BAB I
PENDAHULUAN

1.1 Gambaran Umum

Perjalanan saya di dunia pengembangan perangkat lunak dimulai pada tahun 2018, dengan fokus serius pada *web development*, khususnya *Frontend Engineering*, sejak awal 2020. Saya aktif berkontribusi pada proyek *open source* dan mulai bekerja sebagai *freelance Frontend Engineer* paruh waktu pada akhir 2022 sambil berkuliah. Pengalaman ini mempercepat perkembangan saya, memungkinkan saya menangani kasus nyata seperti optimasi performa web, aksesibilitas, arsitektur *frontend*, dan mendalami teknologi seperti Next.js, Redux, serta manajemen proyek menggunakan Trello.

Saat ini, saya terus mengasah kemampuan teknis dan *soft skill* yang diperoleh. Target besar saya ke depan adalah bergabung dengan perusahaan ternama sebagai *Frontend Engineer* secara penuh waktu, dengan aspirasi jangka panjang untuk berkembang menjadi *Software Engineer* (SWE). Saya juga berencana memperdalam pengetahuan matematika untuk akhirnya menjadi seorang *Quant Developer*, sehingga dapat memberikan kontribsusi yang lebih luas dan berdampak signifikan di industri teknologi.

1.2 Rumusan Masalah

Dalam pengembangan dan kontribusi saya di bidang pengembangan web, khususnya pada proyek yang menggunakan Laravel dan NextJS, saya menghadapi sejumlah kendala teknis yang mempengaruhi efisiensi, performa, dan pengalaman pengguna. Tantangan utama berasal dari inkonsistensi arsitektur antara *landing page* dan *dashboard*. *Landing page* masih mengadopsi Laravel *template* dan jQuery dengan pendekatan *monolith*, sedangkan *dashboard* menggunakan NextJS dan Redux yang berbasis API. Selain itu, terdapat tiga *dashboard* dengan *codebase* dan domain terpisah namun identik, hanya berbeda pada beberapa komponen, *style* seperti warna, dan *endpoint* API. Kondisi ini menyulitkan integrasi fitur,

pemeliharaan kode, dan kolaborasi tim, sehingga memperlambat proses pengembangan.

Disisi lain, penggunaan versi lama Laravel dan NextJS menimbulkan risiko terhadap stabilitas dan keamanan sistem. Implementasi NextJS pada sisi tampilan masih menggunakan JavaScript tanpa memanfaatkan *Typescript*, padahal Typescript direkomendasikan untuk meminimalkan *bug* di *frontend* serta meningkatkan konsistensi dan kemudahan pemeliharaan *codebase*. Selain itu, performa *rendering* yang lambat dan waktu muat (*load time*) yang lama semakin memperburuk pengalaman pengguna.

1. Bagaimana cara mengatasi inkonsistensi arsitektur antara *landing page* (Laravel *template* & jQuery) dan *dashboard* (NextJS & Redux), serta mengelola tiga *dashboard* identik yang terpisah?
2. Apa dampak penggunaan teknologi usang (versi lama Laravel dan NextJS) serta absennya Typescript terhadap stabilitas, keamanan, performa, dan kemudahan pemeliharaan sistem?
3. Mengapa perbedaan stack tech dapat menghambat kinerja performa dalam load time dan menyebabkan technical debt ?

1.3 Batasan Masalah

Untuk menjaga laporan tetap terkendali dan terarah, berikut adalah batasan masalah yang membatasi ruang lingkup pembahasan. Batasan ini mencakup hal-hal yang akan dibahas dan tidak dibahas, sehingga cakupan laporan tidak terlalu luas.

1. Fokus pada teknologi web Laravel dan NextJS.
2. Lingkungan pengembangan terbatas pada dashboard dan landing page berbasis API.
3. Periode waktu pengembangan februari 2023 sampai dengan januari 2024.
4. Aspek performa terbatas pada rendering dan load time pada halaman landing page / marketing dan dashboard.

1.4 Tujuan

Penelitian dan pengembangan yang dilakukan dalam proyek ini bertujuan untuk memberikan kontribusi nyata dalam peningkatan efisiensi, performa dan pengalaman pengguna pada sistem web yang menggunakan teknologi PHP dengan framework Laravel dan framework NextJS dengan Typescript pada sisi tampilan. Tujuan utama yang ingin dicapai antara lain sebagai berikut:

1. **Menyatukan arsitektur landing page dan dashboard**

   Merancang dan mengimplementasikan migrasi terhadap arsitektur lama dari Laravel Template & JQuery pada Landing / Marketing page ke NextJS, sehingga landing page dan dashboard memiliki codebase yang tersinkronisasi, konsisten dan memanfaatkan penggunaan REST API secara maksimal.

2. **Mengelola Dashboard identik yang terpisah**

   Mengembangkan sistem yang memungkinkan pengelolaan dashboard dengan codebase yang sama namun dapat dikustomisasi dengan perbedaan komponen, style dan endpoint API, sehingga mengurangi redudansi dan memudahkan pemeliharaan dengan memanfaatkan teknologi turborepo dari Vercel.

3. **Memperbarui Teknologi ke versi terbaru**

   Melakukan upgrade versi Laravel dan NextJS ke versi terbaru untuk meningkatkan stabilitas, performa dan keamanan sistem, serta memanfaatkan fitur-fitur modern yang tersedia.

4. **Mengimplementasikan Typescript pada sisi Frontend**

   Mengubah implementasi NextJS dari Javascript ke Typescript untuk mengurangi risiko bug, meningkatkan konsistensi kode dan memudahkan pemeliharaan codebase.

5. **Meningkatkan performa rendering dan mengurangi load time**

   Mengoptimalkan performa *rendering* dengan menerapkan Server-Side Rendering (SSR) pada NextJS dan mengurangi waktu muat halaman untuk meningkatkan pengalaman pengguna.