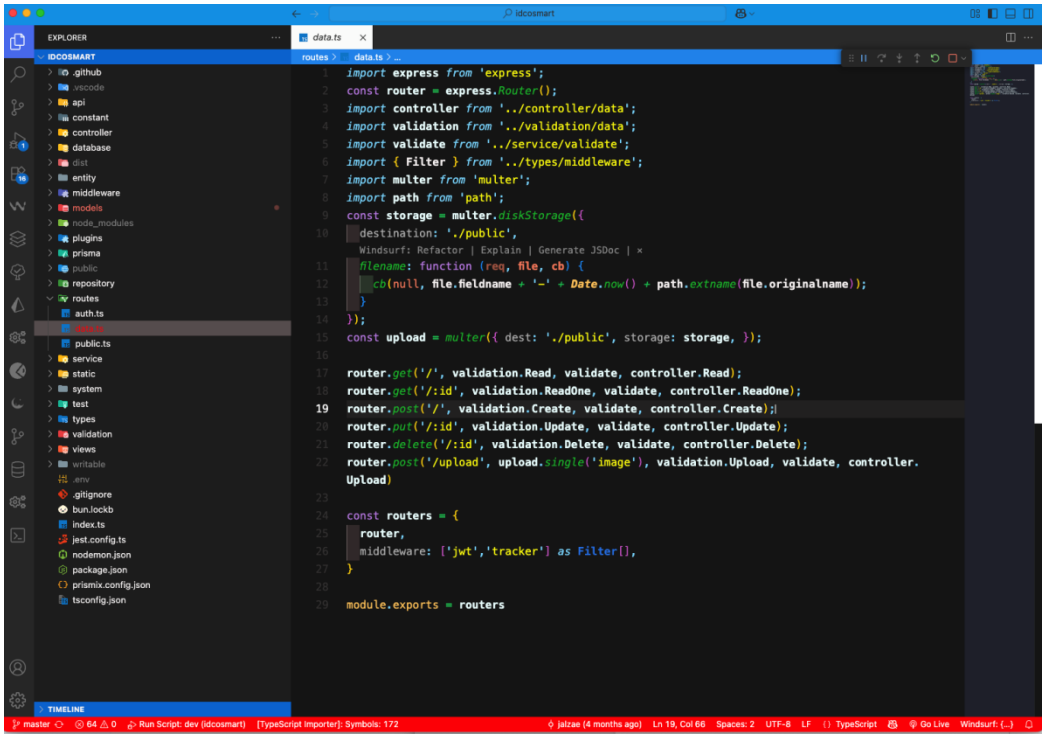
c. Model *Database*

Jika menggunakan ORM (*Object-Relational Mapping*) seperti Sequelize (untuk MySQL/PostgreSQL) atau Mongoose (untuk

MongoDB), *code generation* dapat digunakan untuk membuat definisi model berdasarkan skema *database*.

d. Autentikasi dan Otorisasi

Menggenerasi *boilerplate* untuk *middleware* autentikasi (misalnya, verifikasi JWT) dan *middleware* otorisasi (misalnya, pengecekan peran pengguna).



Gambar 4. 19 Struktur Express Js

4.3.2. *Code Generation* untuk *Frontend* (Nuxt.js)

Nuxt.js sendiri sudah memiliki banyak fitur *code generation* bawaan yang sangat membantu:

a. *Scaffolding* Proyek Nuxt.js

Perintah npx nuxi init <project-name> akan menginisialisasi proyek Nuxt.js dengan struktur folder standar (pages, components, layouts, assets, public, server, dll.).

b. Generasi Halaman (*Pages*)

Nuxt.js secara otomatis membuat routing berdasarkan struktur *folder pages*. Namun, *custom script* dapat dibuat untuk menggenerasi boilerplate halaman dengan template dasar (misalnya, halaman daftar produk, halaman detail produk, halaman keranjang).

c. Generasi Komponen (*Components*)

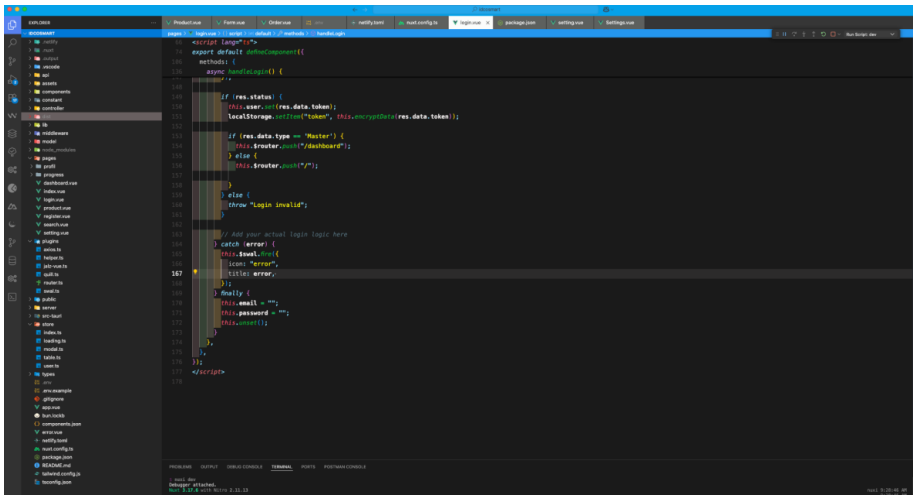
Membuat *boilerplate* untuk komponen Vue.js yang sering digunakan (misalnya, ProductCard.vue, OrderItem.vue, NotificationToast.vue). Ini dapat mencakup template dasar dengan props dan emits yang umum.

d. Integrasi API *Services*

Menggenerasi template untuk file layanan API yang akan berinteraksi dengan *backend* Express.js. Misalnya, api/products.js yang berisi fungsi untuk getProducts(), createProduct(), dll.

e. *State Management* (Pinia)

Menggenerasi *boilerplate* untuk store Pinia, termasuk definisi state, getters, actions, dan mutations (jika menggunakan Vuex Options API).



Gambar 4. 20 Struktur NuxtJs

4.3.3. Integrasi dengan *Tools* Otomasi (VS Code & Windsurf)

Untuk lebih mengoptimalkan proses *code generation* dan otomasi, integrasi dengan *Integrated Development Environment* (IDE) dan *tools* pihak ketiga sangat direkomendasikan:

1. **Visual Studio Code (VS Code)**
    a. *Snippet Extensions*: Membuat *custom snippets* di VS Code untuk blok kode yang sering digunakan. Misalnya, *snippet* untuk membuat controller Express.js baru atau komponen Nuxt.js dengan struktur dasar.
    b. *Task Automation*: Mengkonfigurasi tasks.json di VS Code untuk menjalankan *script code generation* secara langsung dari IDE. Contohnya, menjalankan perintah npm run generate:model <modelName> yang akan memicu *script* untuk membuat model baru.
    c. *Extensions*: Memanfaatkan ekstensi VS Code yang mendukung *code generation* atau *scaffolding* untuk Node.js/Express.js dan Vue/Nuxt.js.
    d. Windsurf (atau *Tools* Serupa untuk Otomasi Alur Kerja):

    Meskipun "Windsurf" mungkin bukan *tool code generation* yang umum, konsep *tool* otomasi alur kerja (seperti task runners atau build automation tools) sangat relevan.

    e. *Custom Scripting*: Mengembangkan *custom script* (misalnya dengan Node.js sendiri, Python, atau shell scripts) yang dapat diintegrasikan ke dalam alur kerja CI/CD (*Continuous Integration/Continuous Deployment*) atau dijalankan secara lokal.

Manfaat *Code Generation* untuk Mochi Studio yang didapatkan sebagai berikut :

a. Percepatan Pengembangan: Mengurangi waktu yang dihabiskan untuk menulis kode berulang.
b. Konsistensi Kode: Memastikan semua bagian aplikasi mengikuti standar dan pola yang sama.
c. Mengurangi Kesalahan: Mengeliminasi kesalahan ketik dan kesalahan manual yang sering terjadi saat menulis kode berulang.
d. Fokus pada Logika Bisnis: Pengembang dapat lebih fokus pada implementasi logika bisnis inti daripada *boilerplate*.
e. Memudahkan *Onboarding*: Pengembang baru dapat lebih cepat memahami struktur proyek dan mulai berkontribusi.

Dengan memanfaatkan *code generation* secara efektif, Mochi Studio dapat membangun sistem pemesanan barang yang lebih cepat, lebih andal, dan lebih mudah dikelola.

4.4. *Testing*

Proses pengujian sistem akan dilakukan secara menyeluruh untuk memastikan kualitas dan keandalan. Pendekatan *Gray Box Testing* akan digunakan, menggabungkan pemahaman tentang struktur internal kode (seperti alur *Finite State Machine*) dengan pengujian fungsional dari sudut pandang pengguna. Pengujian akan mencakup beberapa aspek utama:

1. ***Unit Testing**:* Menguji komponen atau modul kode secara individual untuk memastikan setiap bagian kecil berfungsi dengan benar sebelum diintegrasikan.
2. ***Automated Testing***:
    a. **API Testing**: Menguji *endpoint* API *backend* yang dibangun dengan Express.js untuk memastikan data dikirim dan diterima dengan benar, serta logika bisnis berjalan sesuai harapan.
    b. ***Frontend Testing*:** Menguji antarmuka pengguna yang dibangun dengan Nuxt.js untuk memastikan interaksi elemen, tampilan, dan alur pengguna berjalan mulus tanpa bug.

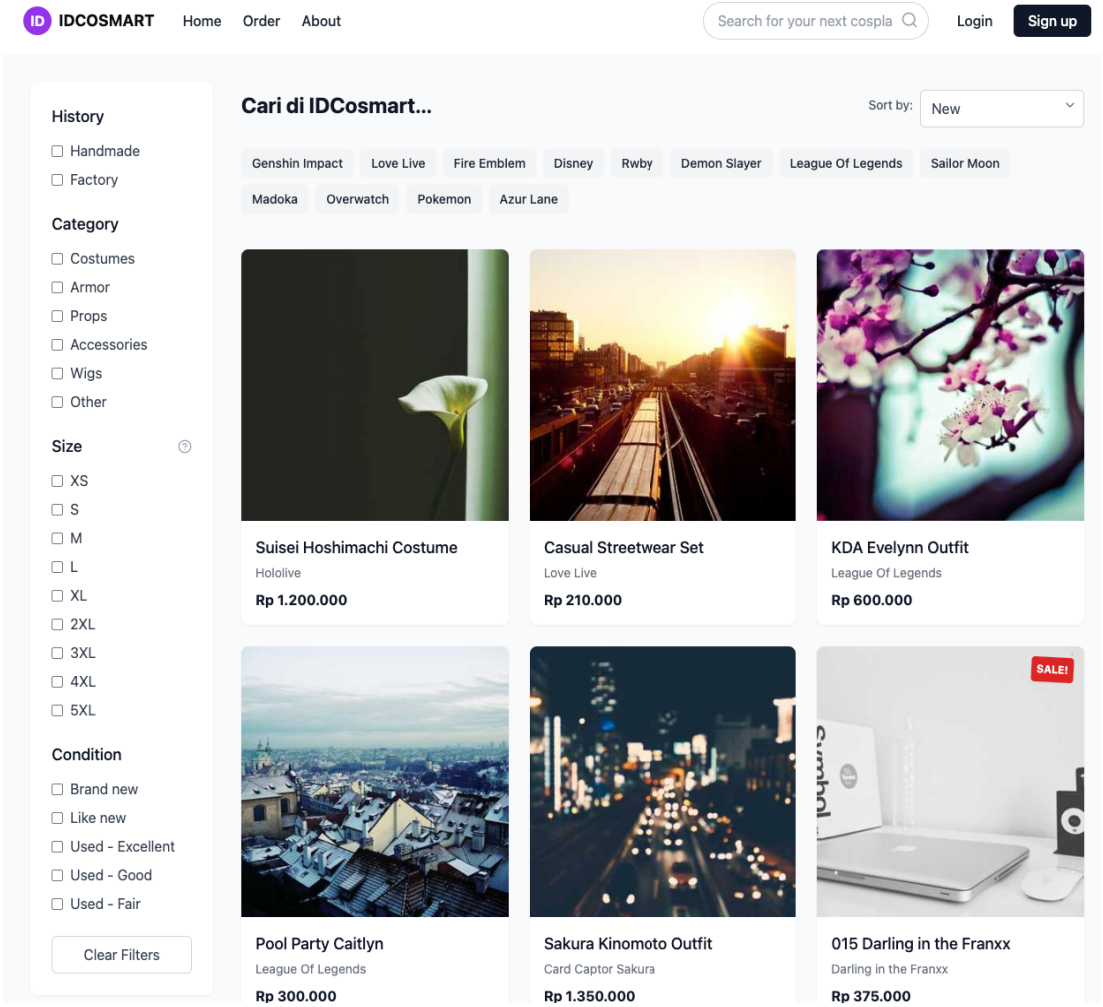
3. ***Functional Testing***: Memastikan semua fitur utama berjalan sesuai spesifikasi, mulai dari proses pendaftaran dan *login* pelanggan, pengunggahan desain, pemilihan spesifikasi produk, hingga perubahan status pesanan oleh admin dan notifikasi kepada pelanggan.
4. ***Usability Testing***: Menguji kemudahan penggunaan *website* oleh pelanggan dan admin Mochi Studio, memastikan antarmuka yang intuitif dan navigasi yang lancar agar pengguna dapat menyelesaikan tugas dengan efisien.
5. ***Performance Testing***: Menguji kemampuan sistem untuk menangani volume pesanan yang tinggi atau banyak pengguna yang mengakses secara bersamaan, memastikan Node.js memberikan respons yang cepat tanpa lag.
6. ***Security Testing***: Memastikan data pelanggan dan transaksi pesanan aman dari potensi ancaman siber seperti SQL Injection, Cross-Site Scripting (XSS), atau serangan lainnya, melindungi privasi dan integritas data.
7. ***Compatibility Testing*:** Menguji sistem di berbagai browser web dan perangkat (desktop, tablet, smartphone) untuk menjamin pengalaman pengguna yang optimal di mana pun sistem diakses.

Pengujian yang komprehensif ini bertujuan untuk memastikan bahwa Sistem Pemesanan Cetak Tekstil Mochi Studio dapat beroperasi secara stabil dan andal. Dengan demikian, sistem akan mendukung kegiatan operasional Mochi Studio secara lebih efisien, mengurangi kesalahan manual, dan meningkatkan kepuasan pelanggan melalui layanan pemesanan yang mudah, cepat, dan transparan.

4.5. *Support*

4.5.1 Publikasi Web

Sistem Pemesanan Barang yang dikembangkan untuk Mochi Studio merupakan sebuah aplikasi berbasis *website* yang dirancang menggunakan teknologi Node.js untuk mendukung operasional studio secara efisien dan terintegrasi. Nantinya website ini akan di *deploy* pada alamat idcosmart.netlify.app.



Gambar 4. 21 Publikasi Web di Netlify

1. **Deskripsi Sistem**
    a. Sistem ini memfasilitasi proses pengelolaan pesanan barang, pemantauan status order secara *real-time*, dan pemesanan barang oleh pelanggan maupun admin studio. Dengan fitur yang mudah digunakan dan berbasis *web*, sistem ini memungkinkan

akses kapan saja dan dari mana saja tanpa perlu instalasi aplikasi tambahan, mendukung fleksibilitas operasional Mochi Studio.

2. Teknologi yang Digunakan

a. ***Backend***: Node.js dengan *framework* Express.js sebagai *server-side runtime environment* yang handal dan scalable.

b. ***Frontend***: Nuxt.js (berbasis Vue.js) untuk antarmuka pengguna yang responsif, dinamis, dan memberikan pengalaman terbaik di berbagai perangkat.

c. ***Database***: MongoDB/MySQL (tergantung arsitektur yang dipilih) untuk penyimpanan data barang, transaksi pesanan, dan pengguna.

d. **API**: RESTful API untuk komunikasi yang efisien dan terstruktur antara *frontend* dan *backend*.

e. ***Deployment***: *Backend* akan di-*deploy* pada Ubuntu Server untuk stabilitas dan kontrol penuh, sementara *frontend* (Nuxt.js) akan di-*deploy* menggunakan Netlify untuk *hosting* yang cepat, aman, dan mudah dikelola, memastikan aksesibilitas luas oleh pelanggan dan tim Mochi Studio.

3. Manfaat Publikasi Sistem Web

Pengelolaan Data Lebih Efisien: Mengotomatiskan proses pemantauan pesanan dan pengelolaan barang yang sebelumnya manual, mengurangi beban kerja administratif.

a. Kemudahan Akses: Pelanggan dan admin dapat dengan mudah melakukan pemesanan dan memantau status order dari perangkat apapun yang terhubung internet.

b. Transparansi dan Akurasi: Meminimalisir kesalahan data pesanan dan status transaksi yang dapat terjadi pada proses manual, meningkatkan kepercayaan pelanggan.

c. Peningkatan Layanan: Memberikan respon cepat terhadap permintaan dan pemesanan pelanggan melalui fitur pemesanan *online* dan pelacakan status *real-time*.
d. Skalabilitas: Sistem berbasis Node.js memungkinkan penambahan fitur dan pengguna seiring perkembangan bisnis Mochi Studio di masa mendatang.

4. Proses Publikasi

Proses publikasi website sistem ini terdiri dari 2 aplikasi terpisah antara *frontend* dan *backend.* Pada publikasi *backend* akan di tempatkan pada repository kemudian di*compile* pada server serta dirunning menggunakan *Process Manager.* Pada bagian *frontend* akan langsung di *deploy* pada hosting jamstick seperti keterangan pada *deploy* diagram.

4.5.2 Spesifikasi Hardware dan Software

Spesifikasi hardware dan software untuk sistem pemesanan print textile dengan model *Finite State Machine* untuk Mochi Studio diperkirakan sebagai berikut :

Tabel IV. 2 Tabel Spesifikasi

Aspek	**Spesifikasi Hardware**	**Spesifikasi Software**
Server	- CPU minimal quad-core (contoh: AMD Ryzen / Intel Core i5 ke atas) untuk menangani banyak request secara simultan - RAM minimal 8 GB agar mendukung proses Node.js yang optimal - Media penyimpanan SSD untuk akses data yang cepat	- Node.js (*runtime environment* berbasis JavaScript menggunakan V8 engine) - Modul *clustering* Node.js untuk meningkatkan performa dan distribusi beban CPU - Database seperti MongoDB atau MySQL untuk penyimpanan data stok, pengguna, dan transaksi - Web server seperti Nginx atau

		Apache sebagai reverse proxy dan load balancer pada deployment produksi
Client	- Perangkat desktop/laptop atau smartphone dengan kemampuan akses browser internet	- Browser modern (Google Chrome, Mozilla Firefox, Microsoft Edge, dll.) yang mendukung HTML5, CSS3, dan JavaScript
Jaringan	- Koneksi internet stabil, minimal broadband untuk akses web secara real-time	- Protokol HTTP/HTTPS untuk akses website, serta implementasi keamanan SSL/TLS agar data transaksi terproteksi

4.6. Spesifikasi Dokumen Sistem Usulan

Dokumen spesifikasi sistem merupakan dokumen yang berisi nantinya bagaimana sistem dibuat dan berjalan. Ini akan menjadi penting untuk pengguna dalam menjalankan dan mengembangkan sistem. Dengan adanya sistem ini maka berikut urutan dokumen sistem yang ada pada sistem sebagai berikut :

1. Pendahuluan

a. **Latar Belakang**: Dokumen ini menjelaskan kebutuhan Mochi Studio untuk beralih dari proses pengelolaan pesanan dan barang yang masih konvensional (menggunakan WhatsApp dan Facebook sebagai catatan order manual) menuju sistem yang terdigitalisasi. Proses manual ini

seringkali menimbulkan tantangan dalam efisiensi, akurasi data, dan pelacakan status pesanan secara real-time.

b. **Tujuan Sistem:** Tujuan utama pengembangan sistem ini adalah untuk meningkatkan efisiensi pengelolaan pesanan dan barang, mempercepat proses transaksi, menyediakan pelacakan status pesanan yang transparan menggunakan model *Finite State Machine* (FSM), serta mendukung pengambilan keputusan yang lebih baik bagi Mochi Studio.

c. **Ruang Lingkup Sistem**: Sistem ini akan mencakup fungsi-fungsi utama seperti pendaftaran dan manajemen profil pelanggan, katalog barang, proses pemesanan online, pelacakan status pesanan, manajemen transaksi oleh admin, serta pembuatan laporan. Aktor utama yang akan berinteraksi dengan sistem adalah Admin Mochi Studio dan Pelanggan.

2. Analisis Sistem

a. **Analisis Sistem Eksisting**: Saat ini, Mochi Studio mengandalkan komunikasi langsung melalui WhatsApp dan Facebook untuk menerima pesanan. Pencatatan pesanan dan detail barang dilakukan secara manual, yang rentan terhadap kesalahan, duplikasi data, dan kesulitan dalam memantau kemajuan setiap order. Proses pengerjaan order juga masih manual dan belum terintegrasi dengan sistem pencatatan.

b. **Elisitasi Kebutuhan**: Pengumpulan kebutuhan fungsional dan non-fungsional akan dilakukan melalui wawancara dengan staf Mochi Studio dan observasi alur kerja yang ada. Kebutuhan ini kemudian akan dikelompokkan dalam tahap-tahap elisitasi untuk memastikan semua aspek tercakup hingga draf final.

c. **Diagram UML**: Diagram UML (*Unified Modeling Language*) seperti *Use Case* Diagram dan *Activity* Diagram akan digunakan untuk merepresentasikan secara visual proses bisnis yang diusulkan dan interaksi antara pengguna dengan sistem.

3. Perancangan Sistem

a. **Spesifikasi Fungsional:** Sistem akan memiliki fungsi utama meliputi: pendaftaran dan manajemen data pelanggan, pengelolaan katalog barang (penambahan, perubahan, penghapusan item), fitur pemesanan online oleh pelanggan, pelacakan status pesanan menggunakan model Finite State Machine (misalnya, dari "Menunggu Pembayaran" hingga "Selesai"), manajemen transaksi oleh admin (konfirmasi, pembatalan, pembaruan status), serta pembuatan laporan penjualan dan produksi.
b. **Spesifikasi Non-Fungsional:** Sistem akan dirancang untuk memiliki performa yang optimal, keamanan data dan transaksi yang kuat, antarmuka yang user-friendly dan intuitif, serta platform berbasis website yang responsif.
c. **Desain Basis Data:** Definisi tabel dan relasi antar tabel akan dirancang untuk menyimpan data pengguna, data barang, data pesanan, status transaksi, dan informasi terkait lainnya. Desain ini akan mengacu pada model data fisik yang valid untuk memastikan integritas dan efisiensi query.
d. **Arsitektur Sistem**: Sistem akan mengadopsi arsitektur *client-server.* Bagian *backend* akan dibangun menggunakan Node.js dengan framework Express.js untuk menangani logika bisnis dan API. Bagian frontend akan dikembangkan menggunakan Nuxt.js (berbasis Vue.js) untuk antarmuka pengguna yang dinamis dan responsif. MySQL atau PostgreSQL akan digunakan sebagai sistem manajemen database relasional.
e. **Desain Antarmuka**: *Mockup* antarmuka pengguna akan dibuat untuk memvisualisasikan tata letak, elemen interaktif, dan alur navigasi, memastikan kemudahan akses dan pengalaman pengguna yang optimal.

4. Implementasi dan Pengujian

a. **Metode Implementasi:** Tahapan pengembangan perangkat lunak akan melibatkan *coding* menggunakan Node.js (Express.js) dan Nuxt.js,

diikuti dengan deployment pada server hosting dengan protokol keamanan (HTTPS) untuk akses yang aman.

b. **Pengujian Sistem**: Pengujian akan dilakukan secara komprehensif menggunakan metode *Gray Box Testing,* yang mencakup pemahaman internal kode dan pengujian dari sudut pandang pengguna. Detail rencana pengujian meliputi:
c. ***Unit Testing*:** Menguji komponen kode secara individual.
d. ***Automated Testing***: Baik untuk API backend maupun tampilan *frontend*.
e. ***Functional Testing***: Memastikan semua fitur berjalan sesuai spesifikasi.
f. ***Performance Testing***: Menguji ketahanan sistem di bawah beban tinggi.
g. ***Security Testing***: Mengidentifikasi dan mengatasi kerentanan keamanan.
h. ***Compatibility Testing***: Memastikan sistem berfungsi di berbagai *browser* dan perangkat.

5. Jadwal dan Estimasi Biaya

a. **Jadwal Implementasi**: Rencana *timeline* pengerjaan akan disusun, mencakup fase analisis, desain, *coding*, pengujian, hingga *deployment* dan implementasi.
b. **Estimasi Biaya**: Perkiraan biaya yang diperlukan akan dihitung, meliputi perangkat keras (jika ada), perangkat lunak (lisensi jika diperlukan), biaya pengembangan, dan pemeliharaan.

6. Dokumentasi Pendukung

a. **Manual Pengguna**: Panduan lengkap akan disediakan untuk membantu admin Mochi Studio dan pelanggan dalam menggunakan sistem.

b. **Dokumentasi Teknis**: Dokumen ini akan merinci struktur kode, skema database, dan konfigurasi server untuk memudahkan pemeliharaan dan pengembangan di masa depan.

Dokumen sistem usulan ini berfungsi sebagai acuan pengembangan sistem pemesanan barang berbasis *website* dengan Node.js, Express.js, dan Nuxt.js untuk Mochi Studio. Diharapkan sistem ini dapat menyederhanakan proses pengelolaan pesanan dan barang secara efektif, akurat, dan terintegrasi, serta meningkatkan kepuasan pelanggan melalui layanan yang lebih modern dan efisien.