PENGIMPLEMENTASIAN REACTJS DAN FIREBASE DALAM PENGEMBANGAN PLATFORM WAHDA

Disusun Oleh:

N a m a : Naufal Yusran

NIM : 19523170

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2023

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGIMPLEMENTASIAN REACTJS DAN FIREBASE DALAM PENGEMBANGAN PLATFORM WAHDA

TUGAS AKHIR

Disusun Oleh:

N a m a : Naufal Yusran

NIM : 19523170

Yogyakarta, 13 Oktober 2023

Pembimbing,

(Dr. Syarif Hidayat, S.Kom., M.I.T.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGIMPLEMENTASIAN REACTJS DAN FIREBASE DALAM PENGEMBANGAN PLATFORM WAHDA

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 30 Oktober 2023

Tim Penguji

Dr. Syarif Hidayat, S.Kom., M.I.T. _____

Anggota 1

Sheila Nurul Huda, S.Kom., M.Cs. _____

Anggota 2

Dr. Sri Kusumadewi, S.Si., M.T. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Naufal Yusran

NIM : 19523170

Tugas akhir dengan judul:

PENGIMPLEMENTASIAN REACTJS DAN FIREBASE DALAM PENGEMBANGAN PLATFORM WAHDA

Menyatakan bahwa seluruh *component* dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 22 Oktober 2023

(Naufal Yusran)

HALAMAN PERSEMBAHAN

Laporan akhir ini saya tuliskan sebagai bukti perjuangan saya dan penuh dengan rasa terima kasih saya persembahkan kepada keluarga, khususnya orang tua saya. Dengan doa dan cinta tanpa syarat dari mereka berikan kepada saya selama perjalanan studi saya selama ini, Saya ingin mengucapkan terima kasih sebesar-besarnya. Dukungan dan dorongan dari mereka lah yang menjadi sumber inspirasi yang tak membuat saya tidak menyerah dalam mengerjakan tugas akhir ini. Sekali lagi, saya ucapkan terima kasih atas segala pengorbanan mereka yang telah membantu saya mencapai jenjang akhir studi saya selama ini.

Bapak Selamat Yanuardi, S.E dan Ibu Gustiati, S.Pd

Terima kasih yang sebesar-besarnya kepada kedua orang tua penulis yang selalu memberikan dukungan baik berupa material maupun imaterial seperti nasihat dan doa, sehingga penulis dapat menyelesaikan tugas akhir dan menjalani masa studi dengan maksimal dan bertanggung jawab.

Ahmed Reza Falevi, S.T dan Vania Azalia Putri

Terima kasih kepada Bang Eja dan Adik Vania selaku saudara atas segala dukungan untuk menjalankan kuliah dan mengerjakan tugas akhir dengan sungguh-sungguh.

HALAMAN MOTO

“Most good programmers do programming not because they expect to get paid or get adulation by the public, but because it is fun to program.”

- Linus Torvalds

“Benevolent dictator? No, I'm just lazy. I try to manage by not making decisions and letting things occur naturally. That's when you get the best results”

- Linus Torvalds

“You won't get sued for anticompetitive behavior.”

- Linus Torvalds

KATA PENGANTAR

Assalamu'alaikum Wr. Wb

Segala puji hanya bagi Allah Swt, pencipta langit dan bumi, yang telah memberikan petunjuk dan anugerah-Nya kepada penulis sehingga penulis berhasil menyelesaikan laporan akhir berjudul "Pengimplementasian ReactJS dan Firebase dalam pengembangan platform Wahda". Laporan akhir ini dituliskan sebagai bagian dari penyelesaian tugas akhir saya melalui jalur rintisan bisnis, serta untuk memenuhi persyaratan dalam meraih gelar sarjana di Program Studi Informatika Universitas Islam Indonesia.

Dalam proses penyusunan laporan akhir ini, tidak tidak lepas dari bimbingan, arahan, dan dukungan yang diberikan oleh berbagai pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih kepada:

1. Allah SWT yang telah melimpahkan rahmat, taufiq, hidayah, dan inayah sehingga penulis dapat menyelesaikan laporan tugas akhir ini.
2. Orang tua serta keluarga tercinta yang telah dan selalu memberikan dukungan baik berupa materi maupun moral..
3. Rekan-rekan rintisan bisnis saya, Tim Juggernut, Muhammad Khoirul Umamil Achyar dan Sulistio yang telah berjuang dan berusaha bersama dalam mengerjakan tugas akhir.
4. Bapak Prof. Fathul Wahid, S.T., M.Sc. Ph. D selaku Rektor Universitas Islam Indonesia.
5. Bapak Prof. Dr. Hari Purnomo., M.T., IPU., ASEAN, Eng., selaku dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
6. Bapak Dr. Syarif Hidayat, S.Kom., M.I.T., sebagai dosen pembimbing kami, yang telah memberikan bimbingan tanpa kenal lelah kepada kami..
7. Bapak Dhomas Hatta Fudholi, selaku Ketua Program Studi Informatika Program Sarjana, Fakultas Teknologi Industri, Universitas Islam Indonesia.
8. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku ketua jurusan Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
9. Seluruh dosen dan tendik di Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia, atas arahan dan pengetahuan yang telah diberikan sepanjang masa studi saya.
10. Kepada teman-teman dan kerabat seperjuangan saya selama masa studi di Informatika Universitas Islam Indonesia.

11. Pemilik NIM 19523064 yang telah membantu, memberikan kritik, saran, dan dukungan kepada penulis dalam rangka menyelesaikan tugas akhir ini.
12. Seluruh pihak yang tidak dapat disebutkan satu per satu, terima kasih atas segala bantuan yang telah diberikan dalam bentuk apapun.

Laporan akhir ini kemungkinan masih memiliki banyak kekurangan yang perlu diperbaiki. Setiap kritik dan saran akan diterima dengan baik oleh penulis. Penulis berharap laporan ini dapat memberikan manfaat bagi para pembaca yang memiliki permasalahan yang sama dengan penulis hadapi. *Aamiin Ya Rabbal'alamin*

Yogyakarta, 22 Oktober 2023

(Naufal Yusran)

SARI

Peningkatan minat masyarakat terhadap pembelajaran pemrograman dalam beberapa tahun terakhir telah mendorong munculnya berbagai platform pembelajaran online. Studi yang melibatkan 1850 siswa di Asia Pasifik, termasuk Indonesia, menunjukkan bahwa mayoritas siswa menyadari manfaat dan potensi coding untuk masa depan mereka. Kebutuhan tenaga kerja di sektor Teknologi, Informasi, dan Komunikasi (TIK) yang terus meningkat juga memperkuat minat ini. Namun, jumlah lulusan TIK di Indonesia masih jauh dari permintaan pasar.

Untuk mengatasi tantangan dalam pembelajaran pemrograman, tim perintis bisnis dari Universitas Islam Indonesia, tim Jaggernut, berinisiasi menciptakan platform Wahda. Wahda membantu pengguna dengan memberikan akses langsung kepada mentor melalui obrolan dan konferensi video, memungkinkan pengguna untuk mengajukan pertanyaan dengan lebih leluasa dengan waktu respons yang lebih cepat dibandingkan fitur diskusi atau tanya-jawab pada platform pembelajaran online.

Platform Wahda terbagi menjadi dua aplikasi, yaitu frontend dan backend. ReactJS digunakan pada sisi frontend untuk membangun antarmuka pengguna yang responsif dan menarik. Sementara pada sisi backend, Firebase digunakan untuk mempermudah pengembangan aplikasi tanpa harus banyak memikirkan urusan backend.

Kedua teknologi ini memungkinkan percepatan pengembangan aplikasi, memperbaiki efisiensi, skalabilitas, dan kualitas pengalaman pengguna. Ini memberikan peluang lebih besar untuk mencapai kesuksesan di tengah persaingan pasar yang semakin ketat.

Kata kunci: Wahda, Waterfall, ReactJs, Firebase, Konsultasi Online, Pemrograman.

GLOSARIUM

API	Singkatan dari Application Programing Interface dan merupakan sebuah kumpulan aturan, protokol, dan alat yang digunakan sebagai jalur komunikasi antar perangkat lunak.
BaaS	Singkatan dari *backend-as-a-service* yang merupakan layanan komputasi *cloud* yang menyediakan infrastruktur backend yang siap dipakai melalui sebuah API.
Backend	Merupakan bagian dari program yang bertugas untuk pemrosesan atau pengolahan data serta penanganan basis data.
Callback	Sebuah fungsi yang diberikan kepada fungsi lain sebagai parameter.
Filter	Proses untuk memisahkan, menyaring, atau mengurutkan data, informasi, atau elemen berdasarkan kriteria tertentu.
Form	Bagian dari antarmuka pengguna yang digunakan untuk menerima input dari pengguna, seperti pengisian teks atau pilihan.
Frontend	Bagian dari sebuah program yang menampilkan antarmuka kepada pengguna untuk dilakukannya sebuah interaksi.
Hooks	Fungsi yang digunakan untuk memisahkan suatu fungsionalitas yang dapat digunakan kembali pada komponen fungsional ReactJs.
Library	Kumpulan kode yang telah ditulis sebelumnya dan dapat digunakan ulang untuk membantu pengembang membangun aplikasi atau sistem dengan lebih efisien.
Startup	Perusahaan rinitan yang umumnya bergerak di sektor teknologi dan memiliki pertumbuhan cepat dan keberhasilan jangka panjang.
Use case	Deskripsi singkat tentang bagaimana suatu aplikasi atau sistem digunakan untuk menyelesaikan masalah atau memenuhi kebutuhan pengguna.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Peningkatan atas minatnya pembelajaran pemrograman atau *coding* terus meningkat pada beberapa tahun terakhir. Berdasarkan survei yang dilakukan oleh Microsoft YouthSpark yang melibatkan 1850 siswa dengan usia di bawah 24 tahun yang berasal dari 8 negara di Asia Pasifik, termasuk Indonesia, ditemukan bahwa mayoritas siswa menyadari manfaat *coding* dalam pendidikan dan besarnya potensi yang ditawarkan *coding* bagi masa depan mereka. Sebesar 91% siswa di Indonesia ingin mengetahui lebih banyak lagi mengenai *coding*, sementara 72% siswa berharap *coding* dapat dijadikan sebagai mata pelajaran utama di sekolah. Studi ini juga menyebut sebanyak 74% siswa mengatakan bahwa *coding* penting untuk masa depan mereka, dan 88% setuju bahwa *coding* merupakan hal yang relevan bagi semua karir di masa depan, apapun bidang spesialisasi yang mereka tekuni (Prihadi, 2015).

Perkembangan minatnya pembelajaran atas pemrograman turut didukung juga oleh tingginya kebutuhan tenaga kerja di sektor teknologi, informasi, dan komunikasi (TIK). Kementerian Ketenagakerjaan (Kemnaker) memperkirakan, kebutuhan tenaga kerja di TIK terus meningkat sejak 2022 hingga 2025. Berdasarkan proteksi yang mereka lakukan mengenai kebutuhan tenaga kerja di sektor TIK sebanyak 1,49 juta orang pada 2023 jumlahnya diperkirakan naik 16% menjadi 1,73 juta orang pada 2024 dan meningkat 32% menjadi 1,97 juta pada tahun 2025. Meski demikian, jumlah talenta TIK di tanah air masih jauh dari permintaan, baik secara kualitas maupun kuantitas. Pada tahun 2020, Badan Litbang SDM Kementerian Komunikasi dan Informatika (Kemenkominfo) memperkirakan hanya ada 430,000 lulusan TIK di Indonesia (Rizaty, 2022).

Dengan bertambahnya minat masyarakat dalam mempelajari pemrograman dan permintaan pasar yang terus berkembang, beberapa *startup* yang bergerak pada bidang pembelajaran mengenai pemrograman secara *online* bermunculan sebagai solusi bagi mereka yang ingin mempelajari pemrograman serta menawarkan layanan bisnis bagi suatu perusahan untuk melatih karyawan mereka dalam mempelajari teknologi informasi. Hal ini mencerminkan pergeseran signifikan adalah cara individu memilih untuk memperoleh pengetahuan dan keterampilan pemrograman. Platform pembelajaran online memberikan akses

yang lebih mudah, fleksibel, dna beragam materi pembelajaran bagi mereka yang ingin memahami dan menguasai pemrograman.

Bermunculannya *platform* pembelajaran ini tentunya bukanlah menjadi hal yang menyelesaikan seluruh masalah dalam mempelajari pemrograman. Ketika seseorang melakukan pemrograman atau mempelajari pemrograman, berbagai masalah dapat muncul. Masalah-masalah ini sering kali berhubungan dengan kesalahan sintaksis dan kesalahan dalam menerapkan logika pemrograman (Syamsudin, 2020). Dalam menanggapi tantangan ini, platform pembelajaran online menyediakan fitur seperti forum diskusi dan tanya-jawab. Namun, beberapa fitur ini memiliki kendala yang dapat mempengaruhi pengalaman belajar pengguna, antara lain:

a) Pertanyaan yang tidak cepat dijawab oleh mentor; hal ini disebabkan oleh tingginya jumlah pertanyaan pada platform pembelajaran online dan kadang-kadang pertanyaan tidak dijawab karena alasan-alasan tertentu, seperti pertanyaan yang dianggap klise.
b) Keterbatasan pemahaman mendalam; fitur forum diskusi dan tanya-jawab umumnya berbentuk teks, sehingga penyampaian jawaban oleh *mentor* kepada pengguna sulit untuk dipahami.
c) Kendala teknis merupakan bagian yang tak terpisahkan dari pembelajaran pemrograman, di mana sering kali melibatkan perangkat keras dan perangkat lunak tertentu yang dapat mengalami berbagai masalah teknis.

Dalam menanggapi tantangan ini, tim perintis bisnis dari Universitas Islam Indonesia, yang dikenal sebagai tim Jaggernut, menciptakan situs Wahda sebagai solusi atas permasalahan yang ditemui dalam platform pembelajaran pemrograman *online*. Wahda membantu pengguna dengan memberikan akses langsung kepada mentor melalui obrolan dan konferensi video, memungkinkan pengguna untuk mengajukan pertanyaan dengan lebih leluasa dengan waktu respons yang lebih cepat dibandingkan fitur diskusi atau tanya-jawab pada platform pembelajaran *online*.

Dalam pengembangannya, platform Wahda dibagi menjadi dua aplikasi. Dua aplikasi terpisah tersebut yaitu *frontend* dan *backend*. *Frontend* merupakan aplikasi yang memberikan tampilan aplikasi yang interaktif kepada pengguna, serta biasanya disebut *client-side*. Sedangkan, aplikasi pada sisi *backend* memiliki tanggung jawab untuk memastikan sisi *frontend* bekerja dengan lancar dengan cara memberikan dukungan data yang sesuai dengan kebutuhan yang diperlukan pada sisi *frontend*. Aplikasi *backend* memiliki peran penting dalam

menjalankan komunikasi antar sistem yang berbeda dan sebagai tempat untuk menerapkan modul-modul ilmiah, seperti *machine learning*, *deep learning*, dan lain-lain. Istilah lain untuk aplikasi di sisi *backend* adalah *server-side*. (Singhal, 2023).

Pada sisi *frontend*, platform Wahda menggunakan ReactJs. ReactJS, yang merupakan library JavaScript *frontend* yang populer, telah terbukti efektif digunakan dalam membangun antarmuka pengguna yang responsif dan menarik serta banyaknya dukungan di komunitas. Konsep komponenisasi yang diterapkan ReactJs telah mengefektifkan pembuatan elemen antarmuka karena reusability dari *component* yang sudah dibuat dapat digunakan kembali. Hal ini tentunya dapat menghemat waktu pengembangan *startup* yang membutuhkan kecepatan pengembangan sehingga peluang *go to market* lebih dapat dijangkau lebih cepat (Nasution & Iswari, 2021).

Sementara itu, layanan yang ada pada Firebase digunakan pada sisi *backend platform* Wahda. Firebase merupakan layanan *backend-as-a-service* yang dikembangkan oleh Google untuk memberikan kemudahan para *developer* dalam mengembangkan aplikasi. Dengan menggunakan Firebase, *developer* bisa fokus dalam pengembangan aplikasi tanpa memberikan *effort* yang besar dalam urusan *backend* (Dicoding, 2020). Firebase memiliki beberapa layanan yang berguna dalam pengembangan platform Wahda, seperti Firebase Authentication, Firestore Database, dan Firebase Hosting.

Dengan memanfaatkan kedua *core technology* ini, pengembangan aplikasi dapat dipercepat. Selain itu, pemahaman mendalam terhadap manfaat dan tantangan yang terlibat dalam pemanfaatan kedua teknologi ini memungkinkan para pengembang untuk meningkatkan efisiensi, skalabilitas, dan kualitas pengalaman pengguna. Hal ini membantu mencapai kesuksesan di tengah persaingan pasar yang semakin ketat.

1.2 Rumusan Masalah

Dengan melihat penjelasan pada bagian latar belakang dari pengembangan *startup* yang dilakukan, rumusan masalah dalam tulisan ilmiah ini adalah bagaimana cara menggunakan ReactJs dan Firebase dalam mempercepat pengembagan aplikasi dan meningkatkan efisiensi serta kualitas pengalaman pengguna pada platform Wahda sebagai platform konsultasi pembelajaran pemrograman online. Penulisan juga akan mendemonstrasikan kemudahan apa saja yang didapatkan pengembang dalam mengembangkan fitur kompleks seperti live-chat dengan memanfaatkan layanan yang diberikan Firebase dan ReactJs dengan mudah dibandingkan tidak menggunakannya sama sekali.

1.3 Batasan Masalah

Batasan masalah yang terdapat pada penelitian ini adalah sebagai berikut:

a) Pengguna yang dilibatkan pada situs *platform* Wahda adalah mahasiswa Informatika Universitas Islam Indonesia.
b) Metode yang digunakan dalam pengembangan perangkat lunak untuk tugas akhir ini adalah dengan menggunakan metode Waterfall.
c) Pada tahap pengujian yang akan dilakukan dalam pengerjaan tugas akhir ini menggunakan *Black Box Testing*.
d) Hasil dari pengembangan dalam pengerjaan tugas akhir ini berupa website kosultasi pemrograman online yang hanya mampu dijalankan menggunakan *browser*.

1.4 Tujuan

Tujuan pengembangan platform Wahda pada pengembangan *startup* yang dilakukan adalah:

a) Mengidentifikasi metode optimal untuk memanfaatkan ReactJs dalam mempercepat pengembangan aplikasi platform Wahda.
b) Menemukan pendekatan yang efektif untuk memanfaatkan Firebase dalam meningkatkan efisiensi pada sisi backend dan memperbaiki kualitas pengalaman pengguna pada platform Wahda.
c) Meningkatkan kualitas platform Wahda sebagai platform konsultasi pembelajaran pemrograman online dengan mengintegrasikan ReactJs dan Firebase secara efisien dan efektif.

Dengan tujuan ini, diharapkan pengembangan platform Wahda menggunakan teknologi ReactJs dan Firebase dapat memberikan manfaat maksimal bagi pengguna serta mempercepat pertumbuhan dan kualitas platform konsultasi pembelajaran pemrograman tersebut.

1.5 Manfaat

Adapun manfaat dari pengembangan platform Wahda, di antaranya:

a) Memberikan wawasan mengenai pengembangan yang lebih efisien dengan menggunakan ReactJs dan Firebase dalam pengembangan aplikasi Wahda sehingga dapat meningkatkan produktivitas dan mengurangi waktu pengembangan.

b) Memberikan pemahaman atas kemampuan ReactsJs dalam membangun antarmuka dengan kode program yang modular dan *maintainable* serta layanan Firebase yang dapat memenuhi kebutuhan *backend* dalam pengembangan aplikasi untuk pembuatan *startup*.
c) Memberikan layanan berupa platform konsultasi yang dapat digunakan sebagai ruang tanya jawab bebas untuk saling berbagi ilmu pemrograman satu sama lain.

1.6 Metode Penelitian

Metode yang digunakan dalam pemgembangan *platform* Wahda adalah metode Waterfall. Tahapan-tahapan yang terdapat pada metode ini meliputi:

1.6.1 Analisis Kebutuhan

Analisis kebutuhan merupakan tahapan krusial dalam pengembangan perangkat lunak. Pada tahap ini, dilakukan pengumpulan data untuk memahami secara mendalam perangkat lunak yang diinginkan dan batasan-batasannya. Informasi diperoleh melalui berbagai metode seperti wawancara, diskusi, atau survei langsung dengan pengguna. Tahapan ini membantu membentuk landasan yang kuat untuk perancangan dan pengembangan perangkat lunak yang sesuai dengan harapan pengguna serta memenuhi kebutuhan yang ada.

1.6.2 Desain

Pada tahapan desain dalam metode waterfall, dilakukan pembuatan rancangan antarmuka dari sistem yang akan dibangun agar memudahkan pengembang dalam pengembangan perangkat lunak. Desain antarmuka bertujuan untuk memastikan bahwa pengguna akan memiliki pengalaman penggunaan yang optimal dan intuitif, sementara rancangan struktur internal bertujuan untuk memudahkan pengembangan dan pemeliharaan perangkat lunak.

1.6.3 Implementasi

Pada tahapan implementasi dalam metode waterfall, pengembang menerjemahkan rancangan sistem yang telah dibuat sebelumnya menjadi kode-kode program yang dapat dieksekusi. Rancangan yang ada pada tahap desain menjadi dasar pembuatan program-program kecil yang disebut *unit.* Hasil dari tahapan implementasi adalah sejumlah *unit* program yang telah diuji dan diintegrasikan secara komprehensif. Unit-unit ini membentuk dasar untuk langkah selanjutnya dalam metode waterfall, yaitu pengujian dan integrasi sistem secara

keseluruhan. Tahapan implementasi yang baik adalah landasan penting untuk memastikan kualitas dan konsistensi seluruh perangkat lunak yang sedang dikembangkan.

1.6.4 Testing

Pada tahapan ini, dilakukan pengujian pada dengan metode *black box*. Tujuan dari tahap ini adalah memastikan kinerja perangkat lunak apakah sudah sesuai dengan keinginan atau belum, serta membantu pengembang dalam mendeteksi *bug* dan mencegah terjadinya *error*.

1.7 Sistem Penulisan

Sistematika penulisan yang terdapat pada tugas akhir ini terdiri dari lima bab. Berikut merupakan penjelasan singkat mengenai kelima bab tersebut:

BAB I. PENDAHULUAN

Bagian ini merupakan penjelasan mengenai latar belakang pengerjaan tugas akhir, rumusan masalah, batasan masalah, tujuan dan manfaat penelitian, dan sistematika penulisan.

BAB II. LANDASAN TEORI

Bab ini memuat penjelasan teori-teori yang menjadi acuan dan terkait dengan studi tentang informasi sampah serta pengembangan aplikasi terkait.

BAB III. METODOLOGI

Bab ini membahas urutan tahapan dalam pengembangan aplikasi dengan metode waterfall, mencakup analisis kebutuhan sistem, perancangan sistem, pembuatan desain sistem dan prototipe, pengujian dan evaluasi prototipe, serta pengembangan sistem dan pengujian sistem.

BAB IV. HASIL DAN PEMBAHASAN

Bab ini membahas secara terperinci tentang output yang dihasilkan dari metode penelitian yang telah dilaksanakan. Hasil penelitian ini mencakup analisis tentang implementasi antarmuka aplikasi dan hasil uji coba aplikasi berdasarkan pengembangan yang telah dilakukan.

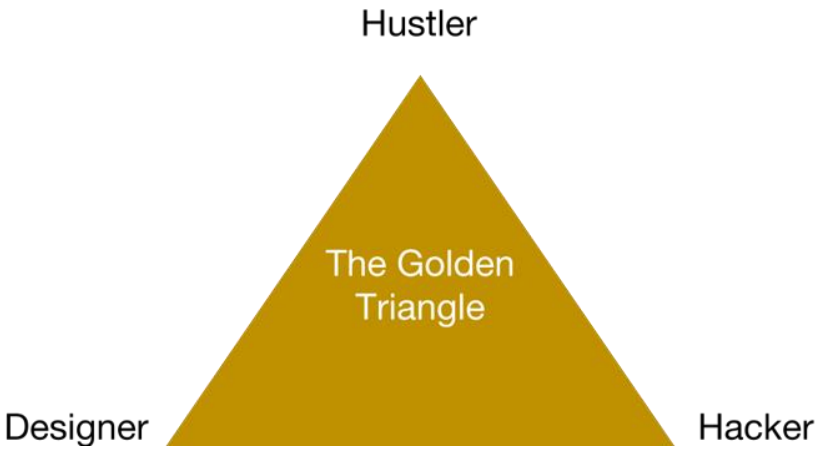
BAB V. PENUTUP

Bab ini mencakup ringkasan kesimpulan yang diperoleh dari hasil penelitian dan diskusi yang telah dijabarkan. Selain itu, bab ini juga menyertakan rekomendasi yang berupa saran atau solusi berdasarkan analisis peneliti terhadap hasil dan pembahasan yang telah dipresentasikan.

BAB II
LANDASAN TEORI

2.1 Startup

Startup atau perusahaan rintisan adalah perusahaan yang baru dibentuk tanpa riwayat operasional dan cepat dalam menghasilkan produk berbasis teknologi. Perusahaan-perusahaan ini mengembangkan perangkat lunak dalam kondisi yang sangat tidak pasti, menghadapi pasar yang berkembang pesat dengan keterbatasan sumber daya yang dimilikinya. Berdasarkan alasan itu, fokus utama dari *startup* adalah memberikan produk kepada pelanggan secepatnya, menghadapi banyak tekanan dari berbagai *stakeholder*, seperti *client*, investor, partner, yang mempengaruhi pengambilan keputusan dalam perusahaan, menghadapi dinamika teknologi dan pasar yang dihadapi sehingga mendorong perusahaan dalam penggunaan teknologi yang disruptif untuk dapat memasuki pasar yang berpotensi tinggi (Paternoster, Giardino, Unterkalmsteiner, Gorschek, & Abrahamsson, 2014).



Gambar 2.1 *Startup Golden Triangle*

Sumber: susbershop.com (2015)

Dalam menginisiasi sebuah *startup* pada praktiknya dibentuk sebuah tim yang memiliki tim yang dijuluki *The Golden Triangle*, yaitu tiga peran utama yang terdiri dari *Hustler*, *Hipster*, dan *Hacker* (Thomas, 2014).

2.1.1 Hustler

Fokus dari peran ini adalah pada aspek pengembangan bisnis dari suatu *startup*. Selain menerapkan inovasi teknologi yang mungkin menjadi kunci sukses bagi *startup*, membangun bisnis yang kuat juga merupakan hal yang tak kalah penting sebagai fondasi bagi *startup* untuk memasuki pasar. Secara rinci, seorang hustler akan memusatkan perhatian pada lima elemen, termasuk pengembangan pelanggan, manajemen produk, penjualan, blogging, dan perekrutan.

2.1.2 Hipster

Peran ini menekankan aspek visual dalam produk dan pemasaran. Di sisi pemasaran, hipster harus memiliki kemampuan untuk menciptakan citra menarik untuk startup. Selain itu, di sisi produk, hipster harus dapat mengintegrasikan semua kebutuhan pengguna ke dalam *User Experience dan User Interface* (UI/UX). Hal-hal yang harus difokuskan oleh hipster mencakup *landing page*, desain produk, pengujian desain, dan *product branding*.

2.1.3 Hacker

Peran ini bertanggung jawab pada implementasi teknologi dalam produk startup. Seorang hacker terlibat dalam pemrograman dan mengikuti perkembangan teknologi. Fokus utama hacker adalah mewujudkan produk.

2.2 Wahda

Wahda adalah sebuah *platform* konsultasi pemrograman *online* yang diinisiasi oleh tim perintisan bisnis, tim Jaggernut. Tim Jaggernut beranggotakan Sulistio yang berperan sebagai *Hustler*, Naufal Yusran yang berperan sebagai *Hacker*, dan Muhammad Khoirul Umamil Achyar yang berperan sebagai *Hipster*. Wahda dikembangkan dikarenakan melihat kondisi lingkungan sekitar yang membutuhkan platform sebagai tempat bertanya-jawab terkait permasalahan terkait pemrograman dan membutuhkan solusi langsung dari orang yang sudah ahli dalam bidang terkait. Sebelum melakukan pengembangan *platform* Wahda, Tim Jaggernut telah melihat beberapa solusi-solusi yang sudah ada saat ini untuk mengatasi masalah tersebut, namun setelah ditelusuri lebih lanjut, solusi-solusi tersebut masih belum cukup untuk menangani masalah yang ada.

Melihat kondisi tersebut, tim Jaggernut menginisiasi *platform* bagi mahasiswa Informatika UII untuk melakukan konsultasi dan berbagi pengetahuan tentang pemrograman. Mahasiswa dapat menggunakan layanan konsultasi, termasuk pesan chat, video konferensi, dan

forum diskusi. Dalam layanan konsultasi, pengguna dibimbing oleh programmer yang tersedia di *platform*. Semua *mentor* di platform Wahda adalah praktisi pemrograman dengan pengalaman di industri teknologi informasi. Selain konsultasi, pengguna dapat memanfaatkan forum diskusi untuk berbagi pengetahuan tentang pemrograman.

2.3 Pemrograman

Pemrograman merujuk pada proses teknologi untuk memberitahu komputer instruksi yang harus dilakukan dalam memecahkan masalah. Pemrograman dapat dipandang sebagai kolaborasi antara manusia dan komputer, di mana manusia membuat instruksi untuk diikuti oleh komputer berupa sebuah kode program atau sebuah bahasa yang dapat dimengerti oleh komputer Tujuan dari pemrograman adalah untuk menemukan rangkaian instruksi yang akan mengotomatisasi kinerja tugas pada komputer, seringkali untuk memecahkan suatu masalah yang diberikan (Coursera, 2023).

2.4 Metode Waterfall

Metode Waterfall merupakan salah satu *Software Development Life Cycle* (SDLC) dimana aktivitas pengembangan perangkat lunak dimulai dari spesification, development, validation, dan evolution lalu membaginya menjadi fase proses seperti spesifikasi kebutuhan, rancangan perangkat lunak, implementasi, pengujian, dan lain-lain. Kelebihan dari metode ini adalah mudah dipahami, *milestone* dari sebuah proyek dapat dipahami dengan baik, *requirement* akan menjadi stabil, dan menyediakan struktur yang jelas untuk pemula (Masyruhatin, Mursityo, & Pramono, 2019). Metode Waterfall memiliki lima tahapan dalam pengembangannya, yaitu requirement analysis atau analisa kebutuhan, system design atau desain sistem, *implementation*, *integration and testing*, *operation and maintenance*. Metode Waterfall menitikberatkan pada perencanaan yang terinci dan dokumentasi yang jelas. Hasil dari setiap tahap menjadi dasar untuk tahap berikutnya. Metode ini sesuai untuk proyek dengan persyaratan yang konsisten dan jelas, di mana tidak ada perubahan besar yang diantisipasi selama pengembangan.

2.5 ReactJs

ReactJs adalah sebuah *framework* pada bahasa pemrograman Javascript yang dikembangkan oleh Facebook. Digunakan untuk membangun antarmuka pengguna interaktif dan aplikasi web dengan cepat dan efisien dengan kode yang jauh lebih sedikit dibandingkan

dengan menggunakan JavaScript biasa. ReactJs menerapkan konsep *component* dalam pengembangan aplikasi. Dengan menggunakan konsep ini, *component* yang sebelumnya telah dibuat dapat digunakan kembali layaknya blok Lego independen (Herbert, 2022). Selain *component,* ReactJs juga memiliki beberapa istilah lain seperti *state*, *context*, dan *prop* atau *property*.

2.5.1 Component

Component merupakan istilah pada ReactsJs untuk bagian-bagian individual dari antarmuka, yang, ketika dirangkai membentuk seluruh antarmuka pengguna aplikasi. *Component* ini memungkinkan pengembang untuk membagi antarmuka pengguna (UI) menjadi bagian-bagian independen dan dapat digunakan kembali, serta memikirkan setiap bagian secara terpisah. Dengan begitu, kode program yang dibuat akan tampak lebih modular dibandingkan menggunakan *plain* Javascript.

2.5.2 State

Pada framework ReactJs, state adalah objek bawaan yang digunakan untuk menyimpan data atau informasi tentang *component*. State pada suatu *component* dapat berubah seiring berjalannya waktu; setiap kali terjadi perubahan, *component* akan melakukan pengulangan render. Perubahan pada state dapat terjadi sebagai respons terhadap tindakan pengguna atau peristiwa yang dihasilkan oleh sistem, dan perubahan tersebut mempengaruhi perilaku *component* dan bagaimana *component* tersebut akan di-*render* (Sufiyan, 2023).

2.5.3 Property

Property atau *prop* adalah cara untuk mengirim data dari *parent component* ke *child component* di ReactJs. Pada *property*, informasi akan disimpan dan diteruskan dari satu *component* ke *component* lain sebagai argumen. *Property* ini bersifat *read-only* di *child component*, artinya *child component* tidak dapat mengubah properti yang diterimanya dari *parent component*. Dalam ReactJs, *component* dapat memiliki *property*, dan *property* ini membantu dalam membuat *component* lebih dinamis dan dapat digunakan ulang.

2.5.4 Context

Context merupakan sebuah *application programming interface* (API) pada ReactJs sebagai cara yang efektif menghasilkan variabel global yang dapat dilewatkan. Ini merupakan alternatif untuk *prop drilling* atau memindahkan properti dari *grandparent component* ke *child*

component ke *parent component*, dan seterusnya. Context juga dianggap sebagai pendekatan yang lebih mudah dan ringan untuk manajemen state (Ronen, n.d.).

2.6 Firebase

Firebase adalah layanan *Backend-as-a-Service* (BaaS) yang dikelola oleh Google dengan fitur-fitur kuat untuk pengembangan, penanganan, dan peningkatan aplikasi. Firebase digunakan sebagai layanan database, otentikasi, *cloud messaging* dan sebagainya dalam membangun aplikasi web dan mobile.

2.6.1 Firebase Firestore

Firestore adalah jenis layanan database NoSQL yang disediakan oleh Firebase. Database ini digunakan untuk menyimpan, menyinkronkan, dan melakukan kueri data dalam suatu sistem. Data yang tersimpan pada sebuah *collection* yang terdiri atas beberapa dokumen. Dokumen pada *collection* memiliki wujud *object* seperti JSON. Firestore Firebase memiliki kemampuan untuk mengoperasikan kueri yang lebih cepat, kompleks, dan mampu menangani skala data yang lebih besar dibandingkan dengan Realtime Database yang juga terdapat pada Firebase. Firestore mendukung sinkronisasi data secara offline. Ini berarti bahwa data dapat disimpan dalam *cache* pada perangkat seperti Android, IOS, dan web. Ketika pengguna mengakses data mereka melalui aplikasi dan melakukan perubahan pada data yang tersimpan di *cache* saat perangkat kembali terhubung ke internet, basis data Firestore akan melakukan sinkronisasi setiap perubahan lokal ke dalam Firestore. Dengan hal itu, dapat diimplementasikan untuk mengembangkan aplikasi yang responsif, powerfull, dan mampu bekerja tanpa bergantung pada latensi jaringan (Firebase, 2023).

2.6.2 Firebase Authentication

Firebase Authentication adalah salah satu layanan unggulan Firebase yang memudahkan dalam menyediakan layanan otentikasi. Firebase menyediakan *library* siap pakai yang sangat fleksibel, mempermudah proses implementasi otentikasi secara efisien. FirebaseUI, yang merupakan bagian dari Firebase Authentication, memberikan opsi autentikasi yang mudah diintegrasikan dan mengelola proses UI untuk pengguna yang ingin login menggunakan email dan kata sandi, atau menggunakan integrasi dengan akun Google, Facebook, Twitter, atau Github. Penggunaan komponen FirebaseUI Auth membantu meningkatkan konversi masuk dan pendaftaran pengguna pada aplikasi. Firebase Authentication terintegrasi erat dengan

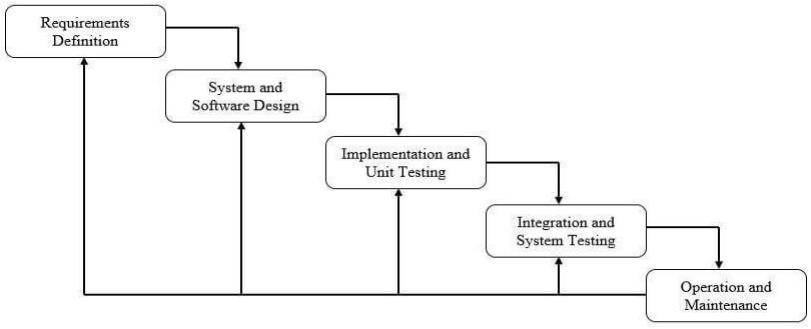
layanan Firebase lainnya dan memanfaatkan berbagai standar industri, seperti OAuth 2.0 dan OpenID Connect, sehingga dapat dengan mudah diintegrasikan dengan *backend* kustom (Firebase, 2023).

2.6.3 Firebase Storage

Storage pada Firebase adalah layanan yang disediakan oleh Firebase untuk menyimpan dan menampilkan konten yang dibuat oleh pengguna, seperti foto, file dokumen, audio, dan video. Layanan ini memiliki kemampuan untuk melakukan proses unggah dan unduh bahkan dalam kondisi jaringan yang tidak stabil. Proses unggah atau unduh dapat dilanjutkan dari titik terputus tanpa harus memulai ulang, mengoptimalkan penggunaan *bandwidth* dan menghemat waktu pengguna.

BAB III
METODOLOGI

Dalam membangun sebuah perangkat lunak tentu memerlukan sebuah metodologi agar proses yang direncanakan dapat berjalan tepat dan membantu segala rangkaian pengembang dalam membangun sebuah sistem sesuai dengan kebutuhan pengguna. Dalam membangun *platform* Wahda, metode pengembangan perangkat lunak yang digunakan adalah metode Waterfall. Dipilihnya metode ini dikarenakan kesepakatan dari tim dan pengerjaan proyek yang dilakukan secara terstruktur dari tim pengembang, dimana pada tahap awal anggota tim telah melakukan analisis kebutuhan dan desain antarmuka sebelum dilakukannya implementasi kode program. Dalam melakukan pengembagan *platform* Wahda, terdapat tiga peran orang yang terlibat, yaitu *Hustler*, *Hipster*, dan *Hacker*. Peran *Hustler* yang dipegang oleh Sulistio dengan melakukan beberapa tugas antara lain melakukan proses analisis kebutuhan, analisis proses bisnis, dan proses perancangan desain sistem. Selanjutnya, untuk Hipster yang dipegang oleh Muhammad Khoirul Umamil Achyar yang bertugas untuk merancang desain antarmuka, rancangan *flow* dalam penggunaan aplikasi , dan melakukan pengujian serta evaluasi desain yang telah dirancang sampai dapat diterima baik oleh pengguna. Posisi Hacker yang menjadi tugas dari penulis, Naufal Yusran, yang bertugaskan untuk mengimplementasikan segala kebutuhan sistem dan hasil desain yang final untuk dilanjutkan proses pengembangan sistem berupa pembuatan kode program dan pengujian pada fungsionalitas sistem. Berikut ini merupakan pemaparan dari tahapan metode dalam pengembangan sistem menggunakan metodologi Waterfall.



Gambar 3.3.1 Tahapan Waterfall dalam pengembagan *platform* Wahda

Sumber: tutorialspoint.com

3.1 Analisis Kebutuhan

Pada tahap ini, kebutuhan dari sistem dianalisis secara terstruktur dan dijelaskan dalam sebuah dokumen. Dokumen ini menjadi dasar untuk langkah-langkah pengembangan selanjutnya. Hasil dari dilakukannya analisis ini adalah dokumen persyaratan yang menjelaskan tujuan yang harus dicapai oleh sistem yang dibangun.

Dalam melakukan analisis kebutuhan, maka dilakukan observasi terhadap beberapa *platform* belajar online yang sudah ada dan ditemukan beberapa fungsional yang tergolong krusial untuk sistem yang akan dirancang, Kebutuhan fungsional tersebut dirangkum pada Tabel 3.1.

Tabel 3.1 Kebutuhan Fungsional Sistem

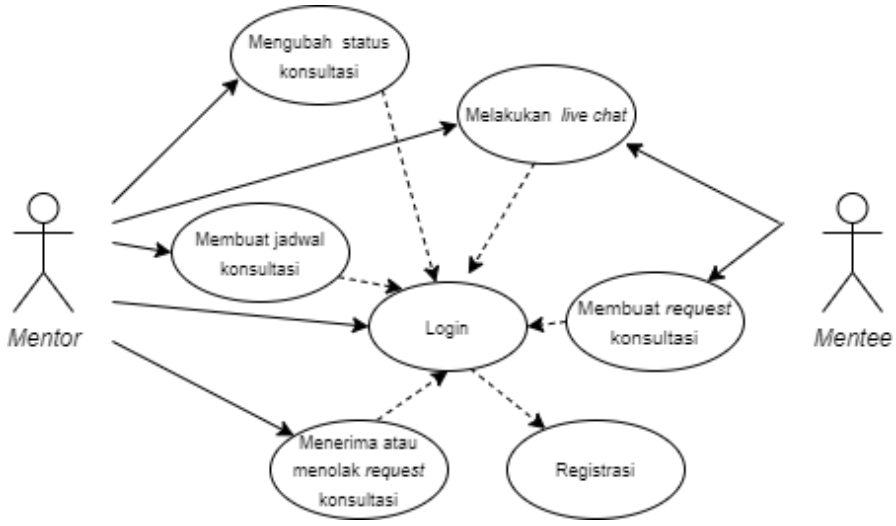
No	Kebutuhan Fungsional
1	Sistem dapat melakukan proses login dan registrasi pengguna dan mentor
2	Sistem dapat melakukan proses menambah dan mengubah jadwal konsultasi tiap mentor
3	Sistem dapat melakukan proses *request* konsultasi oleh pengguna kepada tiap mentor
4	Sistem dapat melakukan proses menerima *request* yang masuk pada tiap mentor
5	Sistem dapat melakukan proses konsultasi melalui *chat* dan *zoom*

3.1.1 Perancangan Alur Sistem

Setelah dilakukannya pengumpulan data mengenai kebutuhan yang diperlukan dalam mengembangkan sistem, maka dilakukanlah perancangan proses dari alur yang akan dibuat. Hal ini meliputi pembuatan *usecase diagram* dan *activity diagram* yang berguna dalam membantu pengembang dalam urutan aktivitas yang dilakukan dalam setiap fungsionalitasnya.

Usecase Diagram

Usecase diagram adalah representasi grafis yang memodelkan fungsionalitas sistem. Dalam *usecase diagram*, terdapat dua komponen kunci, yaitu aktor dan *usecase. Usecase diagram platform* Wahda dapat dilihat pada Gambar 3.2. Penjelasan rinci mengenai dari tiap *usecase*, dapat dilihat pada tabel 3.2. Pada rancangannya, Wahda memiliki dua aktor yang terlibat, yaitu pengguna mahasiswa atau *mentee* dan pengguna pengajar atau *mentor*. Platform Wahda juga memiliki aktor admin, yang dimana admin dapat mengubah role dari suatu pengguna.



Gambar 3.3.2 *Usecase diagram platform* Wahda

Tabel 3.2 Kebutuhan deskripsi *usecase diagram*

No	Nama *Usecase*	Deskripsi
UC. 1	Mengubah status konsultasi	Proses pengubahan status konsultasi yang diterima oleh *mentor* untuk mengetahui status dari konsultasi yang ada.
UC.2	Melakukan *live chat*	Proses melakukan konsultasi melalui media *chat* yang telah tersedia pada *platform* Wahda, Di sini pengguna dapat mengirimkan teks pesan maupun gambar ke lawan penggunanya.
UC.3	Membuat jadwal konsultasi	Proses pengubahan atau penambahan sesi jadwal yang dilakukan oleh pengguna *mentor*.
UC.4	Login	Proses masuk ke aplikasi dengan melakukan otentikasi pada *platform* Wahda
UC.5	Menerima atau menolak *request* konsultasi	Proses menerima atau menolak *request* konsultasi dari *mentee* yang dilakukan oleh pengguna *mentor*
UC.6	Mengubah *role* pengguna	Proses pengubahan *role* pengguna pada *platform* Wahda yang dapat dilakukan oleh *admin*.
UC.7	Membuat *request* konsultasi	Proses pembuatan konsultasi oleh *mentor* tertentu yang dilakukan oleh pengguna *mentee*
UC.8	Registrasi	Proses mendaftarkan diri sebagai pada *platform* Wahda

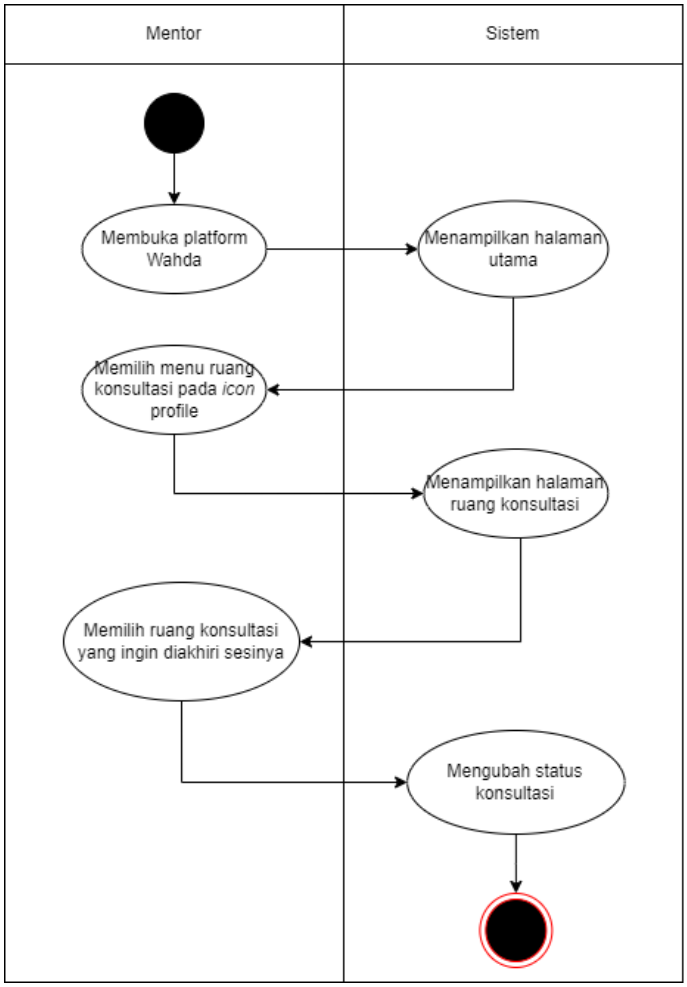
Activity Diagram

Setelah menentukan fungsionalitas yang ada pada platform Wahda, selanjutnya dibuatlah sebuah diagram alur untuk masing-masing fungsionalitas yang direpresentasikan

menggunakan *activity diagram*. Berikut merupakan *activity diagram* dari masing-masing fungsionalitas pada *platform* Wahda yang dilakukan hanya oleh pengguna *mentor* dan/atau pengguna *mentee*:

a) Mengubah status konsultasi

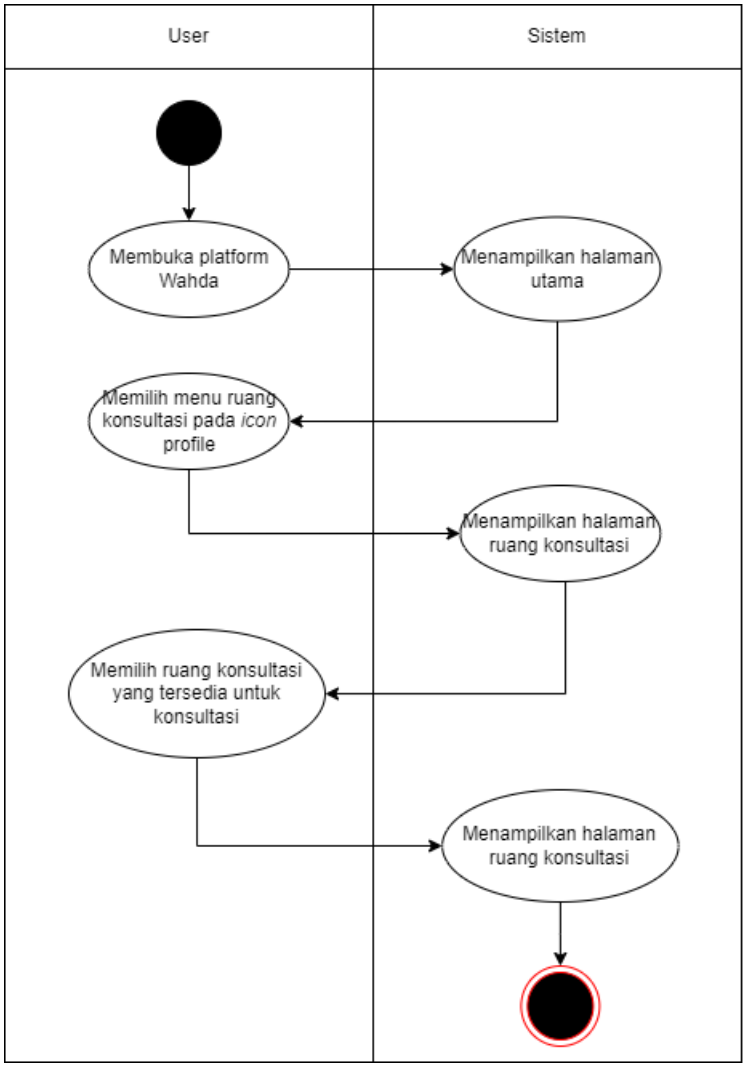
Pada pengubahan status konsultasi, alur yang dilakukan diawali dengan pengguna *mentee* berada pada halaman *homepage*. Setelah itu, pengguna dapat menuju halaman ruang konsultasi yang berada pada *dropdown* icon profile pada *navigation* dan mengubah status konsultasi dari aktif menjadi non-aktif dengan menekan tombol selesai konsultasi.



Gambar 3.3.3 *Activity diagram* mengubah status konsultasi

b) Melakukan *live-chat*

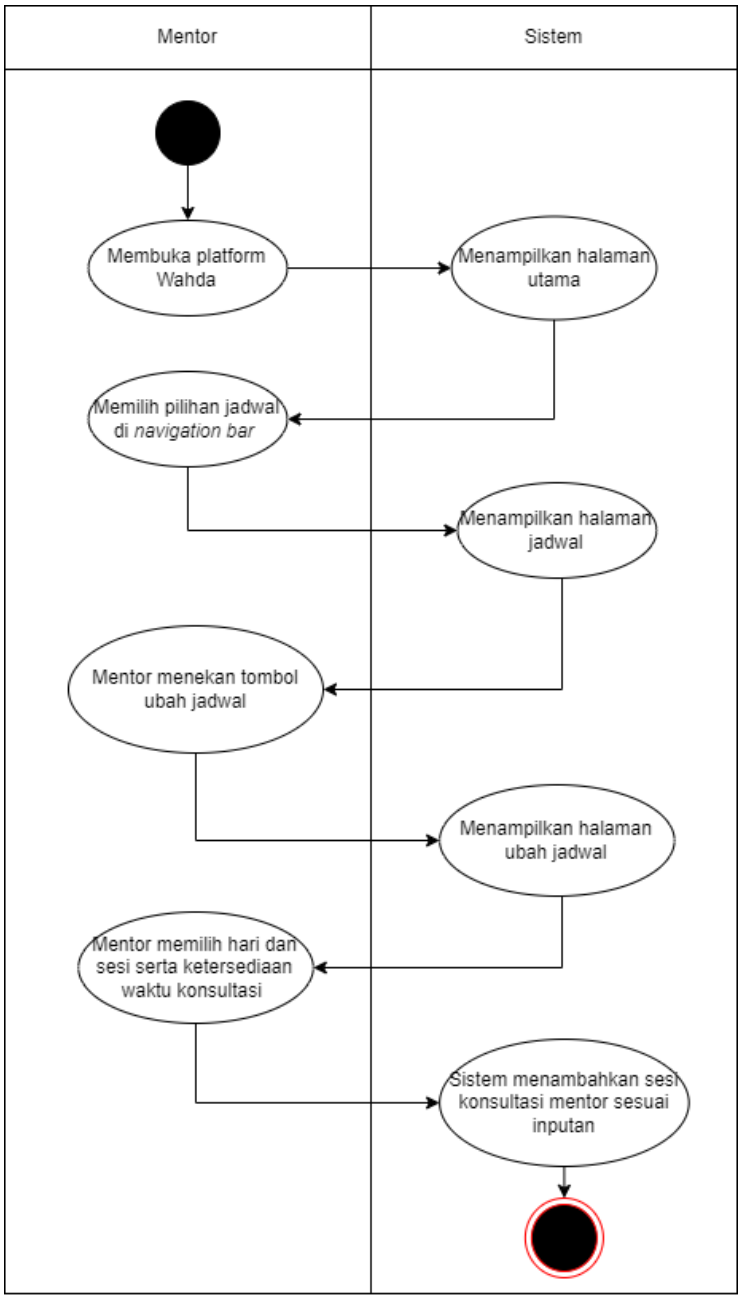
Alur dalam melakukan konsultasi berjenis *live-chat* dilakukan dengan pengguna, baik *mentor* atau *mentee*, masuk ke halaman ruang konsultasi yang berada pada *dropdown icon profile* dan memasuki halaman konsultasi yang ada.



Gambar 3.3.4 *Activity diagram* melakukan konsultasi *live-chat*

c) Membuat atau mengubah jadwal konsultasi

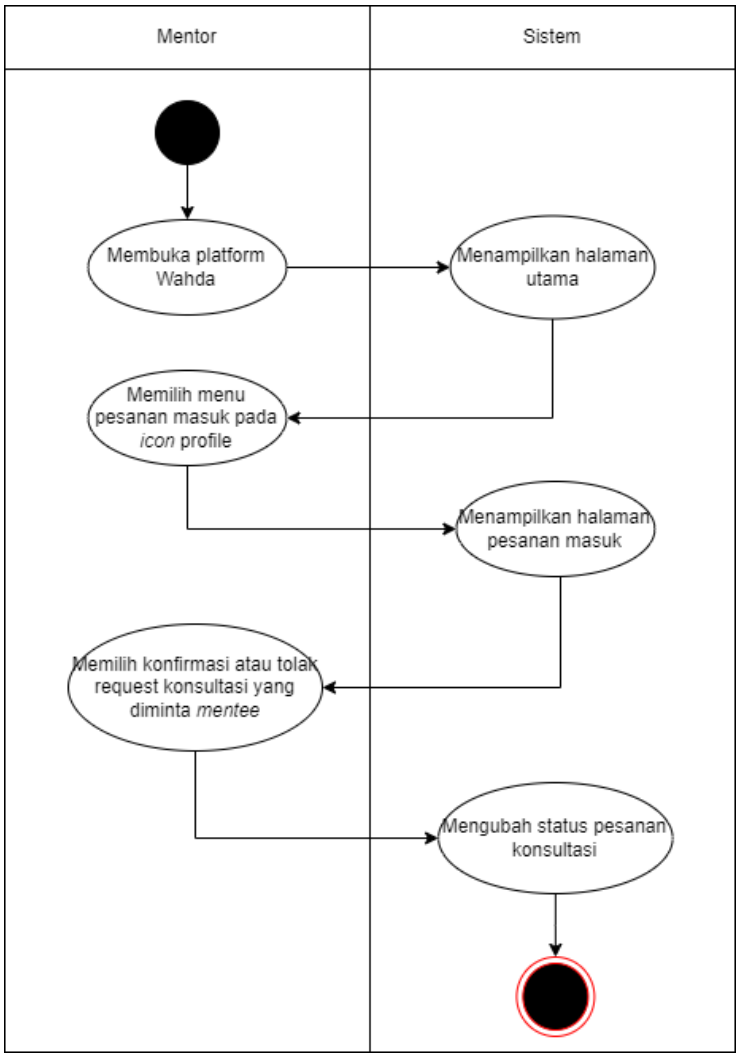
Pembuatan jadwal konsultasi dilakukan dengan pengguna *mentor* membuka halaman jadwal yang dapat dilakukan dengan menekan pilihan jadwal pada navigasi. Dilanjutkan dengan menekan tombol ubah jadwal yang berada pada bagian bawah halaman jadwal dan mengisi jam kesediaan konsultasi dan sesi yang disanggupi *mentor*.



Gambar 3.3.5 *Activity diagram* membuat atau mengubah jadwal konsultasi

d) Menerima atau menolak *request* konsultasi

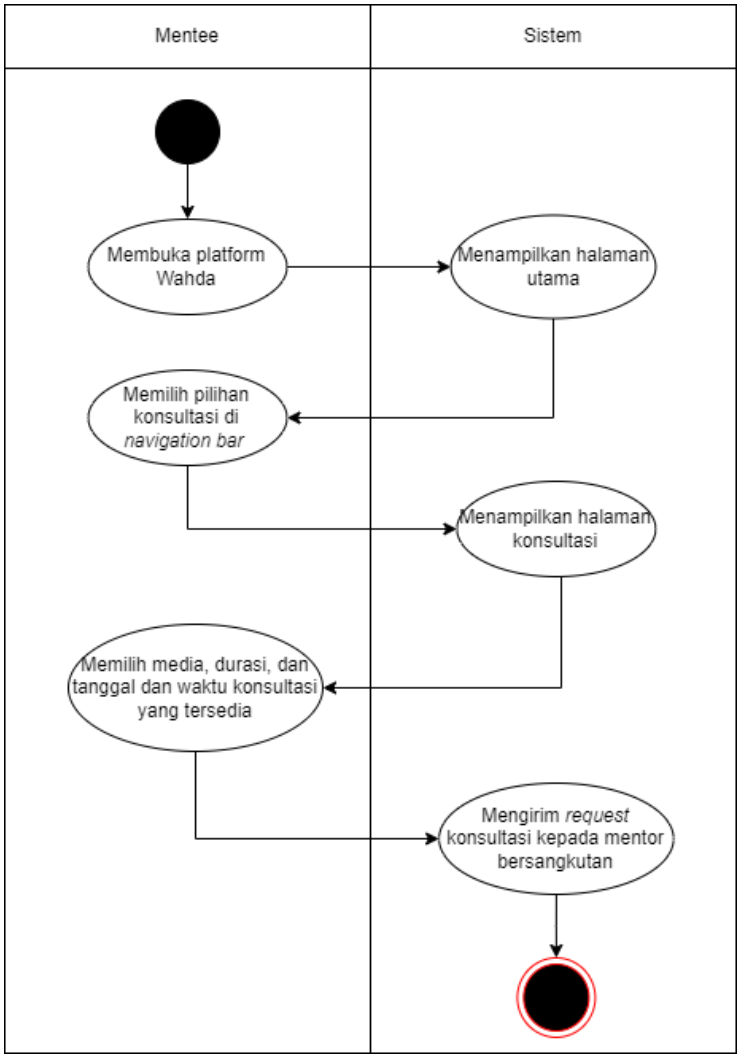
Alur yang harus dilakukan dalam menerima atau menolak request konsultasi dari *mentee* yang dilakukan oleh *mentor* dimulai dari pengguna masuk ke halaman pesanan masuk yang berada pada *icon profile* pada menu navigasi. Dari halaman ini, *mentor* dapat mengkonfirmasi atau menolak request yang ada dengan menekan tombol konfimasi atau tombol tolak.



Gambar 3.3.6 *Activity diagram* mengonfirmasi *request* konsultasi

e) Membuat *request* konsultasi

Pada fungsionalitas ini, pengguna *mentee* dapat melakukan *request* konsultasi dengan masuk ke halaman konsultasi melalui navigasi. Setelah memasuki halaman konsultasi, pengguna wajib memilih mentor untuk konsultasi, berdasarkan keahlian yang tertera di halaman tersebut. Setelah memilih *mentor*, pengguna juga diwajibkan mengisi *form* untuk menjelaskan permasalahannya kepada mentor. Selain itu, pengguna juga harus memilih media konsultasi, durasi, dan tanggal/waktu konsultasi sesuai dengan ketersediaan mentor yang dipilih.



Gambar 3.3.7 *Activity diagram* melakukan *request* konsultasi

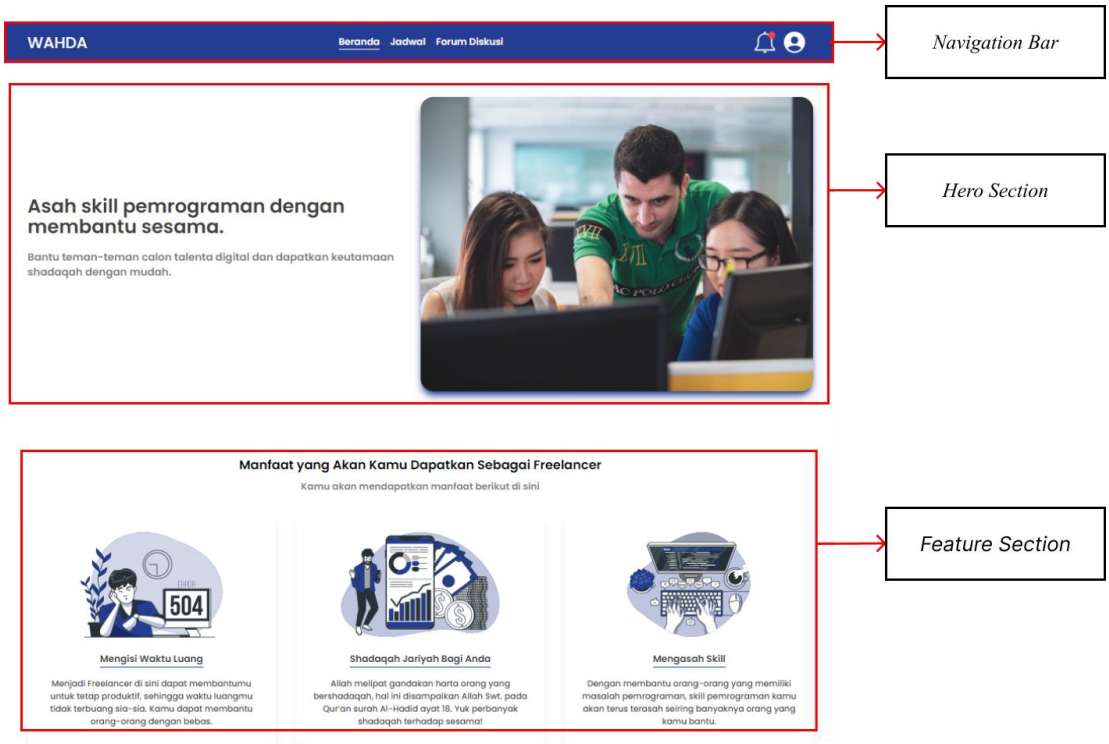
3.2 Desain Antarmuka Sistem

Setelah diselesaikan analisa kebutuhan, maka tahap berikutnya dalam pengembangan *platform* Wahda berdasarkan metode Waterfall adalah tahap desain antarmuka. Desain antarmuka adalah suatu tahapan dimana pengembang membuat tampilan visual dari sistem yang akan dirancang pada tahap implementasi. Tujuan dari tahapan ini adalah menciptakan suatu tampilan yang cantik secara visual, interaktif, intuitif, dan efektif di saat sistem dioperasikan oleh pengguna nantinya.

Perancangan Antarmuka dilakukan oleh Hipster dari tim Jaggernut, Muhammad Khoirul Umamil Achyar, yang menggunakan aplikasi Figma dalam melakukannya. Figma merupakan aplikasi desain yang berbasis cloud dan juga berfungsi sebagai alat prototyping untuk proyek digital. Figma dirancang dengan tujuan untuk mempermudah kolaborasi antara pengguna dalam sebuah proyek dan memungkinkan mereka bekerja secara tim secara fleksibel

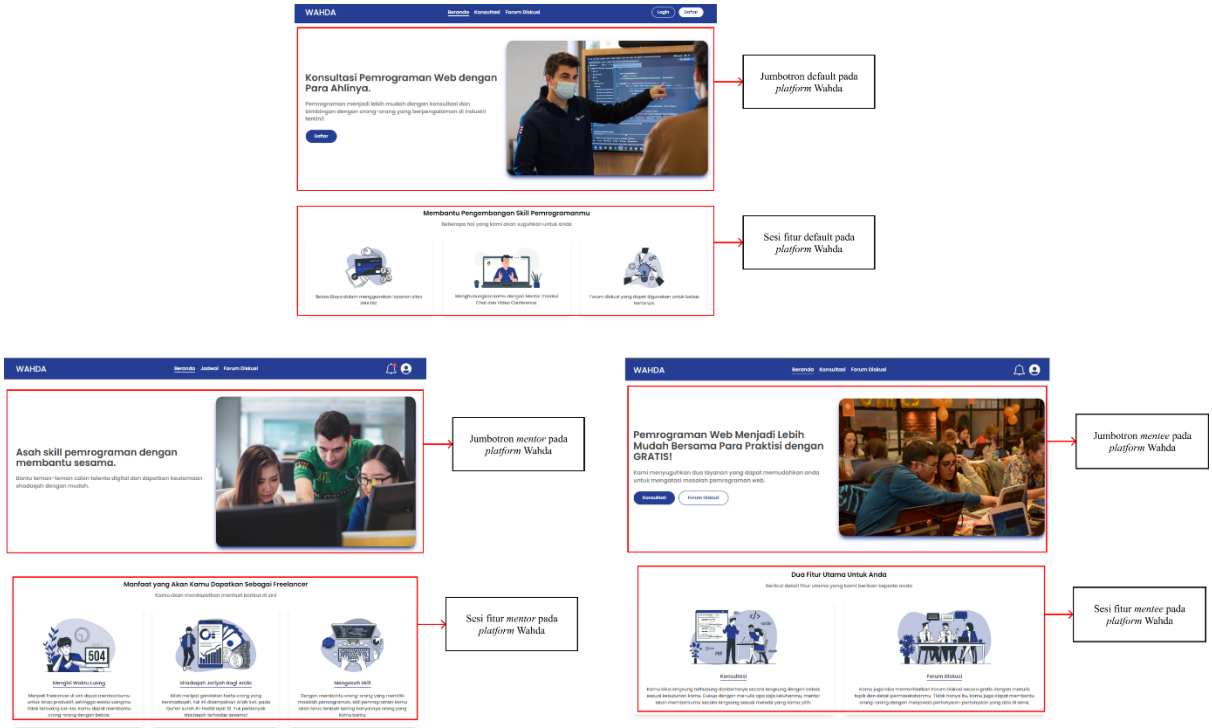
dimanapun mereka berada (Pramudita, Arifin, Alfian, Safitri, & Anwariya, 2021). Beberapa desain antarmuka yang akan dibahas pada bab ini meliputi desain beranda atau *homepage*, halaman request konsultasi, halaman jadwal dan halaman perubahan jadwal, halaman pesanan, dan halaman konsultasi menggunakan *live-chat*.

Pembahasan mengenai beberapa desain ini diambil dikarenakan halaman tersebut merupakan halaman yang dianggap krusial dalam tahap implementasi yang penulis lakukan sebagai *Hacker* dari tim Jaggernut. Pada Gambar 3.8 dapat dilihat desain dari halaman beranda pada platform Wahda. Halaman ini terdiri dari beberapa bagian, yaitu *navigation bar*, *hero*, dan *feature section*.



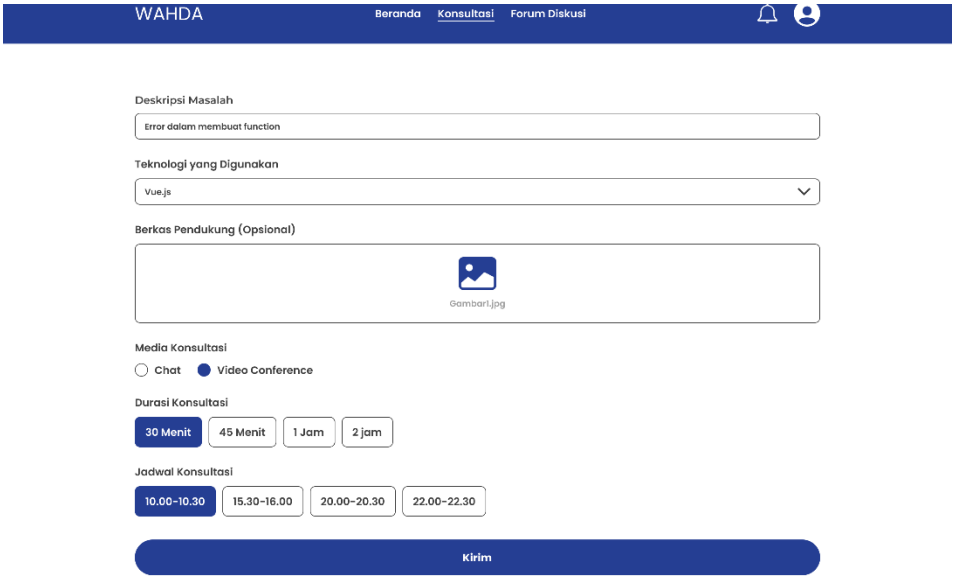
Gambar 3.3.8 Desain halaman beranda platform Wahda

Dalam tahap perencanaan, tampilan dari halaman beranda akan tampil berbeda berdasarkan sebuah *state*. *State* ini ditentukan melalui *role* dari user tersebut disaat melakukan otentikasi. State ini terbagi menjadi tiga, yaitu default, *mentor*, dan *mentee*. Perbedaan antarmuka dari halaman beranda terdapat pada dua bagian yang sudah disebutkan sebelumnya, *hero section* dan *feature section*. Perbedaan atas tampilan halaman beranda dapat dilihat pada Gambar 3.9.



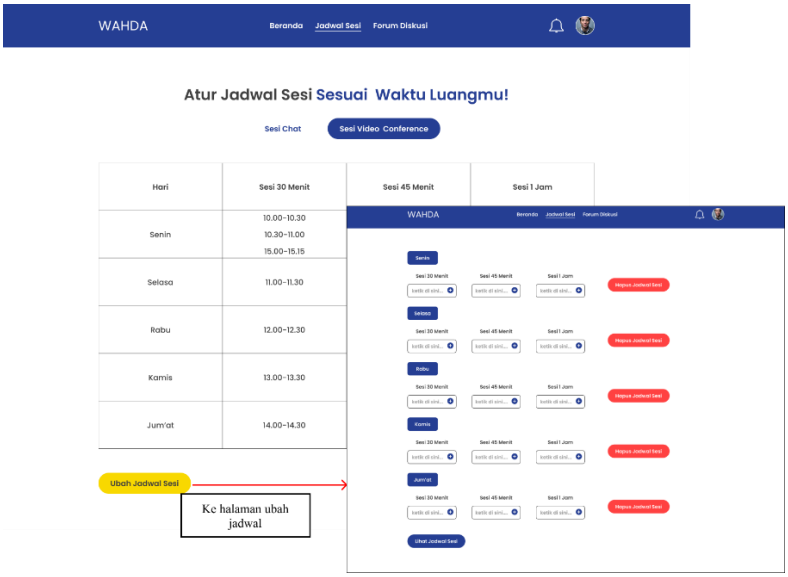
Gambar 3.3.9 Perbedaan tampilan halaman beranda platform Wahda

Selanjutnya, desain halaman untuk mengajukan request konsultasi tergambar pada Gambar 3.10. Halaman ini berbentuk formulir yang harus diisi oleh pengguna yang ingin mengajukan konsultasi terkait permasalahan yang dihadapinya. Terdapat beberapa field dalam formulir ini. Pertama, ada field untuk memasukkan deskripsi masalah yang memungkinkan pengguna untuk menceritakan permasalahan yang sedang dialami. Kedua, terdapat dropdown untuk memilih teknologi yang terkait dengan permasalahan yang dihadapi. Ketiga, pengguna dapat mengunggah berkas pendukung yang relevan (opsional). Keempat dan kelima, terdapat opsi untuk memilih media konsultasi dan durasi konsultasi yang diinginkan. Terakhir, pengguna harus memilih hari dan tanggal yang diinginkan untuk konsultasi.



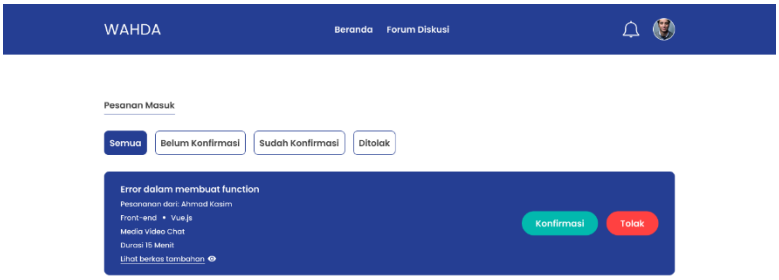
Gambar 3.3.10 Desain halaman pengajuan konsultasi platform Wahda

Berikutnya, terdapat halaman jadwal dan halaman ubah jadwal yang dapat dilihat pada Gambar 3.11, yang digunakan oleh mentor untuk melihat dan mengubah ketersediaan mereka dalam melakukan konsultasi. Jadwal konsultasi dibagi menjadi dua sesi terpisah: sesi live-chat dan sesi video conference, sehingga mentor dapat melihat sesi yang diinginkan berdasarkan medianya. Dalam melakukan perubahan jadwal, pengguna dapat beralih ke halaman perubahan jadwal dengan menekan tombol "ubah jadwal" yang terletak di bagian bawah halaman jadwal.



Gambar 3.3.11 Desain halaman jadwal dan ubah jadwal platform Wahda

Halaman pesanan dapat dilihat pada Gambar 3.12, merupakan salah satu pilihan yang terdapat pada icon profile pada navigasi di platform ini. Halaman ini berfungsi sebagai penampung request konsultasi yang masuk atau telah masuk. Dalam desainnya, akan terdapat beberapa filter pesanan yang dapat dilakukan oleh user, seperti filter pesanan menunggu konfirmasi, sudah dikonfirmasi, dan yang telah ditolak.



Gambar 3.3.12 Desain halaman pesanan masuk platform Wahda

Halaman terakhir yang akan dibahas adalah halaman *live-chat*. Pada halaman ini merupakan halaman yang akan digunakan dalam melakukan konsultasi bermedia *chat*. Halaman ini akan terakses jika mentor telah melakukan konfirmasi atas request konsultasi yang diajukan oleh pengguna *mentee* nantinya. Tampilan pada halaman ini dapat dilihat pada Gambar 3.13.

Gambar 3.3.13 Desain halaman konsultasi media *live-chat* platform Wahda

BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi Sistem

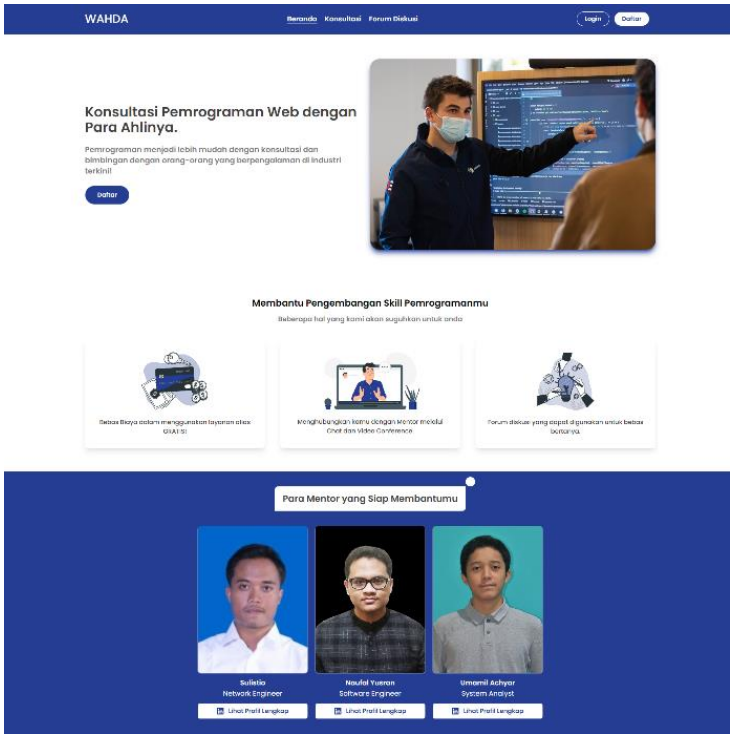
Proses yang selanjutnya dilakukan adalah melakukan implementasi atau pengembangan akan perancangan aplikasi yang telah dibuat. Pada pengembangannya, aplikasi Wahda menggunakan *library front-end* React.Js dan layanan *back-end* Firebase. Digunakan pula Tailwind CSS sebagai *library* yang memudahkan untuk mengembangkan aplikasi web dengan tampilan yang interaktif, responsif, dan memiliki fleksibilitas yang tinggi. Pengembangan dilakukan menggunakan *Visual Studio Code* sebagai *Code Editor* yang memuat berbagai macam ekstensi sehingga memudahkan proses pengembangan. Contoh ekstensi yang digunakan seperti, "Tailwind CSS IntelliSense", "ES7+ React/Redux/React-Native snippets", dan "Auto Rename Tag". Aplikasi Wahda yang dikembangkan merupakan aplikasi berbasis web yang sudah dapat diakses dan digunakan, baik menggunakan perangkat *smartphone* ataupun *laptop*.

4.1.1 Halaman Beranda

Halaman beranda akan ditampilkan saat pengguna mengakses alamat utama dari web Wahda. Pada halaman ini terdapat beberapa *state* atau keadaan tampilan. *State* tersebut ditentukan berdasarkan *state* autentikasi dan *role* dari pengguna.

Antarmuka Untuk Pengguna Baru

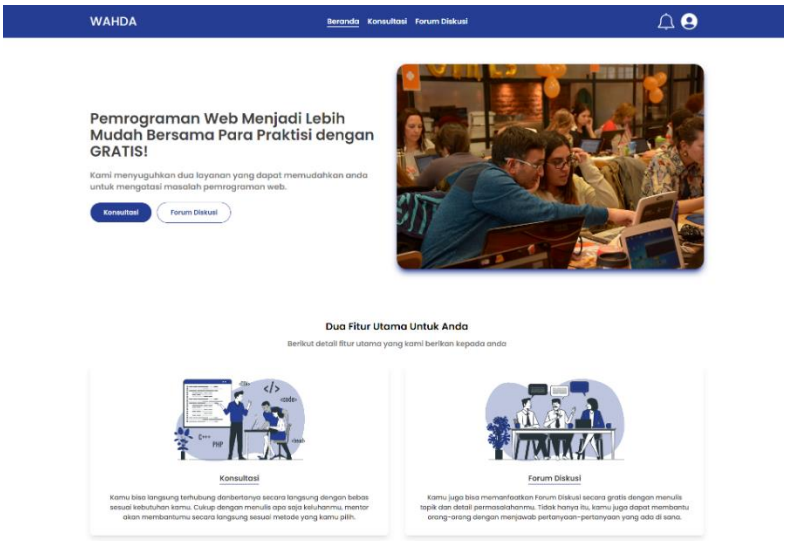
Untuk pengguna baru atau pengguna yang belum *login*, akan ditampilkan halaman seperti pada Gambar 4.1. Halaman tersebut menampilkan tombol "Daftar", fitur-fitur yang ada pada web jika mendaftar, dan daftar hingga 4 mentor yang di-randomisasi.



Gambar 4.1 Halaman Beranda Pengguna Belum *login*

Antarmuka Untuk Pengguna *Logged In*

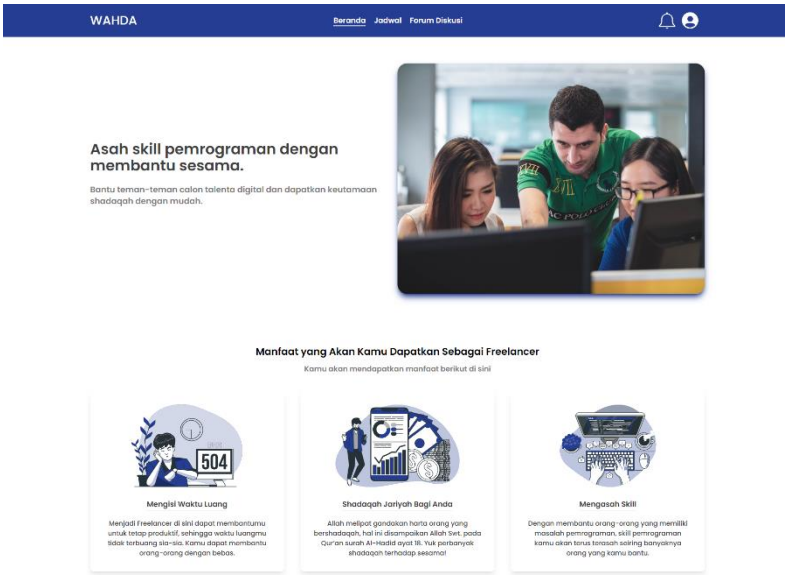
Untuk pengguna yang sudah *login*, akan ditampilkan halaman seperti pada Gambar 4.2. Halaman tersebut menampilkan tombol "Konsultasi" dan "Forum Diskusi", dan fitur-fitur yang ada pada aplikasi.



Gambar 4.2 Halaman Beranda Pengguna Sudah *login*

Antarmuka Untuk Pengguna Mentor

Untuk pengguna mentor, akan ditampilkan halaman seperti pada Gambar 4.3. Halaman tersebut menampilkan konten seputar manfaat yang didapatkan oleh mentor jika membantu *mentee* atau pengguna dalam menyelesaikan masalahnya.



Gambar 4.3 Halaman Beranda Pengguna Mentor

Modularisasi React

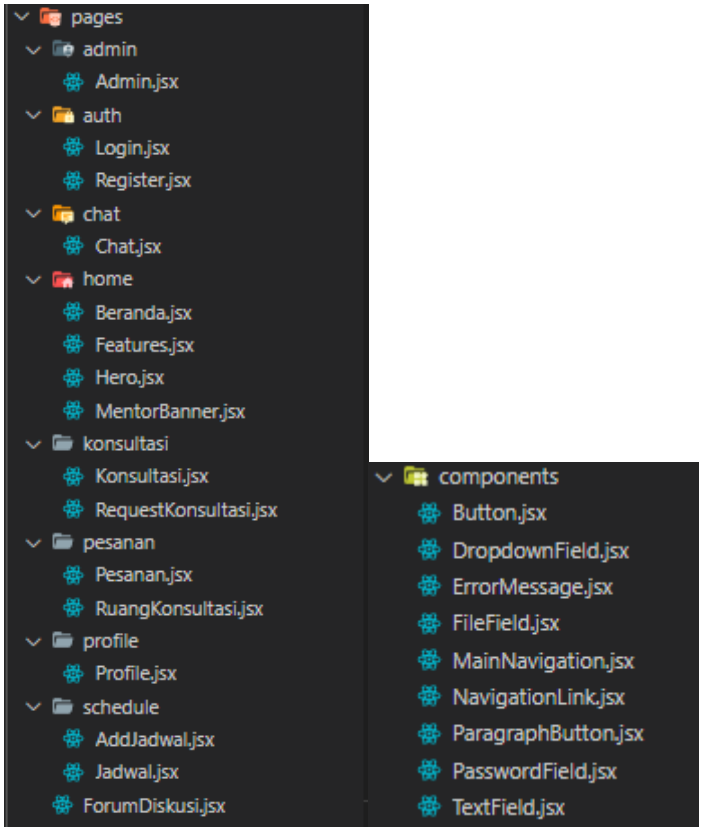
Pada React, terdapat konsep yang memungkinkan untuk memisahkan tiap-tiap tampilan kedalam komponen yang kecil. Hal tersebut membuat pengembangan menjadi lebih mudah dan cepat. Pada halaman Beranda sendiri, terbagi menjadi tiga komponen yang dinamakan *Hero*, *Features*, dan *MentorBanner*, seperti yang dapat dilihat pada Gambar 4.4.

```
const { user } = useContext(AuthContext);

return (
  <>
    <Hero
      image={
        user?.role === "mentor" ? MENTOR_HERO_DATA.image : HERO_DATA.image
      }
      title={
        user?.role === "mentor" ? MENTOR_HERO_DATA.title : HERO_DATA.title
      }
      content={
        user?.role    ===    "mentor"    ?    MENTOR_HERO_DATA.content    :
HERO_DATA.content
      }
      actionButton={
        user?.role === "mentor"
          ? MENTOR_HERO_DATA.actionButton
          : HERO_DATA.actionButton
      }
```

```
    />
    <Features
      title={
        user?.role === "mentor"
          ? MENTOR_FEATURES_DATA.title
          : FEATURES_DATA.title
      }
      subTitle={
        user?.role === "mentor"
          ? MENTOR_FEATURES_DATA.subTitle
          : FEATURES_DATA.subTitle
      }
      featureItems={
        user?.role === "mentor"
          ? MENTOR_FEATURES_DATA.featureItems
          : FEATURES_DATA.featureItems
      }
    />
    {!user && <MentorBanner />}
  </>
);
```

Gambar 4.4 Kode Halaman Beranda.

Untuk membuat komponen yang baru, baiknya dibuat kedalam satu *file* yang baru. Pemisahan ini dapat memudahkan pengembang, dikarenakan fokus pengembang menjadi lebih terbagi kedalam hal-hal yang perlu dikerjakan terlebih dahulu. Efek samping dari hal ini adalah dihasilkannya banyak *file* pada *project*. Pada aplikasi Wahda, untuk tiap halaman atau *page* dibuat satu komponen atau *file* yang baru. Dapat dilihat pada Gambar 4.5 untuk *file-file* yang ada pada aplikasi Wahda.



Gambar 4.5 Daftar *File* Komponen dan *Page* Pada Aplikasi Wahda

Untuk kode dari salah satu komponen yang ada pada halaman Beranda dapat dilihat pada Gambar 4.6 di bawah. Kode tersebut adalah kode untuk komponen *Hero*. Dapat dilihat bahwa isi dari komponen tersebut adalah kode atau *tag-tag* HTML biasa dengan kelas-kelas untuk *styling* dari Tailwind. Untuk konten dari komponen *Hero*, ditentukan dari variabel yang dinamakan *props*. *Props* ini didapatkan saat komponen *Hero* tersebut dipanggil. Dikarenakan *Hero* memiliki tampilan yang berbeda tergantung dari *state* autentikasi atau *role* dari pengguna, maka *props* yang dimasukkan juga berbeda, seperti yang dapat dilihat pada Gambar 4.6.

```
const Hero = ({ image, title, content, actionButton }) => {
  return (
    <section className="container flex flex-col-reverse gap-5 items-center
p-5 pt-20 mx-auto mb-5 lg:flex-row">
      <div className="flex flex-col gap-6">
        <h1  className="text-2xl  font-semibold  text-overlay  text-center
lg:text-left lg:text-3xl xl:text-4xl">
          {title}
        </h1>
        <p  className="text-lg  font-semibold  text-paragraph  text-center
lg:text-left xl:text-xl">
          {content}
        </p>
        <div className="flex justify-center items-center gap-3 lg:justify-
start">
          {actionButton}
```

```
        </div>
      </div>
      <img
        src={image}
        alt="Ilustrasi Section Hero"
        className="w-4/5 my-8 object-cover shadow-lg shadow-primary rounded-
3xl lg:my-16 lg:w-1/2"
      />
    </section>
  );
};
```

Gambar 4.6 Kode Komponen *Hero*

Integrasi Firebase *Authentication* Dengan React

Untuk menentukan *state* autentikasi atau *role* dari pengguna, digunakan fitur yang dinamakan *context* pada React. Fitur ini pada dasarnya berguna sebagai tempat penyimpanan utama dari suatu *state*, sehingga tiap anak komponen dapat mengakses *state* tersebut. Pada aplikasi Wahda, *context* autentikasi membungkus semua komponen yang ada. Hal tersebut dikarenakan diperlukannya data pengguna yang sedang *login* pada sebagian besar halaman atau komponen yang ada, serta terdapat beberapa halaman yang memerlukan *route protecting* yang dimana hanya dapat diakses saat pengguna telah *login* terlebih dahulu. Pada aplikasi Wahda, dibuat *file* yang khusus untuk menangani *context* autentikasi. Kode *AuthContext* dapat dilihat pada Gambar 4.7. Pertama *export* variabel *AuthContext* yang nantinya berguna saat ingin mengakses nilai yang ada pada *context* autentikasi tersebut. Penggunaan *context* dapat dilihat pada Gambar 4.4. Kemudian terdapat fungsi *AuthContextProvider*, fungsi ini yang nantinya akan membungkus komponen yang ingin diberikan *context*, dalam aplikasi Wahda adalah keseluruhan aplikasi. Pada fungsi *AuthContextProvider* dibuat *state user* dan *isFetchUser*. Dimana kedua *state* tersebut dimasukkan ke dalam *value* sehingga anak komponen yang dibungkus oleh *AuthContext* dapat mengakses *state* tersebut. Pada fungsi *AuthContext* terdapat *useEffect*, yang digunakan untuk mengubah *state user* sesuai dengan *user* yang telah *login*. Untuk mengetahui *user* yang telah *login*, dijalankan fungsi *onAuthStateChanged*, dimana fungsi ini berasal dari *library Firebase/Auth*. Fungsi tersebut dijalankan dengan argumen *auth*, dimana objek *auth* merupakan variabel yang menyimpan data akan *project* Firebase yang dikoneksikan. Selanjutnya, argumen kedua yaitu *anonymous function* yang akan dijalankan saat terjadi perubahan *state auth*. Firebase Authentication hanya menyimpan data-data terkait autentikasi saja seperti *email* dan *password*. Untuk itu, digunakan fitur Firebase Firestore sebagai *database* yang akan menyimpan data-data tambahan tiap *user* seperti *role*. Sehingga, pada *anonymous function* tersebut dijalankan fungsi yang akan mengambil data suatu *user* dari Firestore berdasarkan *uid* dari variabel *user* jika variabel *user* tidak *null*. Kemudian akan

dilakukan *setState user* sehingga anak komponen yang menggunakan *state* tersebut akan melakukan *rerender*. Untuk variabel *isFetchUser* itu sendiri digunakan untuk menampilkan tampilan *loading* saat halaman pertama kali dikunjungi dan proses pengambilan data *user* belum selesai.

```
// data config ini didapatkan dari project Firebase
const firebaseConfig = {
  apiKey: "",
  authDomain: "",
  projectId: "wahda-uii",
  storageBucket: "",
  messagingSenderId: "",
  appId: "",
  measurementId: "",
};
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);

export const AuthContext = createContext();

export const AuthContextProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  const [isFetchUser, setIsFetchUser] = useState(true);

  useEffect(() => {
    onAuthStateChanged(auth, (user) => {
      if (user) {
        getDoc(doc(firestore, "users", user.uid)).then((snapshot) => {
          setIsFetchUser(false);
          setUser(snapshot.data());
        });
      } else {
        setIsFetchUser(false);
        setUser(user);
      }
    });
  }, []);

  return (
    <AuthContext.Provider value={{ user, isFetchUser }}>
      {children}
    </AuthContext.Provider>
  );
};
```

Gambar 4.7 Kode *AuthContext*

4.1.2 Halaman Konsultasi

Halaman Konsultasi akan ditampilkan saat pengguna mengakses alamat “/konsultasi” pada aplikasi Wahda. Halaman ini akan menampilkan daftar mentor yang terdaftar pada aplikasi. Pengguna dapat melakukan *request* dengan meng-*hover* gambar dari salah satu

mentor, seperti yang dapat dilihat pada Gambar 4.8. Terdapat juga pilihan untuk melakukan filter yang dapat memilah mentor berdasarkan keahlian yang dikuasai, dengan begitu pengguna dapat dengan mudah memilih mentor. Jika pengguna mengklik tombol "Konsultasi" yang muncul, maka pengguna akan diarahkan ke halaman *Request* Konsultasi. Untuk pengguna pada perangkat *mobile* dapat menekan salah satu gambar mentor dan tombol yang serupa akan ditampilkan.

Pada halaman ini juga ditampilkan tombol teknologi-teknologi yang telah dimasukkan oleh pengguna admin. Tombol ini berfungsi sebagai *filter* terhadap daftar mentor yang ditampilkan. Sehingga pengguna dapat melihat mentor mana saja yang menguasai teknologi tertentu.

Gambar 4.8 Halaman Konsultasi

Kode Halaman Konsultasi

Pada halaman Konsultasi, daftar mentor yang ditampilkan didapat dari *database* Firebase Firestore. Kode untuk mengambil data mentor terdapat pada baris ke-4 pada Gambar 4.9. Setelah mendapatkan daftar mentor yang disimpan pada variabel *documents*, dilakukan *mapping* atau mengubah data-data mentor tersebut kedalam komponen *MentorCard* yang kemudian disimpan kedalam variabel *renderedMentor* dan ditampilkan pada *return* komponen *Konsultasi*. Pada tiap komponen *MentorCard*, terdapat fungsi *onKonsultasiClick*, yang akan mengarahkan ke alamat "/konsultasi/request" untuk melakukan *request* konsultasi, beserta *id* dari mentor yang dipilih. Pada fungsi *onKonsultasiClick* juga terdapat pengecekan akan *state user* yang didapat dari *AuthContext*, jika *user* sama dengan *null* atau *user* belum melakukan *login*, maka akan diarahkan ke halaman Login terlebih dahulu. Untuk mengarahkan *user* ke halaman lain, digunakan fungsi *navigate* yang berasal dari *package* "react-router-dom". Kode selengkapnya dapat dilihat pada Gambar 4.9 di bawah.

Pada halaman ini, proses filtrasi dilakukan dengan mengambil data teknologi yang sudah terdaftar di Firebase. Setelah itu, beberapa pilihan teknologi akan ditampilkan. Saat pengguna

menekan salah satu pilihan tersebut, sistem akan melakukan proses filtrasi terhadap mentor berdasarkan teknologi yang dipilih.

```
const Konsultasi = () => {
  const { user } = useContext(AuthContext);
  const navigate = useNavigate();
  const { documents, error } = useCollection("users", ["role", "==",
"mentor"]);
  const [technologies, setTechnologies] = useState(["Semua"]);
  const [techIndex, setTechIndex] = useState(0);
  const [filteredDocuments, setfilteredDocuments] = useState(null);

  const onClickTech = (tech, index) => {
    if (index === 0) {
      setfilteredDocuments(documents);
    } else {
      const data = documents.filter((doc) => {
        return doc?.mentorData?.technologies.find(item => item === tech);
      });
      setfilteredDocuments(data);
    }

    setTechIndex(index);
  };

  const onKonsultasiClick = (id) => {
    if (user) {
      navigate("/konsultasi/request", {
        state: {
          mentorId: id,
        },
      });
    } else {
      navigate("/login");
    }
  };

  const renderedMentor = filteredDocuments?.map((doc) => {
    if (doc.mentorData) {
      return (
        <MentorCard
          key={doc.uid}
          image={doc.profileImage}
          name={`${doc.firstName} ${doc.lastName}`}
          position={doc.mentorData.position}
          linkedInUrl={doc.mentorData.linkedInProfile}
          onClick={() => onKonsultasiClick(doc.uid)}
        />
      );
    }

    return null;
  });

  useEffect(() => {
    if (documents) {
      setfilteredDocuments(documents);
```

```
    }

  }, [documents]);

  return (
    <section className="container mx-auto py-20 flex flex-col items-center
overflow-hidden">
      <img src={leftIcon} className="absolute left-0" alt="wave" />
      <img src={rightIcon} className="absolute right-0" alt="wave" />

      <h1 className="text-center text-2xl font-semibold text-overlay px-5
mb-4 mt-16 xl:text-4xl">
        Pilih Mentor yang Tersedia{" "}
        <span className="text-primary">Sesuai Kebutuhanmu</span>
      </h1>

      <div className="self-stretch flex mb-16 mt-4 overflow-x-scroll no-
scrollbar px-5">
        <div className="flex gap-5">
          {technologies?.map((tech, index) => {
            let selectedTechClass = "text-overlay bg-white border-border";

            if (techIndex === index) {
              selectedTechClass = "text-white bg-primary border-primary";
            }

            return (
              <div
                key={index}
                className={`${selectedTechClass} text-center font-semibold
rounded-lg p-3 border-2 transition duration-200 cursor-pointer hover:bg-
primary hover:border-primary hover:text-white`}
                onClick={() => onClickTech(tech, index)}
              >
                {tech}
              </div>
            );
          })}
        </div>
      </div>

      <div className="self-stretch overflow-x-scroll no-scrollbar px-5">
        <div className="flex items-center gap-10">
          {error && <ErrorMessage />}
          {renderedMentor}
        </div>
      </div>
    </section>
  );
};
```

Gambar 4.9 Kode Halaman Konsultasi

Konsep *Hooks* Dan Implementasinya Pada Pengambilan Data Dari Firestore

Pada halaman Konsultasi, dilakukan pemanggilan fungsi *useCollection* yang mengambil data dari Firestore. Pada React, konsep fungsi ini dinamakan *hooks*. Pada dasarnya, *hooks* adalah fungsi yang digunakan untuk memisahkan suatu fungsionalitas yang dapat digunakan

kembali pada komponen fungsional React. Komponen fungsional adalah cara untuk membuat komponen seperti membuat sebuah fungsi yang mengembalikan elemen React. Sebelumnya, untuk membuat sebuah komponen pada React, dilakukan dengan membuat kelas yang meng-*extend* kelas *React.Component*. *Hooks* diperkenalkan bersamaan dengan *functional component* yang ditujukan untuk dapat menggantikan bagaimana melakukan isolasi fungsionalitas dari *class based component* sebelumnya. Proses pengambilan data dari Firestore memiliki suatu pola tertentu. Untuk itu, dibuat *hooks* atau fungsi yang dapat digunakan berulang kali. *Hooks* yang digunakan untuk mengakses suatu *collection* pada Firestore dinamakan *useCollection*. *Hooks* ini berguna untuk mengambil seluruh *document* yang ada pada suatu *collection*. Kode untuk *hooks useCollection* dapat dilihat pada Gambar 4.10. *Hooks useCollection* menerima argumen *_collection*, *_query*, dan *_orderBy* yang kegunaannya adalah untuk mengambil suatu *collection* berdasarkan *query* dan *order* tertentu. Untuk contoh dari pemanggilan *hooks useCollection* dapat dilihat pada Gambar 4.9. Pada *hooks useCollection* ini, terdapat dua *state* yaitu *documents* dan *error*. Proses pengambilan data dari Firestore terjadi di dalam *useEffect*, dikarenakan *useEffect* akan melakukan pemanggilan fungsi yang ada di dalamnya pada saat aplikasi pertama kali melakukan *render*. Di dalam fungsi yang dijalankan pada *useEffect*, dilakukan pengambilan data dari Firestore dengan fungsi *onSnapshot*. Fungsi ini didapatkan dari *package* “firebase/firestore”, dimana fungsi membutuhkan suatu objek yang berfungsi sebagai *query* akan data yang ingin diambil sebagai argumen pertamanya, dan argumen keduanya adalah suatu *callback function* yang akan berjalan saat pengambilan data dilakukan dan saat terjadinya perubahan pada *collection* yang sedang dilakukan pengambilan data tersebut. Data yang didapatkan akan dimasukkan ke dalam argumen yang terdapat pada *callback function* tersebut. Data tersebut kemudian dimasukkan ke dalam *state documents*.

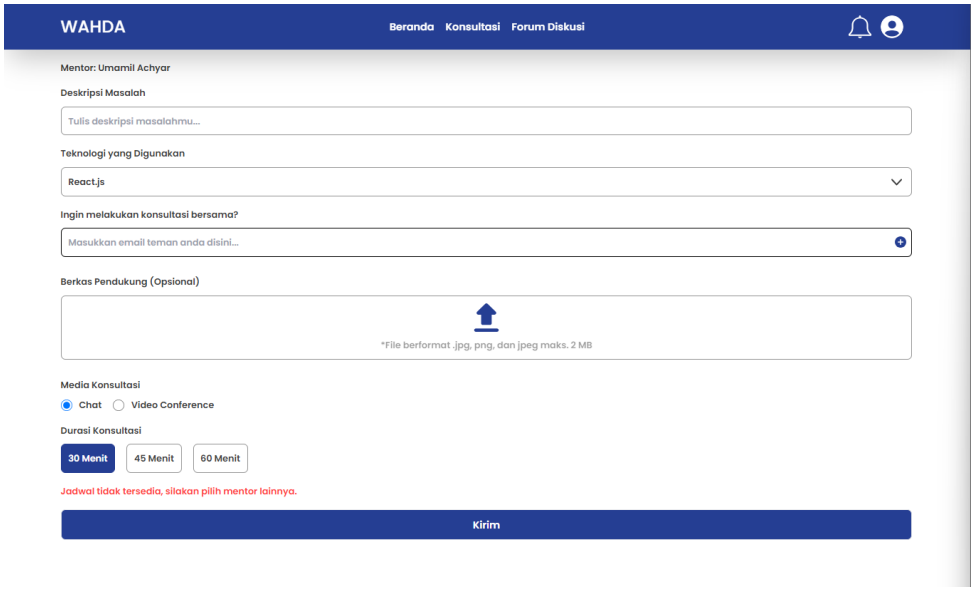
```
export const useCollection = (_collection, _query, _orderBy) => {
  const [documents, setDocuments] = useState(null);
  const [error, setError] = useState(null);

  const queryString = useRef(_query).current;
  const orderByString = useRef(_orderBy).current;

  useEffect(() => {
    let ref = collection(firestore, _collection);

    if (queryString) {
      ref = query(ref, where(...queryString));
    }
    if (orderByString) {
      ref = query(ref, orderBy(...orderByString));
    }
```

```
    const unsubscribe = onSnapshot(
      ref,
      (snapshot) => {
        let results = [];
        snapshot.docs.forEach((doc) => {
          results.push({ ...doc.data(), id: doc.id });
        });

        setDocuments(results);
        setError(null);
      },
      (error) => {
        setError("could not fetch the data");
      }
    );

    return () => unsubscribe();
  }, [_collection, queryString, orderByString]);

  return { documents, error };
};
```

Gambar 4.10 Kode *useCollection*

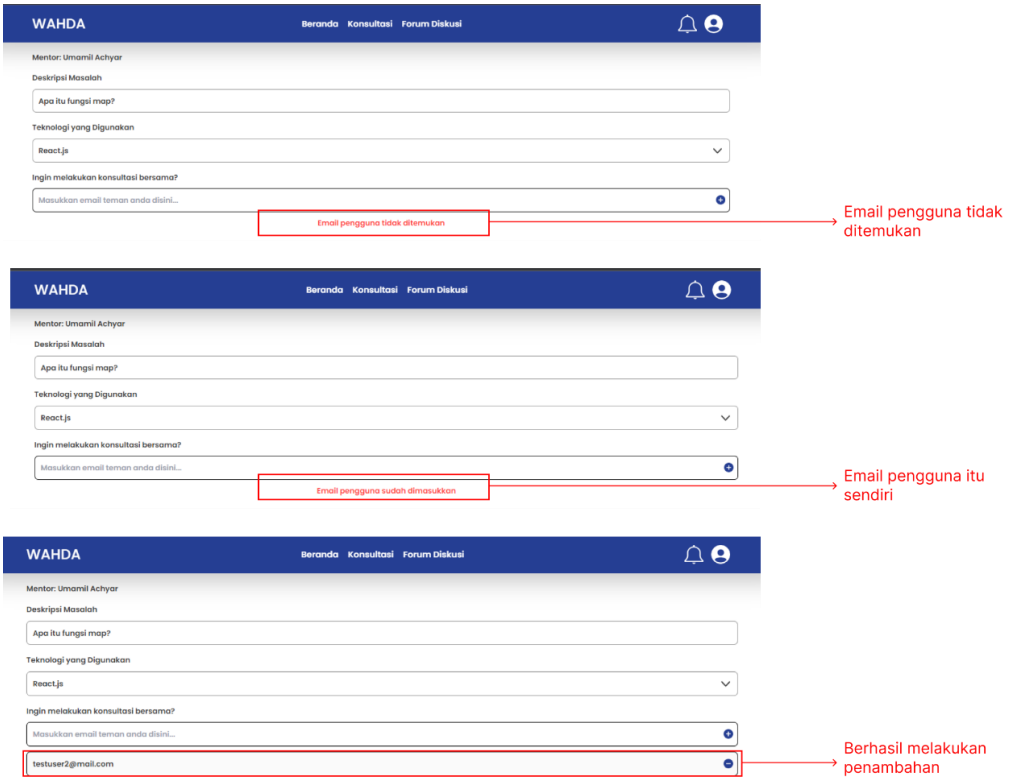
4.1.3 Halaman *Request* Konsultasi

Halaman *Request* Konsultasi akan ditampilkan saat pengguna menekan tombol "Konsultasi" pada halaman Konsultasi. Jika pengguna langsung mengakses alamat "/konsultasi/request" tanpa melalui pemilihan mentor terlebih dahulu maka akan langsung diarahkan ke halaman Konsultasi untuk memilih mentor. Halaman *Request* Konsultasi dapat dilihat pada Gambar 4.11 di bawah. Pada halaman *Request* Konsultasi, pengguna ditampilkan *form* yang harus pengguna isi sebelum melakukan *submit* dengan menekan tombol "Kirim". Untuk *input dropdown* "Teknologi yang Digunakan", data tersebut didapatkan dari masing-masing mentor. Untuk mentor nantinya dapat menambahkan teknologi yang digunakan ada halaman Profil. Setelah itu terdapat pemilihan waktu konsultasi yang bergantung pada media, durasi, dan hari konsultasi yang dipilih oleh pengguna serta jadwal yang sudah diatur oleh masing-masing mentor pada halaman Jadwal. Pada pemilihan hari konsultasi, akan ditampilkan hari-hari dimana terdapat jadwal yang tersedia saja.



Gambar 4.11 Halaman *Request* Konsultasi

Selain dapat melakukan request konsultasi sendiri, pengguna juga dapat mengundang pengguna lain untuk melakukan konsultasi bersama dengan memasukkan email pengguna lain di input pada form request konsultasi. Tentunya dalam melakukan penambahan pengguna yang disertakan dalam melakukan konsultasi terdapat beberapa pengecekan, seperti memasukkan email yang tidak valid dan email pengguna itu sendiri. Hal ini bisa dilihat pada Gambar 4.12.



Gambar 4.12 Perbedaan disaat melakukan penambahan pengguna

Kode Halaman *Request* Konsultasi

Kode untuk halaman *Request* Konsultasi dapat dilihat pada Gambar 4.13 di bawah. Halaman ini menerapkan banyak konsep seperti yang sudah dijelaskan sebelumnya. Konsep *AuthContext* untuk mengambil data pengguna yang *logged in*, yang kemudian digunakan sebagai pengenal akan siapa yang membuat suatu *request* konsultasi. Penggunaan *hooks* sebagai fungsionalitas tambahan, dimana pada halaman ini digunakan *hooks useFirestore* yang berfungsi untuk menambahkan suatu *document* atau data baru pada suatu *collection* yang ada di Firestore. Serta digunakan pula *hooks useRequestKonsultasi*, yang memisahkan fungsionalitas pemilihan jadwal konsultasi agar memudahkan untuk melakukan pemeliharaan kode pada kedepannya. Digunakan pula fungsi *useEffect*, dimana digunakan untuk melakukan pengecekan apakah pengguna masuk ke halaman *request* konsultasi setelah memilih mentor, menghapus tampilan pesan *error* saat pengguna melakukan interaksi pada salah satu *field*, serta digunakan untuk mengambil data mentor yang dipilih oleh pengguna dari Firestore.

Kode yang menghasilkan tampilan terdapat pada baris yang mengembalikan atau *return* elemen HTML yaitu *form*. Pada elemen *form* terdapat atribut *onSubmit* yang berisi fungsi *onFormSubmit* dimana fungsi tersebut akan dijalankan saat pengguna melakukan *submit* dengan menekan tombol “Kirim”. Pada fungsi tersebut akan dilakukan pengecekan untuk data yang telah diisi oleh pengguna, jika data masih ada yang belum terisi maka proses penambahan data ke Firestore tidak dijalankan dan aplikasi akan menampilkan pesan *error*. Setelah itu akan dilakukan proses *upload* ke Firebase Storage jika pengguna memasukkan gambar. Setelah itu dilakukan penambahan data *request* konsultasi serta notifikasi menggunakan *hooks useFirestore*. Jika proses berhasil dilakukan, maka pengguna akan diarahkan ke alamat “/pesanan”.

```
const RequestKonsultasi = () => {
  const { user } = useContext(AuthContext);

  const { state } = useLocation();
  const navigate = useNavigate();

  const [isSubmitting, setIsSubmitting] = useState(false);
  const [errorMessage, setErrorMessage] = useState(null);

  const { addDocument } = useFirestore("requests");
  const { addDocument: addNotification } = useFirestore("notifications");

  const {
    ...
  } = useRequestKonsultasi(state?.mentorId);
```

```
  return (
    <form
      autoComplete="off"
      className="container mx-auto flex flex-col gap-5 p-5 pt-20 text-
overlay font-semibold"
      onSubmit={onFormSubmit}
      onChange={() => setErrorMessage(null)}
    >
      <p className="pt-5">
        Mentor: {`${userData?.firstName} ${userData?.lastName}`}
      </p>
      <div className="flex flex-col gap-5">

        ...

        <div className="flex flex-col gap-3">
          <p> Media Konsultasi</p>
           ...
        </div>

        <div className="flex flex-col gap-5">
          . . .
            <>
              <div className="flex flex-col gap-3">
                <p>Hari Konsultasi</p>
                 ...
              </div>

              <div className="flex flex-col gap-3">
                <p>
                  Waktu Konsultasi{" "}
                  <span className="font-normal">*(24 hours format)</span>
                </p>
                 ...
              </div>
            </>
          )}
        </div>
      </div>
      <div>
        {errorMessage && <ErrorMessage message={errorMessage} />}
        <Button
          className="text-white text-lg w-full py-3 rounded-lg"
          disabled={isSubmitting}
          style={{ backgroundColor: isSubmitting ? "grey" : "#253e93" }}
        >
          Kirim
        </Button>
      </div>
    </form>
  );
};
```

Gambar 4.13 Kode Halaman *Request* Konsultasi

Dalam melakukan penambahan pengguna untuk melakukan konsultasi bersama, fungsi *onAddUserClick*-lah yang bertugas untuk melakukan pekerjaan tersebut. Fungsi ini akan melakukan beberapa pengecekan dan pencarian berdasarkan inputan dari pengguna serta

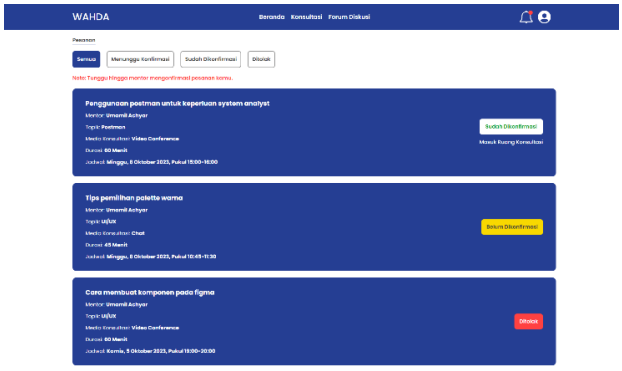
ketersediaan data pengguna di Firebase. Kode fungsi *onAddUserClick* dapat dilihat pada Gambar 4.14.

```
const onAddUserClick = async (userAdded) => {
    try {
      if (userAdded === "") return;
      const foundUser = usersData.find((user) => user.email === userAdded);
      const addedUser = users.find((user) => user === userAdded);

      if (addedUser || user.email === userAdded) {
        setErrorAddUser("Email pengguna sudah dimasukkan");
      } else if (foundUser) {
        setUsers(oldUsers => [...oldUsers, userAdded]);
      } else {
        setErrorAddUser("Email pengguna tidak ditemukan");
      }
    } catch (e) {
      setErrorAddUser(e.toString());
    }
  };
```

Gambar 4.14 Kode penambahan pengguna untuk melakukan konsultasi

4.1.4 Halaman Pesanan

Halaman Pesanan akan ditampilkan saat pengguna mengakses alamat "/pesanan" atau melalui menu navigasi yang ada pada *navbar*. Halaman ini akan menampilkan semua pesanan atau *request* konsultasi yang telah dibuat oleh pengguna atau *request* yang masuk pada mentor. Jika mentor mengunjungi halaman ini maka tiap-tiap *request* yang memiliki status "Belum Dikonfirmasi" akan memiliki tombol "Konfirmasi" dan "Tolak". Pada halaman ini pengguna dapat melakukan *filtering request* atau pesanan yang ada berdasarkan status. Telah dilakukan *sorting* berdasarkan jadwal konsultasi, sehingga jadwal yang terbaru akan ditampilkan terlebih dahulu. Pada halaman ini juga dilakukan pengecekan terhadap pesanan-pesanan yang telah *expired* atau jadwal yang telah lewat dan masih memiliki status "Belum Dikonfirmasi", maka aplikasi akan melakukan perubahan status menjadi "Ditolak" secara otomatis. Saat pesanan telah dikonfirmasi, maka akan ditampilkan tombol yang akan mengarahkan pengguna ke halaman Ruang Konsultasi. Tampilan halaman Pesanan dapat dilihat pada Gambar 4.15 di bawah.



Gambar 4.15 Halaman Pesanan

Kode Halaman Pesanan

Kode halaman Pesanan dapat dilihat pada Gambar 4.16 di bawah. Halaman ini menggunakan *AuthContext* untuk mengambil data pengguna yang *logged in*. *useCollection* untuk mengambil data dari *collection users* dan *requests*. *Collection users* diambil untuk mengetahui siapa mentor dari suatu pesanan atau siapa pengguna yang membuat pesanan jika pengguna yang *logged in* adalah seorang mentor. *useFirestore* digunakan untuk menambahkan data notifikasi saat mentor melakukan penerimaan atau penolakan pesanan, serta saat pesanan telah expired. Untuk filterisasi berdasarkan status, dibuat *state filteredDocument* dan *statusIndex*, dimana saat salah satu tombol filter status ditekan maka akan menjalankan fungsi *onClickStatus* dan dilakukan *filtering* data *requests* berdasarkan status yang dipilih, kemudian data tersebut disimpan ke *filteredDocuments* dan ditampilkan pada halaman.

```
const Pesanan = () => {
  const { user } = useContext(AuthContext);

  const { documents: users, error: errorUsers } = useCollection("users");
  const { addDocument: addNotification } = useFirestore("notifications");
  const { documents: requests, error: errorRequests } = useCollection(
    "requests",
    user?.role === "mentor"
      ? ["mentorId", "==", user.uid]
      : ["uid", "==", user.uid],
    ["createdAt", "desc"]
  );

  const [statusIndex, setStatusIndex] = useState(0);
  const [filteredDocuments, setfilteredDocuments] = useState(null);

  return (
    <section className="container mx-auto flex flex-col pt-20 p-5 gap-5 text-
overlay font-semibold">
      <div className="pt-5 flex">
        <p className="border-b border-primary">
          {user?.role === "mentor" ? "Pesanan Masuk" : "Pesanan"}
```

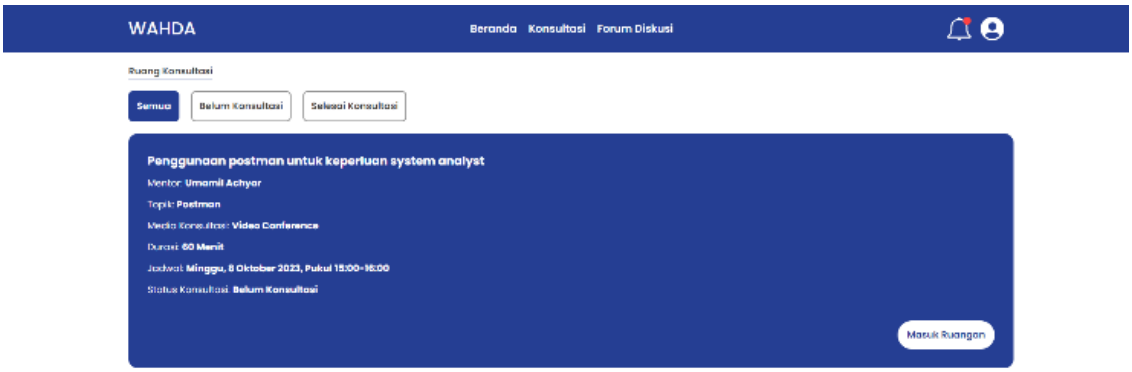
```
        </p>
      </div>

      <div className="flex flex-col gap-5">
        <div className="flex flex-wrap gap-5">
          {STATUSES.map((status, index) => {
            let selectedStatusClass = "text-overlay bg-white border-border";

            if (statusIndex === index) {
              selectedStatusClass = "text-white bg-primary border-primary";
            }

            return (
              <div
                key={status.id}
                className={`${selectedStatusClass}    text-center    font-
semibold rounded-lg p-3 border-2 transition duration-200 cursor-pointer
hover:bg-primary hover:border-primary hover:text-white`}
                onClick={() => onClickStatus(index)}
              >
                {status.status}
              </div>
            );
          })}
        </div>

        {user?.role === "mentor" ? null : (
          <p className="text-danger">
            Note: Tunggu hingga mentor mengonfirmasi pesanan kamu.
          </p>
        )}
      </div>

      {errorRequests && <ErrorMessage message={errorRequests} />}
      {errorUsers && <ErrorMessage message={errorUsers} />}

      {filteredDocuments &&
        users &&
        filteredDocuments.map((request) => buildPesananCard(request))}
    </section>
  );
};
```

Gambar 4.16 Kode Halaman Pesanan

4.1.5 Halaman Ruang Konsultasi

Halaman ruang konsultasi akan ditampilkan saat pengguna mengakses alamat "/ruang-konsultasi" atau melalui menu navigasi yang ada pada *navbar*. Pada halaman ini juga menggunakan *AuthContext* untuk mengambil data user yang nantinya untuk filterisasi data request berdasarkan user tersebut dan beberapa tampilan yang membutuhkan data user. Pengambilan data dilakukan menggunakan useCollection untuk mengambil data dari collections users dan requests. Sama seperti halaman lain pada platform ini, data collections users diambil untuk mengetahui siapa mentor dari suatu pesanan dan siapa pengguna yang

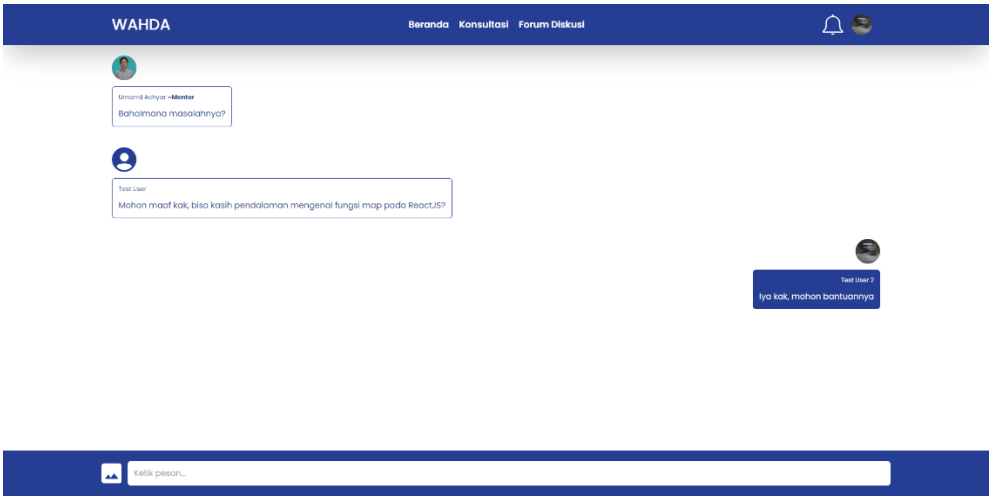
membuat pesanan tersebut, jika mentor yang melakukan *login*. Selain itu pada halaman ini juga menggunakan useFirestore yang digunakan untuk memberi notifikasi kepada *mentee* bahwa sesi konsultasi telah selesai disaat *mentor* telah menekan tombol "Selesai Konsultasi". Halaman ini juga menerapkan filter berdasarkan status. Dalam melakukan filterisasi berdasarkan status, dibuat dua state yaitu filteredDocuments dan statusIndex. Ketika salah satu tombol untuk memfilter status ditekan, maka fungsi onClickStatus akan dijalankan. Fungsi ini bertugas untuk menyaring data permintaan (requests) berdasarkan status yang dipilih. Hasil filter akan disimpan dalam state filteredDocuments dan ditampilkan pada halaman.



Gambar 4.17 Halaman Ruang Konsultasi

4.1.6 Halaman Ruang Chat

Pengguna hanya bisa memasuki halaman chat melalui ruang konsultasi disaat pengguna menekan tombol "masuk ruang konsultasi". Jika user mencoba masuk melalui url "/chat" tanpa melalui pemilihan ruang konsultasi terlebih dahulu, maka user akan diarahkan menuju ruang konsultasi. Di halaman ini pengguna dapat mengunggah media baik berupa teks maupun gambar. Setelah mentor menekan tombol selesai konsultasi pada ruang konsultasi, maka pengguna tidak bisa mengirimkan pesan pada halaman chat tersebut akan tetapi riwayat chat akan tetap tersimpan. Halaman ruang chat dapat dilihat pada Gambar 4.18. Pesan dari pengguna yang *logged in* akan ditampilkan pada posisi sebelah kanan. Untuk pesan pengguna mentor akan ditampilkan tanda "~Mentor" di sebelah namanya untuk memudahkan saat konsultasi dilakukan secara bersama-sama.



Gambar 4.18 Halaman Ruang Chat

Kode Halaman Chat

Seperti halaman lain yang membutuhkan akses data otentikasi pada firebase, halaman ini menggunakan AuthContext yang berfungsi sebagai penanda user siapa yang melakukan pengiriman pesan. Pesan yang terdapat halaman ini diambil dan disimpan di Firebase menggunakan useFirestore. Dengan menggunakan layanan pada Firebase, kode pada halaman chat menjadi lebih simple terlebih untuk menerapkan fitur *live-chat*. Seperti yang dapat dilihat pada gambar 4.19, kode menjadi dua bagian, yaitu untuk menampilkan "chat area" dan form input pesan. Form input pesan dapat ditampilkan dan dihilangkan dengan memanfaatkan ternary operator yang melakukan pengecekan terhadap status konsultasi.

```
return (
  <section className="container mx-auto pt-20 p-5">
    <div className="flex flex-col mt-5 mb-20 gap-10">
      {error && <ErrorMessage message={error} />}
      {documents
        ? documents.map((doc) => (
            <ChatMessage
              key={doc.id}
              message={doc.message}
              isUser={doc.userId === user.uid}
              userId = {doc.userId}
            />
          ))
        : null}
      <div ref={messagesEndRef}></div>
    </div>
    {request?.consultationStatus === "Belum Konsultasi" ? (
      <form
        autoComplete="off"
        style={{ width: "100%" }}
        onSubmit={(event) => onInputSubmit(event)}
```

```
          className="w-full p-5 fixed bottom-0 left-0 bg-primary"
        >
          <div className="container mx-auto flex gap-3">
            <div
              className="flex cursor-pointer"
              onClick={() => fileInputRef.current.click()}
            >
              <input
                type="file"
                name="image"
                id="image"
                className="hidden"
                ref={fileInputRef}
                onChange={onSelectImage}
                accept=".png, .jpg, .jpeg"
              />
              <img src={placeholderIcon} alt="add icon" />
            </div>
            <input
              name="message"
              type="text"
              placeholder="Ketik pesan..."
              className="p-3 w-full bg-white rounded-md"
            />
          </div>
        </form>
      ) : null}
    </section>
  );
```

Gambar 4.19 Kode menampilkan ruang *chat*

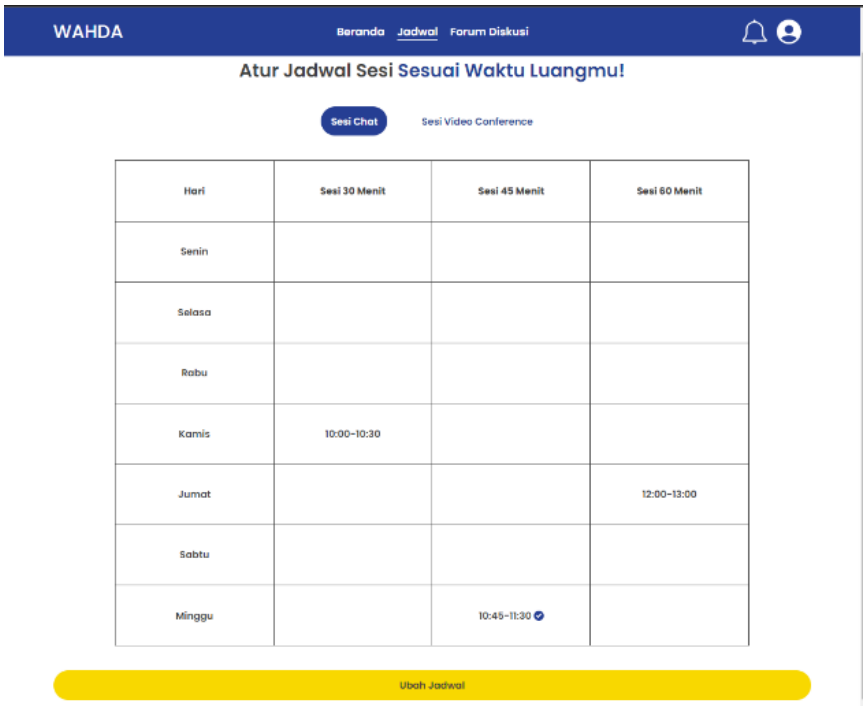
Selain menggunakan Firestore untuk menyimpan data teks pengguna yang telah dikirimkan, layanan Firebase Storage digunakan dalam menyimpan gambar yang telah dikirimkan pengguna melalui halaman chat tersebut. Kode untuk melakukan unggahan gambar dapat dilihat pada Gambar 4.20.

```
    const onSelectImage = async (event) => {
    if (!event.target.files[0]) {
      return;
    }
    const uploadPath = `chats/${user.uid}/${
      doc(collection(firestore, "chats")).id
    }`;
    const storageReference = ref(storage, uploadPath);
    uploadBytes(storageReference, event.target.files[0])
      .then((snapshot) => getDownloadURL(snapshot.ref))
      .then((imageUrl) =>
        addDocument({
          message: imageUrl,
          roomId: state.id,
          userId: user.uid,
        })
      );
  };
```

Gambar 4.20 Kode pengunggahan gambar ke Firebase Storage

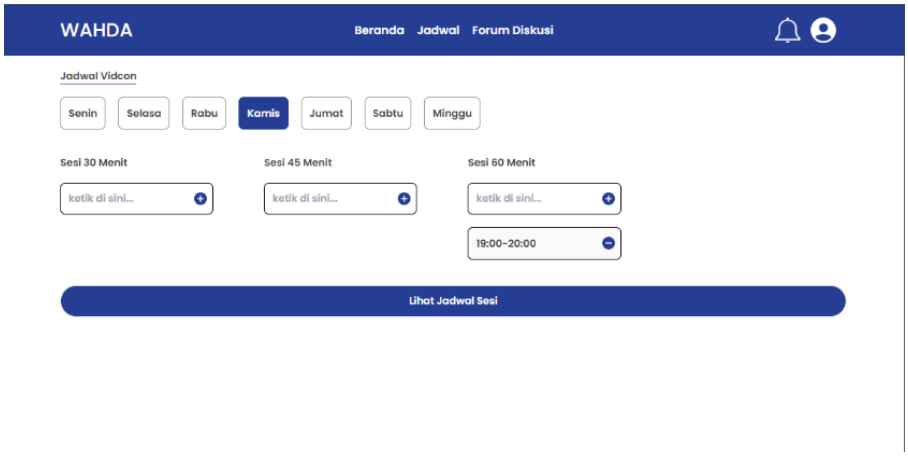
4.1.7 Halaman Jadwal dan Ubah Jadwal

Halaman jadwal hanya dapat diakses oleh pengguna dengan *role mentor* melalui menu pada navabar atau mengunjungi "/jadwal". Pengguna dengan role lain tidaklah bisa mengakses halaman ini, dan jika user mencoba untuk mengakses url tersebut maka akan diarahkan ke halaman *home*. Pada halaman ini mentor dapat melihat jadwal kesediaan konsultasi mereka dalam bentuk tabel dan dua kategori konsultasi yang berbeda. Pada tiap jadwal yang sudah terisi, akan terdapat icon check jika sebuah request konsultasi telah dikonfirmasi oleh mentor dan akan hilang saat mentor menekan tombol "selesai konsultasi" di ruangan konsultasi.



Gambar 4.21 Halaman jadwal konsultasi

Untuk mengakses halaman ubah jadwal pengguna perlu menekan tombol ubah jadwal pada bagian bawah halaman jadwal. Dalam melakukan ubah jadwal pada jenis media konsultasi tertentu, maka diperlukan untuk mengubah jenis konsultasi pada halaman jadwal lalu menekan ubah jadwal. Sehingga jika pengguna mengakses halaman ubah jadwal melalui url "/ubah-jadwal", maka akan diarahkan menuju halaman jadwal.



Gambar 4.22 Halaman tambah jadwal konsultasi

Pada halaman ubah jadwal, akan ditampilkan pilihan hari dan tiga pilihan durasi konsultasi yang user dapat tambahkan untuk menambah jadwal ketersediaan mereka. Selain dapat menambahkan sesi jadwal konsultasi, user juga dapat menghapus sesi suatu konsultasi. Dalam menambahkan sesi konsultasi, user perlu memasukan inputan dengan format HH:MM dan sistem akan otomatis melakukan kalkulasi waktu selesai sesuai dengan sesi durasi yang diinputkan *user*.

Kode Halaman Jadwal dan Ubah Jadwal

Kode pada halaman Jadwal cukup sederhana karena hanya melakukan menampilkan data dan melakukan filterisasi berdasarkan hari dan metode konsultasi yang ada. Dengan alasan tersebut mayoritas kode pada halaman ini berisikan kode UI atau kode HTML dan CSS. Kode untuk halaman jadwal dapat dilihat pada Gambar 4.23.

```
return (
    <section className="container mx-auto flex flex-col gap-10 p-5 pt-20
text-overlay font-semibold overflow-hidden">
      <div className="flex flex-col gap-10 pt-16">
        <h1 className=" text-2xl text-center md:text-3xl">
          Atur Jadwal Sesi{" "}
          <span className="text-primary">Sesuai Waktu Luangmu!</span>{" "}
        </h1>

        <div className="flex justify-center gap-5 md:gap-10">
          <ParagraphButton
            className={`rounded-full ${
              method === "Chat"
                ? "bg-primary text-white"
                : "bg-transparent text-primary"
            }`}
            onClick={() => setMethod("Chat")}
          >
            Sesi Chat
```

```
          </ParagraphButton>

          <ParagraphButton
            className={`rounded-full ${
              method === "Vidcon"
                ? "bg-primary text-white"
                : "bg-transparent text-primary"
            }`}
            onClick={() => setMethod("Vidcon")}
          >
            Sesi Video Conference
          </ParagraphButton>
        </div>
      </div>
       ...
       ...
       ...
```

Gambar 4.23 Kode menampilkan halaman jadwal

Untuk kode pada halaman ubah jadwal, terdapat hal menarik dalaman penerapan penambahan jadwal yaitu dengan dimanfaatkan regex guna pengecekan format inputan jam dengan format HH:MM pada halaman tersebut. Lihat Gambar 4.24 untuk melihat lebih rinci kode dalam menambahkan sesi jadwal pada halaman ubah jadwal

```
const onAddScheduleClick = async () => {
    const finalSchedule = `${addSchedule}-${addMinutesToTime(
      addSchedule,
      parseInt(session.match(/\d+/)[0])
    )}`;

    if (addSchedule.match(formatScheduleRegex)) {
      try {
        await addDocument({
          time: finalSchedule,
          userId: uid,
          day: day,
          session: session,
          method: method,
          status: "Available",
        });

        setAddSchedule("");
      } catch (e) {
        setErrorScheduleValue(e.toString());

        setAddSchedule("");
      }
    } else {
      setErrorScheduleValue(
        "Silakan masukkan jadwal sesuai format: hh:mm (24-hours format)"
      );
    }
  };
```

Gambar 4.24 Kode penambahan jadwal

Disaat user menekan button penambahan, maka fungsi onAddScheduleCheck akan terpanggil. Dalam fungsi ini terdapat beberapa hal yang terjadi. Pertama sistem akan melihat apakah inputan telah sesuai dengan format waktu HH:MM dengan memanfaatkan regex. setelah itu barulah dilakukan perhitungan berdasarkan waktu input oleh user dan durasi sesi yang dimana input dimasukkan. Kode untuk masing-masing fungsi yang disebutkan dan kode menampilkan halaman jadwal dapat dilihat pada gambar 4.25 dan gambar 4.26.

```
const formatScheduleRegex = /^(?:[01]\d|2[0-3]):[0-5]\d$/;

export const addMinutesToTime = (timeString, minutesToAdd) => {
  // Parse the input time string into hours and minutes
  const [hours, minutes] = timeString.split(":").map(Number);

  // Convert hours and minutes to total minutes
  const totalMinutes = hours * 60 + minutes;

  // Add the desired number of minutes
  const newTotalMinutes = totalMinutes + minutesToAdd;

  // Calculate the new hours and minutes
  const newHours = Math.floor(newTotalMinutes / 60);
  const newMinutes = newTotalMinutes % 60;

  // Format the new time as "HH.MM"
  const newTimeString =
    String(newHours).padStart(2, "0") +
    ":" +
    String(newMinutes).padStart(2, "0");

  return newTimeString;
};
```

Gambar 4.25 Kode fungsi dalam penerapan penambahan jadwal

```
 return (
    <div className="flex flex-col items-stretch gap-5">
      <p>{session}</p>
      <div className="relative">
        <input
          type="text"
          placeholder="ketik di sini..."
          className="appearance-none p-3 border border-black w-full rounded-
lg opacity-0.5"
          onChange={(event) => {
            setAddSchedule(event.target.value);

            setErrorScheduleValue("");
          }}
          value={addSchedule}
          onKeyUp={(event) =>
            event.key === "Enter" ? onAddScheduleClick() : null
          }
        />
```

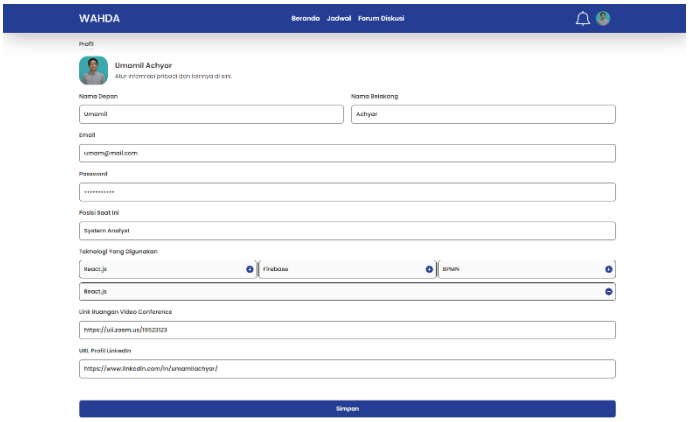
```
        <img
          src={addIcon}
          alt="add"
          className="absolute top-1/2 right-0 -translate-x-1/2 -translate-
y-1/2 cursor-pointer"
          onClick={onAddScheduleClick}
        />
      </div>

      {sortedDocuments?.map((doc) => (
        <div key={doc.id} className="relative">
          <input
            type="text"
            placeholder=""
            className="appearance-none  p-3  border  border-black  w-full
rounded-lg opacity-0.5"
            value={doc.time}
            disabled
          />
          <img
            src={minIcon}
            alt="remove"
            className="absolute top-1/2 right-0 -translate-x-1/2 -translate-
y-1/2 cursor-pointer"
            onClick={() => onRemoveScheduleClick(doc.id)}
          />
        </div>
      ))}
    </div>
  );
```

Gambar 4.26 Kode menampilkan halaman ubah jadwal

4.1.8 Halaman Profile

Pada dasarnya, halaman profil memiliki fungsi utama untuk mengelola informasi pribadi, termasuk pengubahan username dan password. Namun, di sisi seorang mentor, penggunaan halaman profil juga meliputi penyesuaian tautan Zoom untuk sesi konsultasi video conference dan penambahan atau pengurangan teknologi yang dikuasai.

Mentor dapat menambahkan teknologi yang digunakan sesuai pilihan yang ada, dimana pilihan tersebut dimasukkan oleh pengguna admin, dan pilihan-plihan tersebut ditampilkan pada halaman “Konsultasi” yang menampilkan daftar mentor.

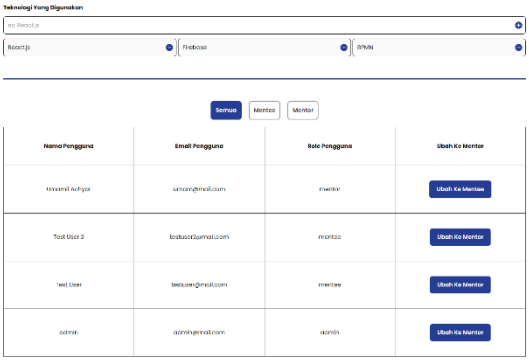


Gambar 4.27 Halaman profile

4.1.9 Halaman Admin

Halaman admin diakses melalui url "/admin". Pada halaman admin dilakukan pengecekan terhadap *role* pengguna yang *login*. *Role* yang diizinkan untuk mengakses halaman ini adalah "admin".

Pada halaman ini, admin dapat mengubah *role* dari pengguna mentee ke mentor. Selain itu, admin juga dapat menambah dan menghapus teknologi yang nantinya dapat dipilih oleh mentor serta akan menjadi *filter* pada tampilan daftar mentor.



Gambar 4.28 Halaman admin

4.2 Pengujian Sistem

Pada tahap pengujian, platform Wahda akan diuji dengan beberapa skenario untuk memastikan fungsionalitas telah berjalan sesuai dengan harapan. Pengujian akan dilakukan dengan metode *Black box Testing*, yaitu sebuah metode yang dipakai untuk menguji sebuah *software* tanpa harus memperhatikan detail dari *software* tersebut [10]. Hasil dari pengujian dapat dilihat pada tabel 4.1.

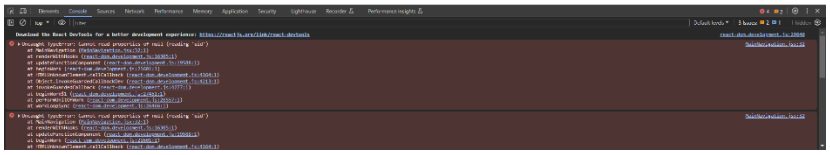
Tabel 4.1 Tabel pengujian platform Wahda

No	Skenario	Hasil yang diharapkan	Hasil
1	Pengguna mengunjungi halaman sesuai dengan *state authentication*, seperti *logged in*, belum *logged in*, dan *logged in mentor*.	Pengguna ditampilkan halaman sesuai dengan keadaan autentikasinya	Berhasil
2	Pengguna mengunjungi halaman yang hanya boleh diakses oleh keadaan autentikasi tertentu seperti, halaman jadwal, halaman *login* dan daftar, dan lainnya	Pengguna akan di-redirect sesuai mekanisme pengecekan keadaan autentikasi per halaman	Berhasil
3	Pengguna *mentee* melakukan *action* navigasi maupun pengisian suatu *field*, seperti *login*, daftar, mengisi *request* konsultasi berdasarkan *mentor* yang dipilih, dan sebagainya	Pengguna berhasil melakukan action yang sesuai	Berhasil
4	Pengguna melakukan kesalahan input baik berupa validasi	Pengguna ditampilkan pesan *error* yang sesuai	Berhasil
5	Pengguna melakukan *action* yang spesifik dengan keadaan autentikasi maupun data yang terasosiasi dengan akun tersebut seperti mengunjungi halaman pesanan atau ruang konsultasi, masuk ke ruang konsultasi tertentu, melakukan *filtering* status pesanan, dan sebagainya	Pengguna berhasil melakukan *action* yang dilakukan	Berhasil
6	Pengguna masuk ke ruang konsultasi dengan media *Chat* dan status konsultasi telah selesai	Pengguna tidak dapat memasukkan pesan	Berhasil
7	Pengguna masuk ke ruang konsultasi dengan media *Chat*	Pengguna ditampilkan *bubble* pesan yang sesuai dengan siapa yang mengirim pesan	Berhasil
8	Pengguna *mentee* melakukan request konsultasi pada suatu *mentor* tertentu	Pengguna ditampilkan halaman *request* dengan data jadwal berdasarkan mentor tersebut, dan hari dimana request dilakukan	Berhasil
9	Pengguna *mentor* dapat mengunjungi halaman jadwal dan mengubah/menambah sesi jadwal	Pengguna ditampilkan halaman jadwal sesuai dengan jadwal yang telah diisi dan dibagi berdasarkan sesi waktu serta metode konsultasi serta mengubah/menambah sesi jadwal mereka	Berhasil

Berdasarkan hasil dari pengujian yang telah dilakukan, dapat disimpulkan bahwa seluruh fungsionalitas di *platform* Wahda telah berjalan sesuai dengan yang diharapkan. Dengan hasil pengujian yang terdapat pada tabel 2, maka *platform* Wahda telah dapat digunakan oleh pengguna umum yang membutuhkan layanan konsultasi pada *platform* ini.

4.2.1 Kendala yang Ditemukan dalam Pengujian

Pada saat melakukan pengujian, masalah yang paling sering ditemukan yaitu *error* karena variabel atau data yang diperlukan saat melakukan *render null*. Hal tersebut dikarenakan React melakukan proses *render* secara langsung, sedangkan data yang diperlukan untuk di-*render* atau ditampilkan masih dalam proses *fetch* ke Firebase sehingga menyebabkan *error*. React sebenarnya dapat meng-*handle* jika terdapat data *null* yang ingin di-*render*. *Error* yang sering muncul adalah saat data tersebut diolah seperti *filter* atau *sort*. *Error* tersebut dapat menyebabkan *crash* aplikasi, seperti yang dapat dilihat pada Gambar 4.28 di bawah. Pada gambar di bawah, *error* tersebut terjadi karena ingin dilakukan akses untuk *field* "uid" pada data *object user*. *Error* tersebut dapat diatasi dengan menambahkan pengecekan seperti pada Gambar 4.29, dimana ditambahkan pengecekan dengan sintaks *ternary* dari Javascript, sehingga jika *user null* maka nilainya adalah *string* kosong atau "".



Gambar 4.29 *Error Null Safety*

```
const { documents } = useCollection(
  "notifications",
  ["uid", "==", user ? user.uid : ""],
  ["createdAt", "desc"]
);
```

Gambar 4.30 Solusi *Error Null Safety*

BAB V
PENUTUP

5.1 Kesimpulan

Selama pengembangan yang telah dilakukan, mulai dari tahap analisis kebutuhan hingga implementasi, untuk mengatasi masalah dalam membangun sebuah platform konsultasi pembelajaran pemrograman *online* maka digunakan sebuah teknologi yang mutakhir, yaitu ReactJs dan Firebase. Berdasarkan tahapan yang telah dibahas dapat diambil kesimpulan sebagai berikut:

a) Proses pengembangan platform Wahda, sebagai platform untuk melakukan konsultasi pembelajaran pemrograman online, telah berhasil dikembangkan dengan menggunakan metode Waterfall.
b) Penggunaan atas *library* ReactJs dapat mempercepat pengembangan sebuah aplikasi *startup* dan mensimplifikasikan kode dalam membangun antarmuka melalui konsep *component* yang ada didalamnya. Selain itu, menggunakan ReactJs juga telah membuat kode antarmuka lebih modular dan mengurangi redudansi, yaitu dengan membuat satu file *component* untuk sebuah bagian dari antarmuka yang nantinya dapat digunakan pada *file* lain dalam suatu *project*.
c) Penggunaan atas Firebase sebagai *backend* dari platform Wahda telah mempercepat pengembangan pembuatan aplikasi sehingga juga telah mempercepat waktu untuk melakukan deployment aplikasi yang telah dibuat. Selain itu, layanan Firebase juga telah memudahkan pengembang dalam hal penyimpanan data dan otentikasi yang ada pada platform ini.
d) Dukungan integrasi yang *seamless* dari ReactJs dan Firebase telah memudahkan pengembang dalam pembuatan fitur yang kompleks pada platform Wahda, seperti *live chat* dan notifikasi.
e) Berdasarkan hasil pengujian dengan metode *black box* yang telah dilakukan, maka dapat disimpulkan bahwa seluruh fungsionalitas platform Wahda telah berjalan dengan lancar tanpa ada bug.

5.2 Saran

Selama melakukan masa pengembangan platform Wahda yang telah dilakukan, maka penulis memiliki beberapa saran yang dapat diterapkan untuk pengembangan lebih lanjut untuk pengembangan berikutnya atau penelitian terkait di masa yang akan datang, yaitu:

a) Melakukan simplifikasi lebih lanjut mengenai alur dalam memasuki ruangan konsultasi, untuk sekarang terdapat halaman ruang konsultasi untuk menampung segala pesanan konsultasi yang diterima, akan tetapi hal ini tidak diperlukan karena halaman pesan masuk sudah cukup untuk dapat langsung memasuki ruangan konsultasi.
b) Realisasi fitur diskusi yang dapat berguna sebagai tempat pembahasan yang dapat dilihat secara umum.
c) Menerapkan enkripsi untuk data *livechat* yang dikirimkan ke *database*, hal ini dilakukan untuk menjaga keamanan dan privasi pengguna dalam menggunakan fitur *live chat* pada platform Wahda. Sehingga data yang dikirimkan pengguna tidak dapat dilihat oleh orang lain selain pengguna dan lawan bicara pengguna.

DAFTAR PUSTAKA

Coursera. (2023, July 24). *What Is Programming? And How To Get Started*. Retrieved from coursera: https://www.coursera.org/articles/what-is-programming

Dicoding. (2020, November 25). *Apa itu Firebase? Pengertian, Jenis-Jenis, dan Fungsi Kegunaannya*. Retrieved from Dicoding.com: https://www.dicoding.com/blog/apa-itu-firebase-pengertian-jenis-jenis-dan-fungsi-kegunaannya/

Firebase. (2023, September 21). *Cloud Firestore*. Retrieved from firebase.google.com: https://firebase.google.com/docs/firestore?hl=id

Firebase. (2023, September 13). *Firebase Authentication*. Retrieved from firebase.google.com: https://firebase.google.com/docs/auth?hl=id

Hendrik, Anjomshooa, A., & Tjoa, A. M. (2014). Towards Semantic Mashup Tools For Big Data Analysis. *Proceding of the Information & Communication Technology-EurAsia Conference 2014*, (pp. 100-145). Bali.

Herbert, D. (2022, Juni 27). *What is React.js? (Uses, Examples, & More)*. Retrieved from hubspot.com: https://blog.hubspot.com/website/react-js

Masyruhatin, S., Mursityo, Y. T., & Pramono, D. (2019). Pengembangan Sistem Informasi Penilaian Hasil Belajar Siswa berbasis Web pada SMA Brawijaya Smart School. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2-3.

Nasution, & Iswari, L. (2021). Penerapan React JS Pada Pengembangan FrontEnd Aplikasi Startup Ubaform. *Automata*, 1.

Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., & Abrahamsson, P. (2014). Software Development in Startup Companies: A Systematic Mapping Study. *ScienceDirect*, 3.

Pramudita, R., Arifin, R. W., Alfian, A. N., Safitri, N., & Anwariya, S. D. (2021). PENGGUNAAN APLIKASI FIGMA DALAM MEMBANGUN UI/UX YANG INTERAKTIF PADA PROGRAM STUDI TEKNIK INFORMATIKA STMIK TASIKMALAYA. *Jurnal Buana Pengabdian*, 150.

Prihadi, S. D. (2015, Maret 29). *Satu dari Sepuluh Siswa di Indonesia Ingin Belajar Coding*. Retrieved from cnnindonesia: https://www.cnnindonesia.com/teknologi/20150328054245-185-42518/satu-dari-sepuluh-siswa-di-indonesia-ingin-belajar-coding

Rizaty, M. A. (2022, Agustus 9). *Kebutuhan Pekerja IT Indonesia Hampir Capai 2 Juta pada 2025*. Retrieved from dataindonesia: https://dataindonesia.id/tenaga-kerja/detail/kebutuhan-pekerja-it-indonesia-hampir-capai-2-juta-pada-2025

Ronen, E. (n.d.). *React Context API: What is it and How it works?* Retrieved from loginradius.com.

Setiawan, A. M. (2013). *Integrated Framework For Business Process Complexity Analysis*. Retrieved from ECIS 2013 Completed Research: http://aisel.aisnet.org/ecis2013_cr/49

Singhal, P. (2023, April 18). *Frontend vs Backend*. Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/frontend-vs-backend/

Sufiyan, T. (2023, April 10). *ReactJS State: SetState, Props and State Explained*. Retrieved from simplilearn.com: https://www.simplilearn.com/tutorials/reactjs-tutorial/reactjs-state

Syamsudin, A. (2020). Analisis Kesalahan Coding Pemrograman Java pada Matakuliah Algoritma Pemrograman Mahasiswa Tadris Matematika IAIN Kediri. *Jurnal Fakta Arbiyah*, 105.

Taufiq, H. (2015). *Argumentasi dan Validitas.* Yogyakarta: Darqin.

Thomas, M. (2014). *A Perfect Startup Team The Golden Triangle*. Retrieved from Quora: https://kmvidgxgosnelleb.quora.com/A-Perfect-Startup-Team-The-Golden-Triangle

Wahid, F. (2014). The Antecedents And Impacts of a Green Eprocurement Infrastructure: Evidence From The Indonesian Public Sector. *International Journal of internet Protocol Technology, 7*(4), 210-218.

Zukhri, Z. (2014). *Algoritma Genetika: Metode Komputasi Evolusioner untuk Menyelesaikan Masalah Optimasi.* Yogyakarta: Andi Publisher.

LAMPIRAN