 e-ISSN: 3110-8652

Pengembangan Website SATUINBOX di PT. X

http://dx.doi.org/10.28932/jste.v2i1.13292

Received: 26 Agustus 2025 | *Revised*: 11 November 2025 | *Accepted*: 12 Desember 2025

Nathaniel Valentino Robert✉#1, Andreas Widjaja*2

#*Program Studi Teknik Informatika, Fakultas Teknologi dan Rekayasa Cerdas, Universitas Kristen Maranatha*
Jl. Prof. drg. Surya Sumantri No.65, Bandung, Jawa Barat 40164, Indonesia
1 2272028@maranatha.ac.id

**Program Studi Magister Ilmu Komputer, Fakultas Teknologi dan Rekayasa Cerdas, Universitas Kristen Maranatha*
Jl. Prof. drg. Surya Sumantri No.65, Bandung, Jawa Barat 40164, Indonesia
2 andreas.widjaja@maranatha.ac.id
✉Corresponding author: 2272028@maranatha.ac.id

How to cite this article:

N. V. Robert, A. Widjaja, "Development of the SATUINBOX Website at PT. X," *Journal of Smart Technology and Engineering*, vol. 2, no. 1, pp. 01-13, 2026. http://dx.doi.org/10.28932/jste.v2i1.13292

Abstrak — **Pesatnya perkembangan bisnis online menuntut solusi untuk pengelolaan komunikasi pelanggan yang efektif. PT. X mengembangkan platform berbasis tiket percakapan untuk mempermudah pengelolaan komunikasi antara penjual dan pelanggan. Tujuan dari kerja praktik ini adalah untuk mengembangkan frontend website dengan fokus pada implementasi desain UI/UX dan integrasi API agar tampilan responsif dan fungsional. Metode yang digunakan adalah Agile Scrum dengan pendekatan daily sprint dalam pengembangan perangkat lunak. Teknik pengumpulan data dilakukan melalui observasi langsung dan wawancara dengan tim pengembang. Selama kerja praktik, pemagang bertanggung jawab untuk mengimplementasikan desain UI/UX melalui proses slicing page, serta mengintegrasikan API backend dengan menggunakan Next.js dan TypeScript. Pengujian responsivitas dilakukan untuk memastikan tampilan berfungsi dengan baik pada berbagai perangkat. Hasil yang dicapai menunjukkan bahwa platform berhasil diterapkan dengan tampilan yang responsif dan fitur yang berfungsi secara optimal, termasuk integrasi API dan optimasi performa. Kesimpulan dari penelitian ini adalah bahwa pengembangan frontend berhasil meningkatkan kinerja platform dan memberikan pengalaman pengguna yang lebih baik.**

Kata kunci— **API; *frontend*; integrasi; optimasi; performa; UI/UX.**

Development of the SATUINBOX Website at PT. X

Abstract — ***The rapid growth of online businesses demands solutions for effective customer communication management. PT. X developed a ticket-based chat platform to facilitate communication management between sellers and customers. The purpose of this internship project is to develop the frontend of the website with a focus on implementing a responsive and functional UI/UX design and API integration. The method used is Agile Scrum with a daily sprint approach in software development. Data collection techniques were carried out through direct observation and interviews with the development team. During the internship, the intern was responsible for implementing the UI/UX design through page slicing and integrating the backend API using Next.js and TypeScript. Responsiveness testing was performed to ensure proper functionality across various devices. The results showed that the platform was successfully implemented with a responsive interface and features that function optimally, including API integration and performance optimization. The conclusion of this study is that the frontend development successfully improved the platform's performance and enhanced the user experience.***

Keywords— ***API; frontend; integration; optimization; performance; UI/UX.***

I. PENDAHULUAN

Dalam era digital yang ditandai dengan pesatnya pertumbuhan bisnis *online*, pengelolaan komunikasi dengan pelanggan secara efektif menjadi salah satu kunci keberhasilan. Keterlambatan dalam memberikan respons atau menangani pertanyaan pelanggan dapat secara signifikan menurunkan tingkat kepuasan, mengikis kepercayaan, dan bahkan mengakibatkan hilangnya potensi penjualan. Menyadari urgensi ini, PT. X mengembangkan Satuinbox, sebuah *platform* inovatif berbasis tiket percakapan

 e-ISSN: 3110-8652

(*chat*) yang dirancang untuk mempermudah dan mengefisienkan proses pengelolaan komunikasi antara penjual dan pelanggan. *Platform* ini tidak hanya menyederhanakan alur kerja pengelolaan percakapan, tetapi juga didukung oleh Satuinbox Internal, sebuah sistem pendukung yang memungkinkan tim admin internal melakukan pemantauan dan analisis penggunaan *platform* secara lebih terstruktur.

Keberhasilan implementasi *platform* seperti Satuinbox sangat bergantung pada kualitas antarmuka pengguna (*frontend*). Antarmuka yang intuitif, responsif, dan fungsional merupakan garda terdepan dalam interaksi pengguna dengan sistem. Oleh karena itu, pengembangan frontend yang optimal, mencakup implementasi desain *User Interface/User Experience (UI/UX)* yang cermat, integrasi *Application Programming Interface (API)* yang mulus dengan layanan *backend*, serta optimasi performa berkelanjutan, menjadi aspek krusial. Penelitian ini mendokumentasikan proses pengembangan *frontend* untuk *website* Satuinbox dan Satuinbox Internal, yang dilakukan dengan menerapkan metodologi *Agile Scrum* guna mengakomodasi kebutuhan pengembangan yang dinamis dan iteratif.

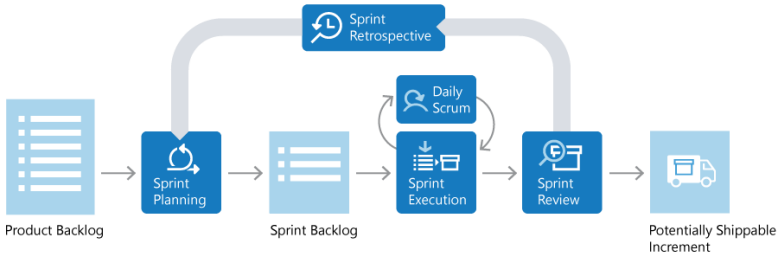
Berdasarkan tantangan dan kebutuhan tersebut, penelitian ini bertujuan untuk: (1) Mengimplementasikan desain *UI/UX* yang telah dirancang menjadi antarmuka *website* yang responsif dan intuitif melalui teknik *slicing page*; (2) Mengintegrasikan *API* yang disediakan oleh tim backend secara efisien, sehingga seluruh fitur dapat berfungsi dengan baik dan data dapat ditampilkan secara akurat dan relevan bagi pengguna; dan (3) Mengoptimalkan performa *frontend*, termasuk interaksi pengguna dan pengelolaan data dari *API*, agar *platform* tetap responsif dan handal. Kontribusi utama dari penelitian ini adalah menyajikan studi kasus pengembangan *frontend* sebuah *platform* pengelolaan komunikasi pelanggan dengan menggunakan tumpukan teknologi *modern (modern tech stack)* dan metodologi *Agile*, yang diharapkan dapat memberikan wawasan praktis bagi pengembang lain yang menghadapi tantangan serupa.

II. METODE

Penelitian dan pengembangan frontend website SATUINBOX dan SATUINBOX Internal ini dilakukan dengan beberapa tahapan metodologis yang mencakup pendekatan pengembangan perangkat lunak, pemilihan arsitektur dan teknologi, proses implementasi frontend, serta teknik pengumpulan data dan pengujian.

A. *Metodologi Pengembangan Perangkat Lunak*

Proses pengembangan perangkat lunak dalam proyek ini mengadopsi metodologi *Agile Scrum*. Pendekatan ini dipilih karena kemampuannya dalam mengakomodasi perubahan kebutuhan yang dinamis dan memungkinkan iterasi pengembangan yang berkelanjutan. Kegiatan inti dalam metodologi ini meliputi perencanaan *sprint (sprint planning)* untuk memilih pekerjaan yang akan diselesaikan dalam satu siklus *sprint*, pelaksanaan sprint *(sprint execution)* yang berfokus pada penyelesaian *sprint backlog*, pertemuan harian (*daily scrum* atau *daily sprint*) untuk evaluasi progres dan identifikasi hambatan, serta *sprint review* dan sprint *retrospective* di akhir setiap siklus untuk evaluasi hasil dan perbaikan proses kerja. *Timeline* proyek diatur dalam bentuk *sprint* bulanan, di mana setiap *sprint* memiliki target dan *deliverable* yang telah ditentukan.[1]



Gambar 1. Apa yang dimaksud dengan Scrum? [2]

B. *Arsitektur Sistem dan Teknologi*

Perancangan sistem SATUINBOX bertujuan untuk mendefinisikan arsitektur, fungsionalitas, dan alur interaksi pengguna secara komprehensif.

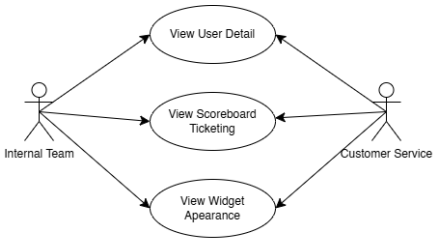
1. *Next.js*
   Next.js, sebuah kerangka kerja (*framework*) *React* sumber terbuka (*open-source*) yang dikembangkan oleh *Vercel*, digunakan untuk membangun aplikasi *web modern*. Fitur utamanya meliputi *server-side rendering (SSR)* untuk waktu muat lebih cepat dan *SEO* yang lebih baik, *static site generation (SSG)*, dan sistem *routing* otomatis berbasis *folder*. *Next.js* juga mendukung *API routes* untuk integrasi *backend*.[3]
2. *Typescript*
   Typescript adalah *superset* dari *JavaScript* yang menambahkan sistem tipe statis, memungkinkan deteksi kesalahan kode lebih awal melalui kompilasi. Hal ini sangat bermanfaat untuk proyek skala menengah hingga besar dan tim pengembang.

Typescript juga meningkatkan dokumentasi kode dengan anotasi tipe, memudahkan kolaborasi tim dan mempercepat proses *debugging*.[4]

3. *Tailwind CSS*
   Tailwind CSS, sebuah *framework CSS* berbasis *utility-first*, digunakan untuk *styling* utama pada *frontend* SATUINBOX. Pendekatan ini mempercepat proses *styling* dan menjaga konsistensi dengan kelas yang memiliki fungsi spesifik, tanpa perlu menulis *CSS* manual. *Tailwind CSS* menyediakan *breakpoint* (seperti *sm, md, lg, xl*) untuk mengatur responsivitas antarmuka di berbagai ukuran layar dan mendukung fitur seperti *dark mode* dan *hover state* langsung di *HTML*. Integrasinya dengan *PurgeCSS* otomatis menghapus kelas *CSS* yang tidak terpakai, mengurangi ukuran *file* dan meningkatkan performa aplikasi.[5]
4. *Axios*
   Axios adalah pustaka *HTTP client* berbasis *Promise* yang digunakan dalam SATUINBOX untuk mengirim *request* dan menerima *response* dari server *backend*. Fitur seperti *interceptor* dan konfigurasi *default* mempermudah pengelolaan autentikasi token dan *error handling* secara global, serta pengaturan header yang konsisten.[6]
5. *Tanstack Query*
   TanStack Query (sebelumnya *React Query*) adalah pustaka manajemen *data fetching* yang efisien, digunakan dalam proyek ini untuk menyederhanakan logika pengambilan data dan mengurangi kompleksitas kode. Fitur utamanya meliputi *caching, background refetching, pagination*, dan pengelolaan status (*loading, error, success*). Pustaka ini optimal untuk aplikasi dengan banyak permintaan data dinamis karena kemampuannya menyimpan data hasil *fetch* ke dalam *cache* dan memperbarui data secara otomatis.[7]
6. *Zustand*
   Zustand adalah pustaka manajemen *state* berbasis *hook* dengan pendekatan minimalis. Di SATUINBOX, *Zustand* dipakai untuk mengelola *state global* seperti informasi pengguna yang sedang *login*, status tampilan, dan data kontekstual lainnya yang perlu diakses lintas komponen. *Zustand* dikenal karena performanya yang tinggi, ringan (tanpa *dispatch* dan *reducer*), dan kemudahan penggunaan tanpa konfigurasi kompleks.[8]
7. *GitLab*
   GitLab berfungsi sebagai *platform version control* dan pengelolaan *CI/CD (Continuous Integration/Continuous Deployment).* Tim menggunakannya untuk menyimpan *source code* dalam repositori, melakukan *merge request*, dan meninjau perubahan kode. *Pipeline CI/CD* di *GitLab* memungkinkan *deploy* otomatis ke server *staging*, memastikan proses testing berjalan otomatis dan konsisten.[9]
8. *Google Chrome DevTools*
   Google Chrome DevTools adalah alat *debugging* penting untuk menguji tampilan dan performa antarmuka. Fitur seperti *Live DOM Viewer*, *Lighthouse Audit*, dan *Performance Monitor* membantu *developer* menganalisis kecepatan *loading*, kesalahan *rendering*, serta responsivitas halaman di berbagai ukuran layar.[10]
9. *Lark*
   Lark adalah *platform* komunikasi dan kolaborasi yang digunakan tim SATUINBOX sebagai pengganti *email* konvensional. *Lark* mengintegrasikan *fitur chat*, dokumen kolaboratif, *task management*, dan kalender, memungkinkan tim berdiskusi, berbagi referensi, dan menyusun dokumen teknis dalam satu aplikasi.[11]
10. *Atomic Design*
    Proses slicing *UI* dalam proyek SATUINBOX adalah konversi desain visual dari *Figma* menjadi komponen antarmuka fungsional berbasis *React (.tsx).* Proses ini dimulai dengan analisis struktur desain (seperti *header, footer, sidebar*, konten) yang kemudian dipecah menjadi komponen kecil yang *reusable* dan disimpan dalam *folder components/*. Pendekatan yang digunakan mengikuti prinsip *Atomic Design* (meliputi *Atoms, Molecules, Organisms, Templates, Pages*), yang membantu menjaga konsistensi antarmuka dan memudahkan pengelolaan komponen dalam skala besar. Penggunaan *Tailwind CSS* dengan pendekatan *utility-first*-nya mempercepat proses *styling* dengan menerapkan *class* langsung pada elemen *TSX*, memastikan efisiensi dan konsistensi dengan desain *UI* awal.[12][13]
11. *Code Splitting*
    Teknik ini membagi *file JavaScript* menjadi *bundle* kecil berdasarkan rute atau komponen menggunakan fitur *dynamic import()* dari *Next.js*. Dengan demikian, hanya kode yang dibutuhkan untuk halaman tertentu yang dimuat, mengurangi waktu muat awal.[14]
12. *Lazy Loading*
    Elemen seperti gambar besar atau komponen *UI* di luar *viewport* dimuat bertahap saat pengguna menggulir halaman. Ini diterapkan menggunakan atribut *loading="lazy"* untuk gambar dan *dynamic import* untuk komponen berat, meningkatkan performa pada koneksi lambat.[15]
13. *Tree-Shaking*
    Proses ini menghapus kode *JavaScript* yang tidak digunakan dari *bundle* akhir aplikasi dengan bantuan *bundler* seperti *Webpack* atau *Vite*. Ini memastikan hanya fungsi atau modul yang terpakai yang disertakan, memangkas ukuran *bundle* secara signifikan.[16]

C. Perancangan Sistem

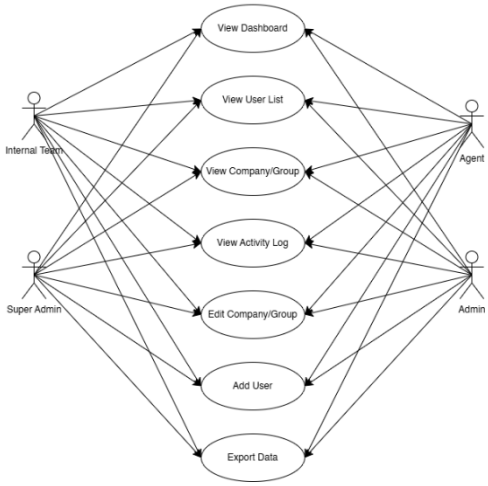
Perancangan sistem ini mencakup *Use Case Diagram*.



Gambar 2. *Use Case Diagram* SATUINBOX

Penjelasan *Use Case* SATUINBOX pada Gambar 2 :

1. *View User Detail* setiap aktor dapat melihat ke halaman *detail user* untuk melihat detail lengkap informasi user tersebut.
2. *View Scoreboard Ticketing* setiap aktor dapat melihat *scoreboard ticketing* seperti *Total Ticket, Ticket Solved, Avg, Reply Time* dan yang lainnya.
3. *View Widget Appearance* setiap user bisa mengakses *widget custom chat* pada bagian *Appearance*.



Gambar 3. *Use Case Diagram* SATUINBOX Internal

Penjelasan *Use Case* SATUINBOX pada Gambar 3 :

1. *View Dashboard* setiap aktor dapat melihat ke halaman *dashboard overview* untuk melihat ringkasan informasi atau data statistik terakhir.
2. *View User List* setiap aktor dapat melihat halaman *User List* seperti Nama Lengkap, *Username, User ID, Create Date, Role* dan sebagainya.
3. *View Company Group* setiap aktor dapat melihat halaman *Company Group* seperti Nama Company, *ID Company Total Member*, Total Divisi.
4. *View Activity Log* setiap aktor dapat melihat halaman *Activity Log* seperti *Log ID, Activity, Timestamp*, User dan *Device&Os* yang digunakan.
5. *Edit Company/Group* setiap aktor dapat mengubah nama *company/group* yang ingin diubah.
6. *Add User* setiap aktor dapat menambahkan user baru.
7. *Export Data* setiap aktor dapat mengexport data yang diperlukan seperti *User List, Company/Group* dan *Activity Log*.

III. PENGUMPULAN DATA DAN PENGUJIAN SISTEM

Untuk memahami kebutuhan sistem dan alur kerja, data primer dikumpulkan melalui:

1. Observasi Langsung: Pengamatan dilakukan pada setiap tahap pengembangan *frontend.*
2. Wawancara: Diskusi dilakukan dengan tim pengembang *frontend* dan *backend*.

Pengujian sistem difokuskan pada antarmuka pengguna (*UI*) untuk memastikan fungsionalitas, responsivitas, dan kompatibilitas:

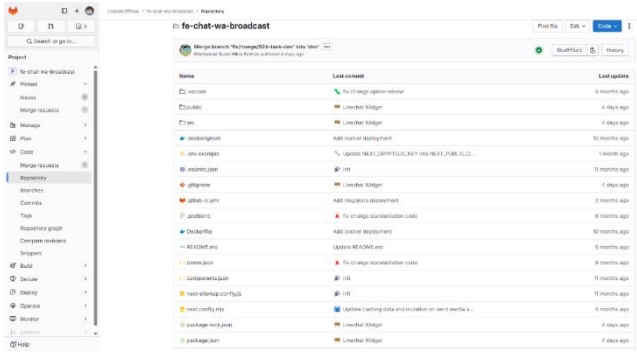
1. Pengujian Responsivitas: Dilakukan menggunakan *Google Chrome DevTools* untuk mensimulasikan berbagai ukuran layar, dengan fokus pada perangkat *laptop* dan *desktop*.
2. Kompatibilitas *Browser*: Pengujian dilakukan pada beberapa *browser* populer seperti *Chrome, Firefox*, dan *Microsoft Edge*.
3. Audit Kinerja: Alat bantu seperti *Lighthouse (via Chrome DevTools)* digunakan untuk mengevaluasi aspek *Performance, Accessibility, Best Practices*, dan *SEO*. Pengujian ini dilakukan pada tahap *staging* sebelum sistem di-*deploy* ke lingkungan produksi.

IV. HASIL DAN PEMBAHASAN

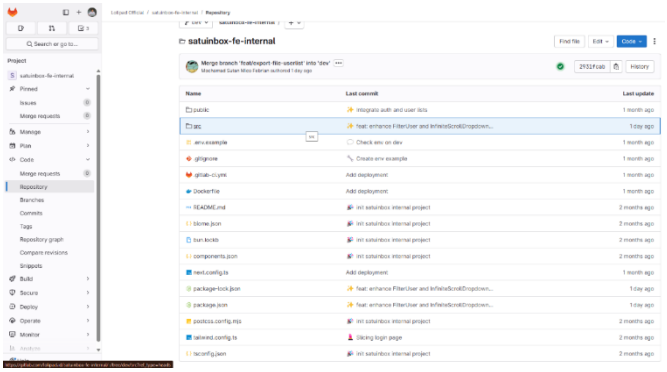
A. Tahapan Implementasi

Proses implementasi proyek SATUINBOX dilaksanakan secara bertahap berdasarkan rancangan sistem yang ada, dengan orientasi pada alur kerja tim menggunakan pendekatan *Agile*. Fitur dikembangkan berdasarkan urgensi dan kompleksitas logika bisnis, dimulai dari konfigurasi awal proyek, penyusunan struktur *folder*, pembangunan antarmuka pengguna (*UI*), integrasi dengan *backend*, hingga *deployment* ke server *staging*. Seluruh proses ini dilakukan secara kolaboratif dan terdokumentasi melalui *sprint* harian selama kerja praktik.

1. *Setup* Proyek awal
   Tahap setup awal proyek SATUINBOX dan SATUINBOX Internal meliputi inisialisasi repositori, konfigurasi struktur *folder*, instalasi dependensi, dan verifikasi inisialisasi. Proyek SATUINBOX dibangun menggunakan *framework Next.js (hybrid rendering), TypeScript, Tailwind CSS* untuk *styling*, serta kombinasi *Zustand, Axios*, dan *TanStack Query* untuk manajemen *state* dan pengambilan data.



Gambar 4. *Repository Porject* SATUINBOX



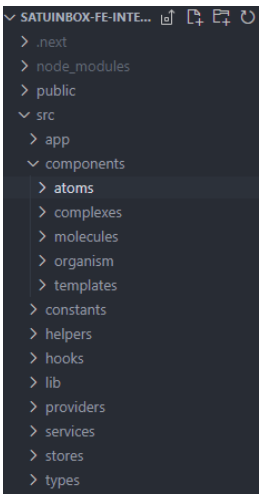
Gambar 5. *Repository Project* SATUINBOX Internal

Pada Gambar 4 dan Gambar 5 adalah dua repositori utama di-*host* di *GitLab* untuk *version control* dan kolaborasi kode:

- *fe-chat-wa-broadcast*: Repositori *frontend* untuk klien.
- satuinbox-*fe-internal*: Repositori *frontend* untuk tim internal.

Kedua repositori ini dikonfigurasi dengan *file .gitlab-ci.yml* untuk mendukung otomatisasi *deploy* ke *staging* melalui *GitLab CI/CD pipeline*.

 e-ISSN: 3110-8652



Gambar 6. *Struktur folder proyek frontend di Visual Studio Code*

Pada Gambar 6 menunjukan struktur *folder* yang dirancang menggunakan pendekatan *atomic design*, sehingga setiap *folder* seperti *atoms, molecules, organism,* dan *templates* mewakili tingkat kompleksitas komponen yang berbeda. *Folder src* menjadi direktori utama, dengan *subfolder* untuk *components, services, stores, types*, dan *hooks*.

B. Implementasi Sistem Desain Modular dengan Metodologi Atomic Design

Dalam pengembangan antarmuka (*user interface*) untuk sistem SATUINBOX dan SATUINBOX Internal, diterapkan metodologi *Atomic Design*. Pendekatan ini dipilih untuk memastikan konsistensi visual, meningkatkan modularitas, dan memaksimalkan penggunaan kembali (*reusability*) komponen di seluruh aplikasi. Seluruh komponen dibangun menggunakan *React, TypeScript*, dan di-*styling* dengan *Tailwind CSS* untuk utilitas kelas yang cepat dan efisien. Tahap awal berfokus pada pembuatan komponen fundamental yang menjadi fondasi dari semua elemen antarmuka.

1. Komponen *Atoms*

   Atoms adalah unit terkecil yang tidak dapat dipecah lagi, seperti tombol, *label*, dan input. Contoh implementasi atom yang krusial adalah komponen *Badge*. Komponen ini dirancang untuk menampilkan *status* (misalnya, 'Aktif', 'Gagal') dengan gaya visual yang berbeda-beda. Untuk mencapai reusabilitas, digunakan pustaka *class-variance-authority (CVA)* yang memungkinkan pembuatan varian komponen dengan properti (*props*) sederhana.

```
// Kode Program 1: Implementasi Varian pada Komponen Badge.tsx
const badgeVariants = cva(
  // Base styles
  "inline-flex items-center rounded-md border px-2.5 py-0.5 text-xs font-semibold...",
  {
    variants: {
      variant: {
        default: "border-transparent bg-primary text-primary-foreground...",
        secondary: "border-transparent bg-secondary text-secondary-foreground...",
        destructive: "border-transparent bg-destructive text-destructive-
foreground...",
        info: "border-transparent bg-blue-100 text-blue-600...",
        // ...varian lainnya
      },
    },
    defaultVariants: {
      variant: "default",
    },
  }
);
```

Selain *Badge*, komponen dasar lain seperti *Switch* dikembangkan untuk kontrol biner (misalnya mengaktifkan /menonaktifkan fitur) dan menjadi elemen kontrol standar di seluruh aplikasi.

2. Komponen *Molecules*
   Molecules adalah gabungan dari beberapa *atoms* untuk membentuk unit fungsional yang lebih kompleks. Beberapa *molecules* kunci yang dikembangkan antara lain:
   - *Toast*
     Komponen notifikasi yang memberikan umpan balik visual non-intrusif kepada pengguna setelah melakukan suatu aksi (sukses, *error, info*).
   - *Copy Data*
     Menggabungkan teks/*link* dengan ikon 'salin'. Komponen ini memanfaatkan *molecule Toast* untuk memberi konfirmasi kepada pengguna bahwa data telah berhasil disalin ke *clipboard*.
   - *LabelStatus*
     Merupakan evolusi dari *atom Badge*, yang secara spesifik digunakan untuk menampilkan label status yang lebih kontekstual seperti *'Verified'* atau *'Warning'* dengan warna yang telah ditentukan.

Setelah *Atoms* telah dibangun selanjutnya membuat suatu *organisms*. *Organisms* dibangun dengan menggabungkan berbagai *molecules* dan/atau *atoms* menjadi sebuah bagian antarmuka yang fungsional dan mandiri.

1. *DatePickerWithRange*

```
return (
  <div className={cn("grid gap-2", className)}>
    <Popover>
      <PopoverTrigger asChild>
        <Button id="date" variant="outline">
          <span className={cn("justify-start text-left font-normal", date && "text-muted-foreground")}>
            <CalendarIcon className="mr-2 h-4 w-4" />
          </span>
          {shortcutText ? (
            <span>{shortcutText}</span>
          ) : (
            <span>{format(date.from, "LLL dd, yyyy")} - {format(date.to, "LLL dd, yyyy")}</span>
          )}
        </Button>
      </PopoverTrigger>
      <PopoverContent className="w-auto p-0" align="start">
        <div className="flex">
          <div className="flex flex-col items-start justify-start border-r-2 border-input text-gray-500">
            {shortcuts.map(shortcut => (
              <Button
                key={shortcut.label}
                className={cn('w-full justify-start rounded-none pr-10 pt-2 text-sm font-medium',
                  shortcutText === shortcut.label ? 'bg-blue-500 text-blue-600 hover:bg-blue-500' : 'text-[167928] hover:bg-gray-100')}
                variant="ghost"
                onClick={() => {
                  setSetShortcut(getRange())
                  setShortcut(shortcut.label)
                  setIsShortcut(true)
                  setCalendarMonth(shortcut.calendarUpdate)
                }}
              >
                {shortcut.label}
              </Button>
            ))}
          </div>
        </div>
        <p className="mt-auto cursor-pointer pb-2 pl-5 text-sm font-medium text-blue-600 hover:text-blue-500" onClick={() => {
```

```
          setDate(getThisMonthRange())
          setShortcutText('This Month')
          setIsShortcut(true)
          setCalendarMonth(today)
        }}>Reset</p>
      </PopoverContent>
    </Popover>
    <Calendar
      initialFocus mode="range" month={calendarMonth}
      selected={data}
      onMonthChange={(month) => setCalendarMonth(month)}
      onSelect={(selectedDate) => {
        setDate(selectedDate)
        setIsShortcut(false)
        setShortcutText('')
        if (selectedDate?.from) {
          setCalendarMonth(selectedDate.from)
        }
      }}
      numberOfMonths={2}
    />
  </div>
```

Organisme ini menyediakan fungsionalitas pemilihan rentang tanggal yang komprehensif. Komponen ini menggabungkan *atom Button* (sebagai pemicu), *molecule Calendar* (berbasis *react-day-picker* yang dikustomisasi), dan *Popover* untuk menampilkan antarmuka kalender. Fitur utamanya adalah penyediaan *shortcut* untuk rentang waktu umum (misal: "7 Hari Terakhir", "Bulan Ini"), yang menyederhanakan proses *filtering* data bagi pengguna.

2. *DropdownInfinite* dan *VirtualContainer*

```

const DropdownMenuComponent = ({ className, children, ...props }) => {
  return (
    <DropdownMenu {...props}>
      <DropdownMenuTrigger
        className="inline-flex items-center bg-primary text-primary-foreground rounded-md px-4 py-2"
      >
        {children}
      </DropdownMenuTrigger>
      <DropdownMenuContent className="overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-md">
        {children}
      </DropdownMenuContent>
    </DropdownMenu>
  )
}

DropdownMenuComponent.displayName = DropdownMenuPrimitive.displayName

const DropdownMenuItemComponent = ({ className, ...props }) => {
  return (
    <DropdownMenuItem
      className={cn('cursor-pointer select-none items-center rounded-sm px-2 py-1 text-sm outline-none transition-colors', className)}
      {...props}
    />
  )
}

DropdownMenuItemComponent.displayName = DropdownMenuPrimitive.Item.displayName

const DropdownMenuCheckboxItem = ({ className, checked, ...props }) => {
  return (
    <DropdownMenuCheckboxItem
```

```
      className={cn('cursor-pointer select-none items-center rounded-sm px-2 py-1 text-sm outline-none transition-colors', className)}
      checked={checked}
      {...props}
    />
  )
}

DropdownMenuCheckboxItem.displayName = DropdownMenuPrimitive.CheckboxItem.displayName
export { DropdownMenuComponent, DropdownMenuItemComponent, DropdownMenuCheckboxItem }
```

Untuk mengatasi tantangan performa saat menampilkan data dalam jumlah sangat besar pada elemen *dropdown*, dikembangkan organisme *DropdownInfinite*. Komponen ini mengimplementasikan dua teknik optimasi krusial:

- *Infinite Scroll*: Data tidak dimuat sekaligus, melainkan diambil dari *API* secara bertahap (*paginated*) saat pengguna menggulir ke bawah.
- *Virtualization*: Diterapkan melalui komponen *VirtualContainer*, teknik ini hanya me-*render* item yang terlihat di layar (*viewport*) pengguna. Hal ini secara drastis mengurangi jumlah elemen *DOM* yang harus dikelola oleh *browser*, sehingga menjaga antarmuka tetap responsif bahkan dengan ribuan data.

C. Slicing Page: Implementasi Antarmuka Pengguna

Tahap *slicing* mengubah desain *UI* menjadi komponen-komponen *React (TSX)* yang fungsional dan modular, dengan *styling* yang responsif menggunakan *Tailwind CSS*. Fokus utamanya adalah menciptakan komponen yang dapat digunakan kembali (*reusable*) dan konsisten di seluruh aplikasi.

- *Ticket Performance Dashboard*: Bagian ini menampilkan metrik kinerja tiket. Komponen utama, *TicketSection*, memeriksa hak akses pengguna melalui *useSession*. Jika diizinkan, komponen *TicketInfo* akan merender data seperti total tiket, tiket terselesaikan, dan waktu respons rata-rata. Setiap metrik ditampilkan dalam komponen *InfoCard* individual yang juga menangani *state* saat data sedang dimuat dengan menampilkan *LinearSkeleton*.
- *Detail Team Page*: Halaman ini menampilkan profil lengkap seorang anggota tim. Komponen ini menggunakan *useParams* untuk mendapatkan *ID* pengguna dari *URL* dan *useRouter* untuk navigasi. Informasi pengguna seperti nama, *email*, dan peran ditampilkan dalam *CardProfile*. Terdapat juga menu *Popover* yang berisi aksi administratif (seperti *EditRole*), yang hanya muncul jika pengguna yang melihat memiliki hak akses yang cukup (*isHaveAccess*).
- *Chat Widget - Appearance*: Halaman pengaturan tampilan *chat widget* ini menggunakan komponen *Accordion* untuk mengelompokkan berbagai opsi. Pengguna dapat mengubah tema warna, mengaktifkan/menonaktifkan fitur seperti logo dan foto agen menggunakan *Switch*, serta melihat pratinjau langsung dari perubahan yang mereka buat.
- *User Lists*: Halaman ini berfungsi untuk mengelola seluruh data pengguna dalam sistem. Halaman ini mengintegrasikan beberapa komponen fungsional: *filter* pencarian (*UserListsFilter*), *dialog* untuk menambah pengguna baru (*AddNewUser*), dan dialog untuk mengekspor data (*ExportUserList*). Data pengguna ditampilkan dalam sebuah tabel modular (*TemplateTableSelect*) yang mendukung paginasi dan penyaringan data secara dinamis. Logika untuk *filter* dikelola oleh *custom hook useUserListFilters* untuk menjaga kerapian kode.

```
import { useState, useCallback } from "react";
import PageTitle from "a/components/molecules/PageTitle";
import ExportUserList from "../components/actions/ExportUserList";
import AddNewUser from "../components/actions/AddNewUser";
import UserListsFilter from "./components/UserListsFilter";
import TemplateTableSelect from "a/components/organism/table/TemplateTableSelect";
import { useGetUser } from "../services/fetchUserList.service";
import UserListsTableColumns from "../components/table/UserListsTableColumns";
import { useUserListFilters } from "./hooks/useUserListFilters";

const ManageUserListsPage = () => {
  // Menggunakan custom hook untuk mengelola state filter
  const { filters, handleFilterChange, hasActiveFilters } = useUserListFilters();

  return (
    <div className="px-5 flex flex-col">
      <div className="w-full flex justify-between items-center py-5">
        <PageTitle />
        <div className="flex items-center gap-2">
          {/* Komponen untuk aksi ekspor dan tambah pengguna */}
```

```
          <ExportUserList filters={filters} />
          <AddNewUser />
        </div>
      </div>

      {/* Komponen untuk filter data tabel */}
      <UserListsFilter onChange={handleFilterChange} />

      <div className="mt-5 mb-3 w-full">
        {/* Komponen tabel yang dapat digunakan kembali */}
        <TemplateTableSelect
          dataFetchService={useGetUser}
          columns={UserListsTableColumns}
          filters={filters}
          noDataMessage="Tidak ada data"
          setIsFilter={hasActiveFilters()}
        />
      </div>
    </div>
  );
};

export default ManageUserListsPage;
```

- *Detail User*: Mirip dengan *Detail Team*, halaman ini menampilkan informasi spesifik satu pengguna. Halaman ini menggunakan *CardProfile* untuk info dasar dan *CardMember* untuk metadata seperti status verifikasi dan aktivitas terakhir. *FeaturesTable* juga digunakan untuk menampilkan daftar akses fitur dan *log* aktivitas pengguna.
- *Company/Group List*: Halaman ini berfungsi untuk menampilkan dan mengelola daftar grup perusahaan. Struktur halamannya mirip dengan *User Lists*, di mana ia menggunakan komponen *filter*, tabel, dan fungsi ekspor data yang relevan untuk entitas perusahaan.
- *Overview/Dashboard*: Ini adalah halaman dasbor utama yang menyajikan ringkasan data dan aktivitas. Halaman ini menggunakan *InfoCard* dan *UserCard* untuk menampilkan statistik kunci. Visualisasi data tren ditampilkan menggunakan grafik *LineAreaChart* dari *library ApexCharts*. Selain itu, terdapat tabel ringkasan aktivitas pengguna terbaru.

D. Integrasi API: Aliran Data Frontend-Backend

Integrasi *API* menjadi jembatan antara *frontend* dan *backend*, memastikan aliran data yang efisien dan *real-time*. Proses ini memanfaatkan *library React Query* untuk manajemen *server state* dan *Axios* untuk melakukan permintaan *HTTP*.

- *Custom Hooks* untuk *API* Terpusat: Logika permintaan *API* untuk setiap modul (misalnya, *User List* atau *Company/Group*) dienkapsulasi dalam *custom hooks* sendiri, seperti *useManageUserListApiRequest*. Pendekatan ini memisahkan logika *API* dari komponen *UI*, membuat kode lebih bersih, terorganisir, dan mudah diuji. *Hook* ini menangani semua operasi *CRUD* (*Create, Read, Update, Delete*) dan ekspor data.
- Manajemen *Server State* dengan *React Query*:
  - *useQuery*: Digunakan untuk mengambil data (operasi *GET*). *React Query* secara otomatis menangani *caching*, *refetching*, dan *stale time*, sehingga data yang ditampilkan selalu segar tanpa perlu memuat ulang halaman secara manual.
  - *useMutation*: Digunakan untuk operasi yang mengubah data di *server* (*POST, PATCH, DELETE*), seperti menambah pengguna (*useAddUser*) atau mengekspor file (*useExportUser*). *Hook* ini menyediakan *state* untuk *isLoading, isError*, dan *isSuccess*.
  - *useInfiniteQuery*: Diterapkan pada fitur yang memerlukan *infinite scrolling* atau paginasi, seperti pada daftar *log* aktivitas, untuk memuat data secara bertahap saat pengguna menggulir halaman.
- Penanganan *Error* dan Keamanan: Setiap permintaan *API* dibungkus dalam *blok try-catch* untuk menangani kegagalan. Pesan *error* yang relevan diekstrak dan dilempar ke komponen *UI*. *useAxiosAuth* digunakan sebagai *interceptor Axios* untuk menyematkan token autentikasi pada setiap permintaan secara otomatis.
- Validasi Data *Frontend*: Untuk memastikan integritas data sebelum dikirim ke *API*, *library Zod* digunakan untuk mendefinisikan skema validasi pada *form*, seperti pada *ChangeCompanyNameSchema*.

Contoh *source code* : *Hook API* dan Penggunaannya

1. *Hook API* Terpusat (*useManageUserListApiRequest.ts*)

```
// Kode Program 4.30 (disederhanakan)
import useAxiosAuth from "@/hooks/client/useAxiosAuth";

const useManageUserListApiRequest = () => {
  const api = useAxiosAuth();

  // Mengambil daftar pengguna dengan filter
  const getUsers = async (params: GetUserParams) => {
    const response = await api.get("/v1/users", { params });
    return response.data;
  };

  // Menambah pengguna baru
  const addUsers = async (payload: AddUserSchemaType) => {
    const response = await api.post("/v1/users", payload);
    return response;
  };

  // Mengekspor data pengguna
  const exportUsers = async (opts: ExportOptions) => {
    const response = await api.post("/v1/users/export", opts, { responseType: "blob" });

    return response.data;
  };

  return { getUsers, addUsers, exportUsers };
};
```

2. *Hook Service* dengan *React Query* (*fetchUserList.service.ts*)

```
// Kode Program 4.31 (disederhanakan)
import { useQuery } from "@tanstack/react-query";
import useManageUserListApiRequest from "../hooks/useManageUserListApiRequest";

export const useGetUser = (params: GetUserParams) => {
  const { getUsers } = useManageUserListApiRequest();

  return useQuery({
    queryKey: ["GetUser", params], // Kunci unik untuk caching
    queryFn: async () => {
      try {
        return await getUsers(params);
      } catch (error: any) {
        // Penanganan error terpusat
        throw { message: error.response?.data?.message || "Unexpected error" };
      }
    },
    staleTime: 2 * 60 * 1000, // Data dianggap fresh selama 2 menit
  });
};
```

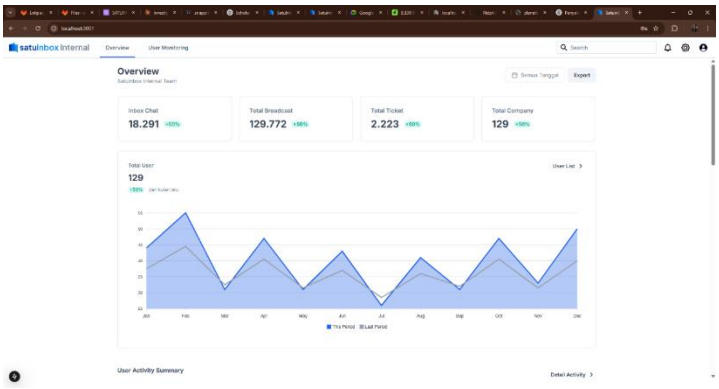
E. Produk yang dihasilkan

Pada bagian ini, akan dijelaskan produk yang dihasilkan selama pengembangan *platform* Satuinbox dan Satuinbox Internal. Fokus utamanya adalah komponen-komponen antarmuka pengguna (*UI*) yang fungsional, responsif, dan terintegrasi penuh dengan sistem *backend* untuk menyajikan data secara akurat. Hasil pengembangan mencakup berbagai elemen fungsional yang bertujuan mempermudah pengguna dalam mengelola komunikasi dan data operasional.

Produk utama yang dihasilkan adalah serangkaian halaman dan fitur yang terdefinisi dengan baik dalam *platform* Satuinbox Internal dan Klien. Halaman *login* menjadi gerbang keamanan utama, menampilkan antarmuka yang minimalis untuk proses autentikasi pengguna. Setelah masuk, pengguna disajikan dengan berbagai dasbor dan alat manajemen.

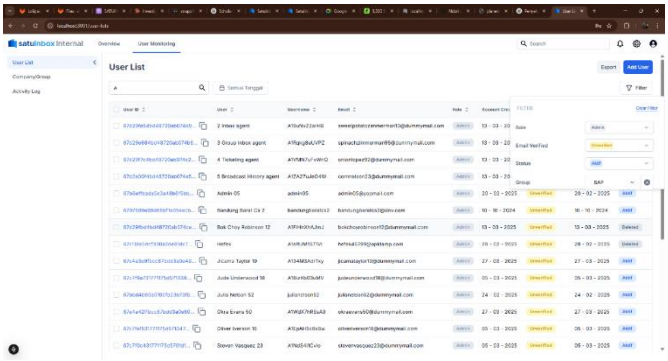
Untuk sisi klien, terdapat *Scoreboard Ticket Performance* yang menampilkan statistik penting seperti jumlah total tiket, tiket terselesaikan, dan yang belum terselesaikan, beserta waktu respons rata-rata. Fitur ini bersifat kondisional dan hanya muncul jika pengguna memiliki hak akses *ticketing*, memastikan relevansi tampilan sesuai peran pengguna. Selain itu, halaman *Detail User*

Team memberikan kemampuan bagi *Super Admin* untuk mengelola profil dan hak akses anggota tim secara mendetail, termasuk mengaktifkan atau menonaktifkan fitur *platform* seperti *Inbox, Broadcast,* dan *Ticketing* untuk pengguna tertentu.



Gambar 7. *Tampilan Halaman Overview SATUINBOX Internal*

Untuk *platform* internal, halaman *Overview* berfungsi sebagai dasbor utama yang memberikan gambaran umum kinerja sistem. Seperti yang ditunjukkan pada Gambar 7, halaman ini menyajikan metrik-metrik kunci seperti *total chat, broadcast*, tiket, dan perusahaan, lengkap dengan persentase perubahan untuk analisis tren. Di bawahnya, terdapat grafik visual yang membandingkan jumlah total pengguna aktif dari waktu ke waktu, memberikan evaluasi pertumbuhan *platform* yang jelas.



Gambar 8. *Tampilan Halaman User List SATUINBOX Internal*

Manajemen data menjadi inti dari *platform* internal. Halaman *User List*, yang divisualisasikan pada Gambar 8, menyajikan daftar lengkap pengguna yang terdaftar. Halaman ini tidak hanya menampilkan informasi dasar seperti *ID*, nama, dan *email*, tetapi juga dilengkapi dengan sistem *filter* yang kuat. Tim internal dapat menyaring pengguna berdasarkan peran (*Role*), status verifikasi *email*, status akun (Aktif/Dihapus), atau keanggotaan dalam grup tertentu, yang sangat vital untuk administrasi pengguna skala besar.

Selain manajemen pengguna, *platform* juga dilengkapi halaman *Company/Group List* untuk mengelola data perusahaan atau grup klien, menampilkan detail seperti jumlah anggota dan divisi. Untuk kebutuhan audit dan pemantauan, halaman *Activity Log* mencatat seluruh aktivitas pengguna secara rinci, mulai dari *login* hingga interaksi dengan fitur lainnya. *Log* ini dapat difilter berdasarkan rentang waktu atau jenis aktivitas, memberikan wawasan mendalam tentang pola penggunaan sistem.

Sebagai fitur pendukung yang krusial, platform menyediakan fungsi ekspor universal pada halaman-halaman yang padat data seperti *User List*, *Company/Group List*, dan *Activity Log*. Melalui sebuah *dialog modal*, pengguna dapat memilih untuk mengekspor data ke format *Excel*, *PDF*, atau *CSV* dengan rentang tanggal yang spesifik. Setelah proses selesai, sebuah notifikasi akan muncul yang mengonfirmasi bahwa *file* berhasil dibuat dan siap untuk diunduh, mempermudah analisis data dan pelaporan di luar *platform*.

Seluruh *platform* ini dibangun dengan desain yang responsif. Tampilan pada perangkat *mobile* telah dioptimalkan untuk memastikan semua fungsionalitas, mulai dari melihat grafik di halaman *Overview* hingga menggunakan *filter* di halaman *User List*, tetap dapat diakses dengan mudah dan intuitif, menjaga produktivitas pengguna di mana pun mereka berada.

V. SIMPULAN

Berdasarkan rumusan masalah yang dijabarkan dan analisis yang telah dilakukan, dapat disimpulkan bahwa pengembangan *frontend platform* SATUINBOX dan SATUINBOX Internal telah berhasil mencapai tujuannya secara menyeluruh. Proses

implementasi antarmuka pengguna melalui tahap *slicing page* berhasil diselesaikan secara optimal, menghasilkan tampilan yang sepenuhnya responsif dan intuitif, sesuai dengan rancangan *UI/UX* yang telah ditetapkan. Keberhasilan ini didukung penuh oleh integrasi *API* dengan sistem *backend* yang berjalan lancar dan efisien, memastikan semua fitur berfungsi dengan baik dan data yang disajikan selalu akurat. Lebih lanjut, dari sisi performa, aplikasi menunjukkan hasil yang sangat memuaskan. Berkat optimasi kode dan strategi pengelolaan data yang efektif, interaksi pengguna tetap mulus tanpa adanya *lag* atau *error* yang signifikan, bahkan ketika dihadapkan pada *volume* data yang besar sekalipun.

DAFTAR PUSTAKA

[1] V. Stray, "An Empirical Investigation of the Daily Stand-Up Meeting in Agile Software Development Projects," pp. 1-148, 2014.
[2] j. v.-t. c. E. mijacobs, "Apa yang dimaksud dengan Scrum?," Microsoft, 1 February 2024. [Online]. Available: https://learn.microsoft.com/id-id/devops/plan/what-is-scrum. [Diakses 2 May 2025].
[3] B. Frain, Responsive Web Design with HTML5 and CSS3, Packt Publishing Ltd, 2015, 2015.
[4] Microsoft, "TypeScript is JavaScript with syntax for types.," Microsoft, 2020. [Online]. Available: https://www.typescriptlang.org/. [Diakses 12 April 2025].
[5] Tailwind Labs, "Tailwind CSS: Utility-First CSS Framework," Tailwind Labs, 2023. [Online]. Available: https://v1.tailwindcss.com/. [Diakses 12 April 2025].
[6] J. J. ". Sarjeant., "Promise based HTTP client for the browser and node.js," Axios Project, 2020. [Online]. Available: https://github.com/axios/axios [Diakses 12 May 2025].
[7] TanStack, "TanStack Query: Powerful asynchronous state management for TS/JS, React, Solid, Vue, Svelte and Angular," TanStack, 2023. [Online]. Available: https://tanstack.com/query/latest. [Diakses 12 April 2025].
[8] Poimandres, "Zustand: A bear necessities state management solution for React," 2023. [Online]. 8. Available: https://github.com/pmndrs/zustand [Diakses 18 April 2025].
[9] GitLab Inc., "GitLab: The most-comprehensive AI-powered DevSecOps platform," GitLab Inc., 2023. [Online]. Available: https://about.gitlab.com/ [Diakses 12 April 2025].
[10] Google, "Chrome DevTools Overview," Chorme Developer, 2023. [Online]. Available: https://developer.chrome.com/docs/devtools?hl=id. [Diakses 12 April 2025].
[11] ByteDance Ltd., "Lark: Collaboration & Communication Platform," Lark, 2023. [Online]. Available: https://www.larksuite.com/en_sg [Diakses 12 April 2025].
[12] B. Frost, Atomic Design, Brad Frost, 2016. Available: https://atomicdesign.bradfrost.com/
[13] R. F. Augusdi, A. A. Yunanto, D. I. Permatasari dan A. F. Muhammad, "Development of Sandbox English Conversation Training Applications with Atomic Design," 2021. Available: https://www.researchgate.net/publication/356008303_Development_of_Sandbox_English_Conversation_Training_Applications_with_Atomic_Design
[14] P. G. Amores, "Planning, Designing, and Building a Customized UI Library: A Step-by-StepGuide for Improving Frontend Development.," pp. 19-192, 2024
[15] O. Bhavare, "Optimizing Web Performance: Tree Shaking Explained," Medium, 12 November 2023. [Online]. Available: https://medium.com/@omkarbhavare2406/optimizing-web-performance-tree-shaking-explained-007208163c88 [Diakses 15 April 2025].
[16] D. Ihnatovich, "Tree Shaking in JavaScript: A Simple Explanation for Beginners," Medium, October 2024. [Online]. Available: https://medium.com/@ignatovich.dm/tree-shaking-in-javascript-a-simple-explanation-for-beginners-7d7c85ad2e20. [Diakses 15 April 2025].