IMPLEMENTASI MONGO DB, EXPRESS JS, REACT JS DAN NODE JS (MERN) PADA PENGEMBANGAN APLIKASI FORMULIR, KUIS, DAN SURVEI *ONLINE*

Disusun Oleh:

N a m a : Nasution

NIM : 17523199

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2021

HALAMAN PENGESAHAN DOSEN PEMBIMBING

IMPLEMENTASI MONGO DB, EXPRESS JS, REACT JS DAN NODE JS (MERN) PADA PENGEMBANGAN APLIKASI FORMULIR, KUIS, DAN SURVEI *ONLINE*

TUGAS AKHIR

Disusun Oleh:

N a m a : Nasution

NIM : 17523199

Yogyakarta, 07 Januari 2022

Pembimbing,

(Lizda Iswari, S.T., M.Sc.)

HALAMAN PENGESAHAN DOSEN PENGUJI

IMPLEMENTASI MONGO DB, EXPRESS JS, REACT JS DAN NODE JS (MERN) PADA PENGEMBANGAN APLIKASI FORMULIR, KUIS, DAN SURVEI *ONLINE*

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, 07 Januari 2022

Tim Penguji

Lizda Iswari, S.T., M.Sc. _____

Anggota 1

Andhika Giri Persada, S.Kom., M.Eng. _____

Anggota 2

Rian Adam Rajagede, S.Kom., M.Cs. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Nasution

NIM : 17523199

Tugas akhir dengan judul:

IMPLEMENTASI MONGO DB, EXPRESS JS, REACT JS DAN NODE JS (MERN) PADA PENGEMBANGAN APLIKASI FORMULIR, KUIS, DAN SURVEI *ONLINE*

Menyatakan bahwa seluruh komponen 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, 07 Januari 2022

(Nasution)

HALAMAN PERSEMBAHAN

Bismillahirrohmaanirrohim

Alhamdulillahi Robbil `Alamin, segala bentuk puja, puji dan rasa syukur atas segala limpahan rahmat, nikmat serta karunia yang tidak terhingga dari Allah SWT yang salah satunya adalah penulis dapat menyelesaikan tugas akhir dengan baik dan lancar. Perjuangan memang sulit apabila dilakukan secara individu dan untuk itulah keberadaan entitas lain dalam hidup membantu melewati berbagai macam kesulitan yang ada.

Oleh karena itu penulis mempersembahkan karya ini kepada orang-orang yang penulis sayangi yang telah berjuang, berdoa serta membantu dalam terciptanya karya ini:

1. Kedua orang tua penulis, Bapak Samsudin dan Ibu Wahida yang sangat penulis sayangi, cintai dan hormati. Tidak ada kata yang mewakili perjuangan, pengorbanan serta bantuan yang telah kalian berikan kepada penulis dari waktu penulis kecil hingga sekarang. Yakinlah tanpa semua itu penulis tidak akan mampu sampai pada titik yang sekarang ini. Walau kata ini tidak akan mampu membalas begitu banyak hal yang telah kalian berikan tapi penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada kalian berdua.
2. Saudara/saudari kandung penulis, kakak Lilis Sulastri dan adik Dzakir Khafadi yang penulis sayangi dan hormati. Terima kasih telah menjadi orang-orang terdekat untuk mencurahkan segala hal baik itu kesenangan dan kesedihan terutama dalam proses terciptanya karya ini.
3. Keluarga besar, sahabat, teman, guru, mentor dan dosen yang penulis hormati, terima kasih sudah memberi dukungan, semangat dan motivasi kepada penulis serta telah hadir mewarnai hidup penulis.
4. Ibu Lizda Iswari, S.T., M.Sc. selaku dosen pembimbing tugas akhir penulis, terima kasih atas segala bentuk bimbingan, arahan, nasihat, motivasi dan ilmu yang luar biasa yang telah ibu berikan baik secara langsung maupun tidak langsung serta kesabaran atas segala bentuk kekeliruan penulis dalam proses pembelajaran.
5. Beasiswa Unggulan Kementrian Pendidikan dan Kebudayaan yang telah begitu banyak membantu finansial penulis dalam menempuh pendidikan pada jenjang Strata 1 (S1) baik untuk pembiayaan kuliah serta biaya hidup selama empat tahun ini.

6. Jurusan Informatika Teknologi Industri Universitas Islam Indonesia yang telah menjadi wadah terbentuknya karakter dan pribadi penulis.

HALAMAN MOTO

خَيْرُ الناسِ أَنفَعُهُم لِلنَّاسِ

"Sebaik-baik manusia adalah yang paling bermanfaat bagi manusia lain"
(Hadits Riwayat ath-Thabrani, Al-Mu'jam al-Ausath, juz VII, hal. 58, dari Jabir bin Abdullah r.a.. Dishahihkan Muhammad Nashiruddin al-Albani dalam kitab: As-Silsilah Ash-Shahîhah)

"Jangan katakan tidak mungkin kepadaku, sebelum kau mati dalam mencobanya"
(Sultan Muhammad Al-Fatih)

"Setinggi-tinggi ilmu, semurni-murni tauhid, sepintar-pintar siasat"
(Oemar Said Tjokroaminoto)

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakaatuh

Alhamdulillahi Robbil `Alamin. Penulis ucapkan puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya kepada penulis sehingga penulis diberikan kekuatan dan kemudahan dalam menyelesaikan tugas akhir ini. Sholawat dan salam penulis ucapkan kepada Manusia terbaik Baginda Nabi Muhammad SAW dengan hantaran kata Allahumma Sholli'alaa Sayyidina Muhammad Wa'alaa aali Sayyidina Muhammad. Dengan meneladani semangat Rasulullah SAW dalam menimba ilmu dan mengamalkannya sehingga memotivasi penulis dalam menyelesaikan tugas akhir ini yang berjudul **"IMPLEMENTASI MONGO DB, EXPRESS JS, REACT JS DAN NODE JS (MERN) PADA PENGEMBANGAN APLIKASI FORMULIR, KUIS, DAN SURVEI *ONLINE*"** ini dengan baik dan lancar.

Laporan tugas akhir ini ditujukan sebagai salah satu syarat dalam menyelesaikan pendidikan pada jenjang Strata 1 (S1) Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Manajemen waktu antara kerja, organisasi dan kuliah menjadi faktor yang menyulitkan penulis dalam proses terciptanya laporan tugas akhir ini sehingga penulis sadar bahwa laporan tugas akhir ini tidak akan mungkin ada dan bisa selesai tanpa dukungan serta motivasi dari berbagai pihak, selain rasa syukur dan terima kasih kepada Allah SWT juga izinkan penulis untuk menyampaikan rasa terima kasih yang sedalam-dalamnya kepada kepada:

1. Kedua orang tua penulis, Bapak Samsudin dan Ibu Wahida untuk segala doa dan dukungan secara moril, materi, non-materi, kepercayaan dan kasih sayang serta motivasi yang selalu diberikan baik kepada penulis selama ini.
2. Saudara/saudari kandung penulis, kakak Lilis Sulastri dan adik Dzakir Khafadi yang penulis sayangi dan hormati.
3. Keluarga besar, sahabat, teman, guru, mentor dan dosen yang penulis hormati.
4. Bapak Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
5. Bapak Prof. Dr. Ir. Hari Purnomo, M.T., selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
6. Bapak Hendrik, S.T., M.Eng., selaku Ketua Jurusan Informatika Universitas Islam Indonesia

7. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Program Studi Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
8. Ibu Lizda Iswari, S.T., M.Sc. selaku dosen pembimbing tugas akhir penulis. Terima kasih atas segala bentuk bimbingan, arahan, nasihat, motivasi dan ilmu yang luar biasa yang telah ibu berikan baik secara langsung maupun tidak langsung serta kesabaran atas segala bentuk kekeliruan penulis dalam proses pembelajaran.
9. Bapak Hari Setiaji, S.KOM., M.Eng. selaku dosen pembimbing akademi (DPA) penulis.
10. Rekan-rekan Tim Ubaform, yang telah mau berjuang bersama dan menemani perjalanan penjaluran perintisan bisnis.
11. Rekan-rekan Organisasi dan UKM yang pernah penulis ikuti HAWASI UII, INVOSE, PSC dan Unisi Cyber Guard, yang telah mau berjuang bersama dan belajar bersama baik dalam dakwah yang saling meningatkan serta menasehati dalam kebaikan maupun dalam pemahaman ilmu akademik sehingga penulis mampu menjadi pribadi yang sekarang ini.
12. Beasiswa Unggulan Kementrian Pendidikan dan Kebudayaan yang telah begitu banyak membantu finansial penulis dalam menempuh pendidikan pada jenjang Strata 1 (S1) baik untuk pembiayaan kuliah serta biaya hidup selama empat tahun ini.
13. Seluruh pihak yang tidak bisa penulis sebutkan satu per satu, terima kasih atas semua bentuk dukungannya.

Semoga atas segala bentuk dukungan dan doa serta bimbingan yang penulis terima selama ini menjadi nilai ibadah dan dibalas oleh Allah SWT. Saya memohon maaf sebesar-besarnya apabila selama mengerjakan Tugas Akhir terdapat banyak kesalahan. Saya menyadari akan keterbatasan kemampuan yang saya miliki. Semoga laporan ini dapat bermanfaat bagi semua yang membacanya terutama bagi penulis sendiri.

Yogyakarta, 07 Januari 2022

(Nasution)

SARI

Pengembangan aplikasi pembuatan formulir, kuis dan survei *online* ini tentunya memiliki urgensi untuk menciptakan sebuah layanan yang dapat membantu mempermudah pembuatan formulir, kuis dan survei dalam satu layanan dengan beberapa keunggulan tampilan yang dapat dikostumisasi sesuai kebutuhan. Tentunya layanan seperti ini telah ada, namun tidak mencakup ke tiga fitur tersebut serta tampilan yang ada sekarang masih kaku. Oleh karena itu pengembangan aplikasi pembuatan formulir, kuis dan survei datang dengan membawa inovasi terhadap tampilan yang lebih modern. Untuk mengembangkan layanan tersebut tentunya membutuhkan teknologi yang mampu menunjang segala kebutuhan pengembagan baik dari segi kemudahan maupun performa yang dihadirkan.

Dalam beberapa tahun terakhir pengembangan web *(web development)* telah mengalami perubahan besar pada setiap lapis bagian *(stack)* pembutan web. Lapis bagian ini dapat dikategori secara umum dengan bagian *front-end* dan *back-end*. Munculnya ECMAScript2015 dan perkembangan pesat *database* NoSQL telah memunculkan paradigma baru "Javascript everywhere" yang mana hal ini menyebabkan penggunaan javascript meningkat dan menyebar dengan cepat terutama dikalangan *web developer*. MERN merupakan singkatan dari teknologi MongoDB, Express JS, React JS, dan Node JS yang merupakan salah satu metode pengembangan web yang menggunakan javascript sebagai bahasa pemrograman untuk secara keseluruhan pada setiap lapisan atau bagian pengembangan web baik itu pada bagian *front-end* maupun *back-end*. Fungsi utama dari keempat teknologi ini adalah Mongo DB berperan sebagai database yang digunakan untuk menyimpan data, Express JS yang digunakan untuk mengembangan API sisi *backend* sehingga data dapat diakses dari *frontend*, React JS berperan sebagai *library* untuk mengembangakan *user interfaces* berbasis SPA dan Node JS yang memiliki fungsi sebagai sebuah *environment* pengembangan dalam rangka memudahkan *developer* dalam melakukan instalasi *library* pendukung, debug dan lain sebagainya.

Dalam rangka mengembangkan layanan pembuatan formulir, kuis dan survei pada skripsi ini akan berfokus terhadap implementasi teknologi MERN yang digunakan untuk pengembangan baik terhadap performa yang dihasilkan, kemudahan serta efisiensi waktu yang dihadirkan dari implementasi teknologi tersebut. Selain itu, pengembangan akan terus dilakukan sampai pada tahap *deployment* menggunakan platform MongoDB atlas dan Heroku.

Kata kunci: MERN *stack*, MongoDB, Express JS, Node JS, React JS.

GLOSARIUM

API	*Application programming interfaces*. Sebuah set *interface* yang dapat menghubungkan aplikasi yang satu dengan aplikasi lain atau sebagai perantara antar berbagai aplikasi yang berbeda baik dalam satu platform yang sama atau lintas platform.
Back-end	Bagian belakang sebuah situs web yang berisikan fungsionalitas sistem yang dapat memproses setiap aktivitas yang dilakukan oleh pengguna situs web.
Framework	Sebuah kerangka program yang digunakan untuk membantu developer untuk mengembangkan kode secara konsisten
Front-end	Bagian dari situs web yang dapat dilihat langsung oleh pengguna. Sering juga disebut dengan *user interface* (antarmuka pengguna).
Library	Kode program yang dikembangkan oleh orang lain, sebagai tambahan untuk melengkapi atau digunakan secara instan dalam menjalankan dan melengkapi fitur tertentu.
Hosting	Tempat untuk menyimpan segala macam kebutuhan website baik berupa kode program, gambar, dan lain sebagainya agar dapat diakses secara *online* di internet secara publik.
Domain	sebuah nama untuk mengidentifikasi sebuah jaringan tanpa menggunakan *internet protocol* (IP).
RDBMS	*Relational Database Management System*. Perangkat lunak untuk memelihara, membuat *query*, dan memperbarui data dan metadata dalam *database* model relasional.
SQL	*Structured Query Language*. Bahasa yang digunakan untuk pemrograman dan mengelola data dalam *database* relasional (RDBMS)
Startup	Perusahaan rintisan yang bergerak untuk menyelesaikan sebuah masalah yang ada dengan solusi tertentu. *Startup* biasanya perusahaan yang belum lama beroperasi dan erat kaitannya dengan teknologi.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan *startup* baru di Indonesia bahkan di dunia setiap tahun meningkat pesat, hal ini sesuai dengan hasil riset dari DailySocial dengan judul *Startup Report 2020 Business Resiliency during the pandemic* disebutkan bahwa Jakarta menempati urutan kedua dari 100 kota di dunia dalam daftar ekosistem *startup* yang sedang berkembang. Tidak hanya itu Masih merujuk pada riset yang sama Jakarta menempati posisi teratas posisi dengan nilai ekosistem $26,3 miliar, diikuti oleh Guangzhou ($19,2 miliar) dan Kuala Lumpur ($15,3 miliar). Hal ini patut untuk dipertimbangkan bahwasanya dalam perintisan *startup* akan terdapat banyak sekali kompetitor yang sama dalam proses bisnis yang sama. Meski begitu dalam hal ini perintis harus mampu menciptakan nilai atau *value* unggul baru yang dapat menjadi nilai jual sehingga mampu memenangkan persaingan pasar. Selain menciptkan *value* baru ada faktor lain yang mempengaruhi persaingan *startup* salah satu nya adalah dengan meningkatkan performa layanan yang disediakan.

Performa sebuah sistem atau aplikasi memiliki pengaruh tersendiri dalam kenyamanan kostumer dalam menggunakan layanan yang disediakan bahkan menurut Kinsta salah satu layanan web hosting mengatakan 1 dari 4 pengunjung akan meninggalkan website jika memuat lebih dari 4 detik dan 46% pengguna tidak mengunjungi kembali situs web dengan performa yang buruk (KINSTA, 2021). Selain itu, sebuah studi oleh Walmart telah mengungkapkan bahwa setiap penurunan 100 ms dalam waktu pemuatan menghasilkan pertumbuhan 1% dalam pendapatan tambahan. Dalam studi lainnya juga dilakukan oleh salah satu situs keamanan digital Akamai pada tahun 2017, yang mana studi tersebut menyatakan bahwa pada saat situs diakses dan kemudian terdapat penundaan *loading* yang lebih lama dari biasanya yaitu setiap 100 milidetik akan mengurangi tingkat konversi sebesar 7%, tingkat konversi yang dimaksud adalah penurunan penjualan akibat akses situs yang lama (Einav, 2019).

Berdasarkan beberapa studi tersebut dapat ditarik kesimpulan bahwa performa sangat penting untuk memenangkan persaingan dengan kompetitor. Oleh karena itu dalam pengembangan *startup* Ubaform telah kami lakukan beberapa perbandingan terhadap teknologi yang digunakan dalam pengembangan *startup* Ubaform dalam rangka menciptakan performa yang cepat sehingga dalam hal ini perancangan dan pengembangan dapat dilakukan secara

tepat. Pengembangan yang terencana secara sistematik akan meminimalisir kesalahan-kesalahan yang mungkin akan terjadi saat proses pengembangan. Dalam pengembangan *startup* Ubaform perancangan terencana mengenai teknologi yang akan digunakan telah dipertimbangkan secara matang. Selain untuk mengurangi kesalahan, merancang pengembangan yang matang dapat membawa dampak pada kemudahan pada proses pengembangan *startup*. Rancangan tersebut diharapkan dapat mampu untuk menjawab kebutuhan dari pengembangan web Ubaform seperti bagaimana penyimpanan data yang dibutuhkan, fleksibilitas penggunaan data, data yang terstruktur, performa yang cepat dan lain sebagainya. Banyak teknologi yang telah ada yang dapat digunakan dalam pengembangan sebuah startup berbasis website. Hal ini perlu dipertimbangkan dan direncanakan dengan sebaik mungkin untuk kenyamanan pengguna dan untuk kemudahan serta kecepatan dalam pengembangan *startup*. Selain itu teknologi yang digunakan haruslah mampu relevan untuk beberapa tahun kedepan. Hal ini akan berdampak pada biaya yang nantinya akan dikeluarkan oleh perusahaan, biasa nya untuk mempelajari dan menerapkan teknologi yang lebih baru membutuhkan pengeluaran yang lebih besar.

Berdasarkan hal tersebut dalam perintisan *startup* Ubaform saat ini mengimplementasikan MERN (MongoDb, Express JS, React JS, Node JS) sebagai teknologi yang digunakan dalam pengembangan. Pemilih teknologi disesuaikan dengan kebutuhan startup Ubaform sebagai layanan pembuatan formulir, kuis dan survei yang membutuhkan performa yang cepat, penyimpanan data terstruktur, data yang fleksibel, kemudahan serta kecepatan pengembangan. Dari 4 (empat) teknologi tersebut dapat dikategorikan menjadi 2 (dua) *stack* yaitu *frontend* dan *backend.* Untuk *frontend* yang akan menerapkan SPA *(Single Page Application)* memilih teknologi React JS dirasa cocok untuk mendukung hal tersebut. Penggunaan SPA telah memiliki banyak dukungan pengembangan seperti *library* pendukung, komunitas, dokumentasi kode dan juga banyak perusahaan besar yang sudah beralih menggunakan SPA sehingga akan ada banyak pengembang yang memiliki skill pada bidang tersebut. Penerapan SPA berkaitan erat dengan performa startup Ubaform karena SPA memuat sebagian besar sumber daya aplikasi (HTML/CSS/Javascript) hanya sekali. SPA tidak perlu memuat ulang seluruh halaman tetapi hanya sebagian data tertentu yang dibutuhkan pengguna sehingga akan mengurangi request dari frontend ke sisi server. Kemudian untuk *backend* implementasi Express JS dan untuk database menggunakan MongoDb serta untuk mendukung ketiga teknologi tersebut tentunya adalah Node JS.

1.2 Rumusan Masalah

Berdasarkan uraian dari latar belakang, maka ada beberapa masalah yang dapat dirumuskan, antara lain:

1. Bagaimana implementasi MERN pada pengembangan aplikasi *startup* Ubaform dapat mempercepat proses pengembangan *startup*.
2. Bagaimana implementasi MERN pada pengembangan aplikasi *startup* Ubaform dapat memberikan kemudahan bagi developer.
3. Bagaimana implementasi MERN pada pengembangan aplikasi *startup* Ubaform dapat meningkatkan performa atau kinerja *startup*

1.3 Batasan Masalah

Agar identifikasi masalah yang akan dibahas lebih jelas, maka dibuatlah batasan-batasan masalah sebagai berikut:

1. Implementasi menggunakan MERN (MongoDB, Express JS, React JS, dan Node JS).
2. Pengimplementasian MERN *stack* diterapkan pada *startup* Ubaform.
3. Data yang digunakan untuk RESTful API merupakan data *dummy.*
4. Penelitian ini hanya berfokus pada implementasi MERN di beberapa fitur *startup* aplikasi Ubaform seperti fitur autentikasi, manajemen template, manajemen tema manajemen *employee*, manajamen kostumer, *create, read, update, delete form, quiz, survey* dan *payment*.
5. Hasil akhir berupa situs web

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah penerapan MERN pada pengembangan aplikasi web *startup* Ubaform yang dapat meingkatkan performa, memudahkan pengembang *(developer)* dan mempercepat proses pengembangan sehingga tahap pertama dapat segera dirilis dan tidak menutup kemungkinan untuk pengembangan lebih lanjut pada masa yang akan datang.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

a. Bagi penulis

Manfaat yang didapatkan dari pekerjaan perintisan bisnis ini bagi penulis adalah dapat mengaplikasikan ilmu yang dipelajari baik yang didapat pada saat perkuliahan maupun secara mandiri atau otodidak dan menjadi tugas akhir penulis untuk menyelesaikan studi sarjana.

b. Bagi pengguna *(user)*

Pekerjaan perintisan bisnis ini dapat memberikan gambaran pembuatan formulir, kuis atau survei di kemudian hari (jika terealisasi) dengan beberapa fitur berbeda yang dapat meningkatkan kualitas tampilan terhadap formulir, kuis atau survei yang dibuat.

c. Bagi pemilik *startup*

Manfaat yang didapatkan oleh pengelola atau perintis *startup* Ubaform dengan adanya perintisan bisnis ini adalah mendapatkan pengetahuan baru terkait kemungkinan ide bisnis lama yang sudah ada namun dikembangkan dengan meningkatkan fitur baru seperti kostumisasi tema dan lain sebagainya.

1.6 Metodologi secara umum

Pada penelitian ini, proses implementasi MERN pada aplikasi *startup* Ubaform dirancang menggunakan metodologi yang secara umum melewati 7 (tujuh) tahapan, antara lain:

1. Analisis dan penelitian *(Research and Analysis)*
2. Rencana dan strategi *(Planning and Strategy)*
3. Desain (*Designing and Wireframing*)
4. Pembuatan konten *(Content Creation)*
5. Koding dan pengembangan *(Code and Development)*
6. Pengujian *(Testing)*
7. Peluncuran dan pemeliharaan *(Deployment and Maintenance)*

1.7 Susunan Laporan

Dalam rangka untuk mempermudah pemahaman pembaca pada laporan implementasi MERN pada aplikasi *startup* Ubaform ini, maka materi-materi pembahasan ditulis dengan susunan laporan sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini laporan berisi pembahasan mengenai gambaran umum dari keseluruhan tugas akhir serta permasalahan seperti apa yang diangkat sebagai acuan dasar. Pada bab ini terdiri dari latar belakang masalah, rumusan masalah, batasan masalah, manfaat penelitian, metodologi secara umum dan susunan laporan.

BAB II STUDI PUSTAKA

Pada bab ini laporan berisi pembahasan mengenai teori-teori pendukung dan informasi yang relevan sebagai landasan dalam implementasi MERN pada aplikasi *startup* Ubaform.

BAB III METODOLOGI

Pada bab ini laporan berisi pembahasan mengenai metodologi yang digunakan dalam implementasi MERN pada aplikasi *startup* Ubaform yang terdiri dari 5 (lima) tahapan, yaitu: Analisis *(Analysis)*, Perancangan *(Planning)*, Implementasi *(Implementation)*, Pengujian *(Testing)*, dan Peluncuran *(Deployment).*

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini laporan berisi pembahasan mengenai deskripsi atau penjelasan dari hasil pekerjaan yang dilakukan selama perintisan bisnis.

BAB V KESIMPULAN

Pada bab ini akan dikemukakan kesimpulan dari keseluruhan proses mulai dari awal sampai akhir yang telah dilakukan pada implementasi MERN pada aplikasi *startup* Ubaform. Selain itu akan dikemukakan juga saran sebagai tindak lanjut yang diperlukan guna pengembangan dan perbaikan lebih lanjut.

BAB II
LANDASAN TEORI

Untuk memperkuat argumentasi pembahasan laporan implementasi MERN pada aplikasi *startup* Ubaform, bab ini akan memuat landasan teori terhadap beberapa hal diantaranya:

1. *Startup* Ubaform
2. Situs Web *(Website)*
3. Mongo DB
4. Express JS
5. React JS
6. Node JS

2.1 *Startup* Ubaform

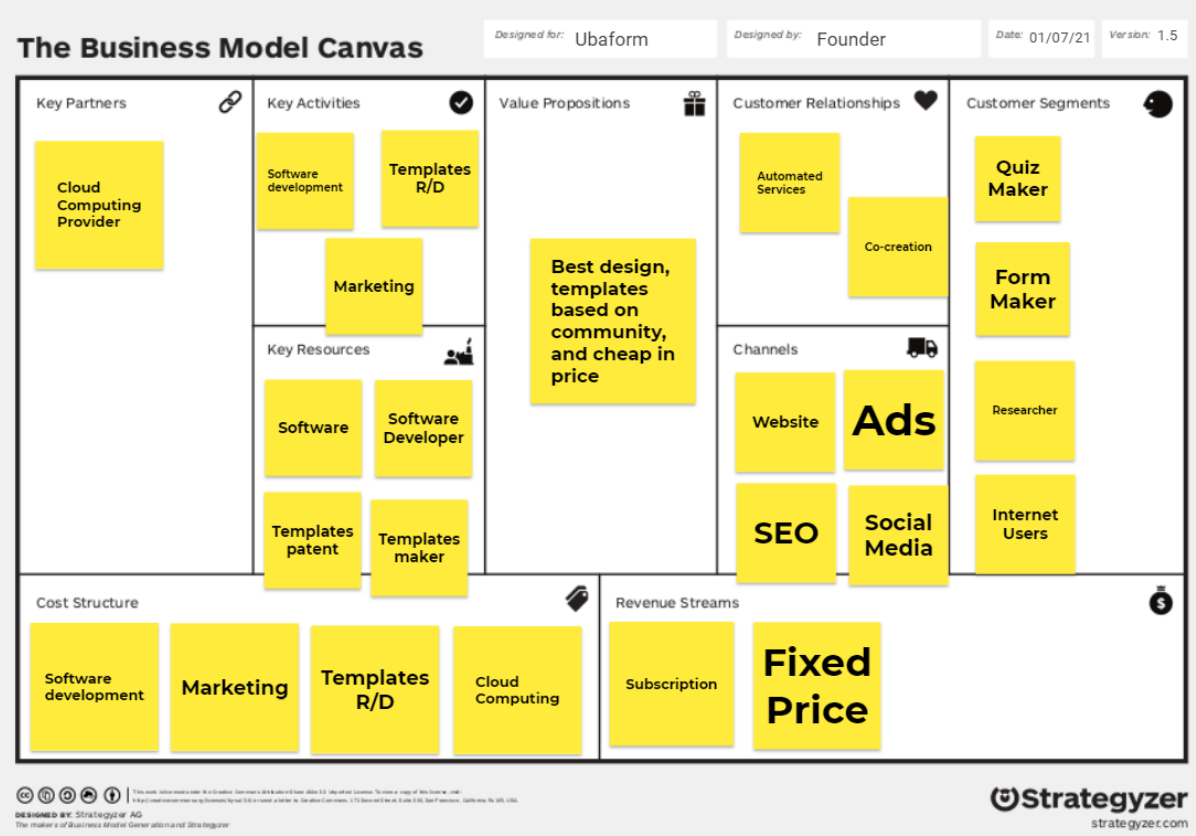
Ubaform merupakan *startup* yang menyediakan layanan pengelolaan dan pembuatan formulir, kuis, dan survei secara *online*. *Startup* Ubaform mengutamakan keunggulan *user interfaces* yang modern, interaktif dan mudah serta akses layanan yang cepat. Ubaform juga sekaligus merupakan nama tim pada aktivitas perintisan bisnis selama semester 7 (tujuh). Tim Ubaform beranggotakan 3 (tiga) orang mahasiswa Informatika UII angkatan 2017, yaitu Saya sendiri (Nasution - 17523199), Muhammad Fathi Asshidiqi (17523233), Zikri Dwi Andika (17523226).

2.1.1 Ide Bisnis *Startup* Ubaform

Ada banyak ide bisnis yang telah kami utarakan dan diskusikan bersama. Beberapa contoh ide bisnis tersebut diantaranya seperti pembuatan undangan *online*, sistem penyedia desain grafis, sistem penyedia kebutuhan pertanian dan masih banyak lagi. Namun setelah mempertimbangkan beberapa hal ide bisnis Ubaform menjadi pilihan akhir. Langkah pertama dalam pengembangan *startup* Ubaform yang kami lakukan adalah membuat logo sebagai bentuk penguatan citra dan konsep bisnis Ubaform, seperti pada gambar 2.1.

Gambar 2.1 Logo Ubaform

Setelah pembuatan logo, selanjutnya dilakukan pembuatan model bisnis untuk mengetahui alur pengembangan bisnis pada *startup* Ubaform, seperti pada gambar 2.2.



Gambar 2.2 *Model Business Canvas* (BMC) Ubaform

Berdasarkan hasil diskusi bersama anggota tim maka terbentuklah *Model Business Canvas* (BMC) pada gambar 2.2 yang mana mendeskripsikan model bisnis yang diterapkan pada *startup* Ubaform. Adapun beberapa komponen yang dideskripsikan pada BMC tersebut diantaranya *Customer Segments* yang berarti siapa pengguna atau kostumer Ubaform, *Cost Structure* yang berarti komponen apa saja yang membutuhkan biaya, *Channel* yang berarti media atau sarana apa saja yang bisa digunakan untuk menyampaikan produk atau jasa, *Key Resources* yang berarti sumber daya yang dimiliki perusahaan yang dibutuhkan untuk mewujudkan *value propositions*, *Value Propositions* yang berarti keunggulan atau nilai produk yang mendatangkan manfaat untuk ditawarkan kepada *customer segments*, *Key Activities* yang

berarti segala aktivitas yang berhubungan dengan produktivitas bisnis yang berkaitan dengan sebuah produk.

2.1.2 Sistem dan Fitur Ubaform

Untuk perancangan sistem pada web Ubaform dengan layanan pembuatan formulir, kuis dan survei dilakukan dengan penuh pertimbangan terhadap teknologi yang ada saat ini. *Startup* Ubaform dilengkapi fitur kostumisasi tema yang mana membutuhkan data yang dinamis yang terstruktur, selain itu dalam pembuatan formulir, kuis dan survei data yang disimpan disesuaikan berdasarkan inputan yang dilakukan oleh pengguna seperti jumlah dan jenis pertanyaan serta penyesuaian terhadap bentuk jawaban yang diinginkan. Selain itu sistem yang dibangun membutuhkan performa yang cepat baik itu disisi *frontend* maupun *backend* serta kemudahan dalam pengembangan.

Oleh karena itu pemilihan teknologi untuk *backend* dirancang menggunakan *framework* Express JS dan untuk *frontend* menggunakan *library* React JS untuk membuat web berbasis SPA untuk meningkatkan performa serta penyimpanan data menggunakan database Mongo DB sehingga skema penyimpanan data lebih terstruktur dan fleksibel, selain itu untuk kemudahan dan mempercepat proses pengembangan digunakan NPM *(Node Package Manager).*

2.2 Situs Web *(Website)*

Menurut kamus Oxford, *Website* merupakan *"a set of pages on the internet, where a company or an organization, or an individual person, puts information"* (Oxford University Press, 2021) jika diartikan maka kurang lebih merupakan satu set halaman di internet, di mana perusahaan atau organisasi, atau individu, menempatkan informasi. Lalu menurut Kamus Besar Bahasa Indonesia (KBBI) situs web merupakan program komputer yang menjalankan peladen (server) yang menyediakan akses kepada beberapa laman (KBBI, 2021). Situs web biasanya memerlukan server dan domain untuk beroperasi sehingga dapat diakses dimana saja dan kapan saja.

Masih merujuk pada kamus Oxford, Server merupakan "*a computer program that controls or supplies information to several computers connected in a network; the main computer on which this program is run*" (Oxford University Press, 2021) jika diartikan maka kurang lebih merupakan program komputer yang mengontrol atau memasok informasi ke beberapa

komputer yang terhubung dalam jaringan; komputer utama tempat program ini dijalankan. Jika dianalogikan server seperti sumber mata air yang menjadi tempat dimana setiap air dapat diambil. Server berisi berbagai data dan informasi situs web yang akan ditampilkan atau diakses oleh pengguna seperti *source code,* gambar, audio, video, text dan lain sebagainya.

Masih merujuk pada KBBI Domain adalah wilayah, daeran atau ranah (KBBI, 2021). Dalam hal ini dapat diartikan bahwa Domain merupakan sebuah nama atau alamat untuk mengidentifikasi sebuah jaringan tanpa menggunakan internet protocol (IP). Pada dasarnya sebuah situs web memiliki alamat IP berupa angka-angka tertentu yang digunakan untuk mengakses situs web tersebut. Berdasarkan hal inilah diperlukan domain untuk menggantikan alamat IP yang berupa angka tadi kedalam sebuah huruf atau kata dikarenakan kebanyakan manusia lebih mudah mengingat huruf atau kata dari pada kumpulan angka. Domain biasanya ditulis dengan diawali titik baru jenis domainnya seperti .com, .id dan lain sebagainya.

2.3 Mongo DB

Dalam pengembangan website salah satu hal penting adalah pemilihan *database* yang digunakan sebagai tempat penyimpanan data. Pemilihan *database* tidak hanya sebatas memiliki fungsi sebagai penyimpan data namun haruslah mendukung keunggulan serta performa dari produk yang dikembangkan sehingga layanan yang disediakan dapat memuaskan pengguna. Selain itu pemilihan *database* dapat mempengaruhi beberapa aspek lain seperti perkembangan website di masa yang akan datang, sumber daya manusia, finansial dan waktu pengembangan. Berdasarkan aspek tersebut pada pengembangan *startup* Ubaform diputuskan menggunakan *database* Mongo DB.

Mongo DB merupakan *an open source, nonrelational database management system (DBMS) that uses flexible documents instead of tables and rows to process and store various forms of data* (IBM Cloud Education, 2020) jika diartikan maka kurang lebih merupakan sistem manajemen basis data nonrelasional (DBMS) *open source* (tidak memungut biaya) yang menggunakan dokumen fleksibel, bukan tabel dan baris untuk memproses dan menyimpan berbagai bentuk data.

Beberapa poin penting yang menjadi pertimbangan dalam pemilihan Mongo DB secara lengkap dijelaskan sebagai berikut:

2.3.1 Mongo adalah *Database* NoSQL

Database NoSQL *(non*-SQL atau *not only* SQL*)* merupakan *database* yang menyimpan data dalam format yang berbeda dari tabel relasional atau *Relational Database Management System* (RDBMS) (Schaefer, 2021). *Database* NoSQL dikembangkan pada akhir 2000-an yang berfokus pada skalabilitas, kecepatan query, kompatibel pada perubahan aplikasi yang sering, dan membuat pemrograman lebih sederhana bagi *developer* (Schaefer, 2021). Meski RDBMS dalam sejarahnya sangat populer serta sudah membuktikan dirinya sebagai penopang yang kokoh untuk penyimpanan data selama beberapa dekade. Namun dengan pesatnya pertumbuhan jumlah penggunaan data digital dalam beberapa tahun terakhir (Kemp, 2021), maka hal ini juga berbanding lurus dengan meningkatnya tuntutan akan jenis data yang disimpan serta frekuensi operasi akan data yang dibutuhkan oleh perangkat lunak *database*. Model *database* relasional (SQL) terlalu kaku akan 2 (dua) hal yaitu:

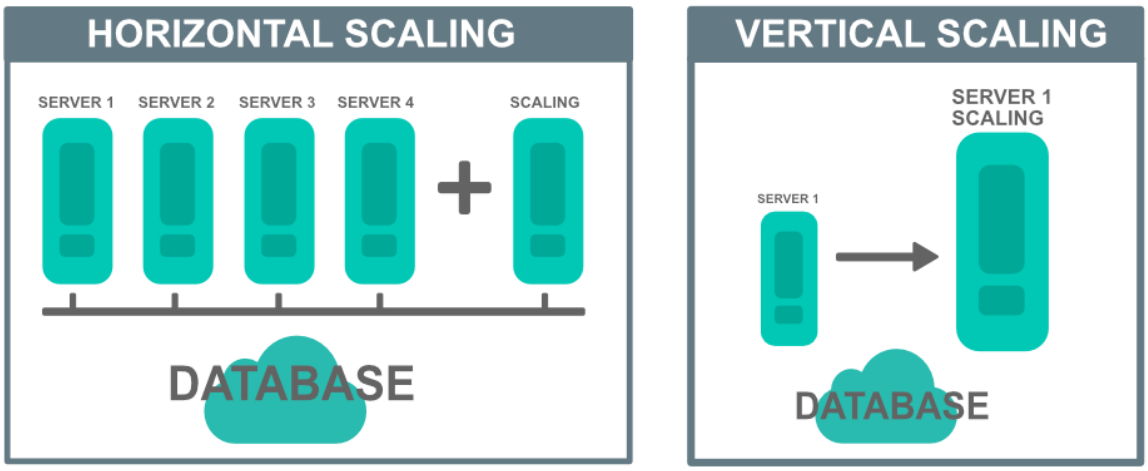
1. Skema penyimpanan data

   Dalam SQL skema data direpresentasikan sebagai baris dalam tabel seperti pada Microsoft Excel dimana setiap atribut unik dari skema diwakili oleh kolom dan tabel. Dalam hal ini atribut terbatas pada kumpulan tipe data dasar SQL yang ada seperti integer, string, dan tanggal (Paul DuBois, 2021), yang mana akan menyebabkan masalah ketika data yang memerlukan penyimpanan tidak teratur atau dinamis. Contoh data yang tidak tercangkup oleh tipe data yang ada pada SQL adalah tipe data array, item bersarang *(nested items)*, dan objek kostum *(custom objects).*

2. Peningkatan skala *database (Database Scaling)*

   Peningkatan skala *database* pada umumnya dapat dikategorikan menjadi 2 (dua) jenis yaitu *vertical scaling* dan *horizontal scaling. Vertical Scaling* berarti hanya meningkatkan kekuatan server *database*. Contohnya dengan melakukan upgrade komponen CPU atau memory. *Horizontal Scaling* berarti peningkatan yang dilakukan dengan menambahkan lebih banyak server pada *database*. Dengan ini *database* bisa didistribusikan terhadap server-server yang sudah ditambahkan sehingga masih bekerja dengan satu *database* tetapi ada beberapa server yang menjadi hostnya. *Database* SQL umumnya hanya mendukung skala jenis *Vertical Scaling* yang mana jenis ini memiliki kekurangan seperti ketidaknyamanan karena membutuhkan konfigurasi ulang dan waktu offline atau tidak beroperasi *(downtime).* Selain itu bermigrasi ketika memori telah di-*upgrade* sangat beresiko dan lebih mahal. Meskipun *database* SQL mengetahui konsep membagi data

menjadi pecahan horizontal *(sharding)* (Microsoft, 2017), akan tapi terdapat batasan tertentu dan biasanya sulit diterapkan (Mullins, 2018). Untuk memudahkan pemahaman berikut gambar 2.3 *Database Scaling Type*:



Gambar 2.3 *Database Scaling Type*

Salah satu solusi yang muncul untuk mengatasi kurangnya fleksibilitas ini adalah *database* NoSQL. *Database* model NoSQL menawarkan lebih banyak fluiditas dengan memungkinkan *developer* untuk menyimpan sejumlah besar data tidak terstruktur sehingga memberi banyak fleksibilitas dan dengan *Horizontal Scaling*. *Horizontal Scaling* yang mengacu pada kebalikan dari *Vertical Scaling* dimana sumber daya sistem terdiri dari beberapa unit perangkat keras terpisah yang terhubung ke satu *cluster* logis *(logical cluster)* yang bisa ditingkatkan *(upgrade)* dengan menambahkan server tambahan dan dihubungkan lagi dengan *cluster* logis. Hal ini tidak akan mengganggu operasi sistem pada server lain karena masing-masing server di *cluster* logis menjalankan salinan perangkat lunak (Mullins, 2018).

Seiring waktu, ada empat (4) jenis *database* NoSQL (Schaefer, 2021). Secara lengkap dijelaskan sebagai berikut:

1. *Document database*s

   Jenis ini menyimpan data dalam bentuk yang mirip dengan objek JSON *(JavaScript Object Notation)* dimana setiap dokumen berisi label dan nilai yang berpasangan. Nilai dari label biasanya dapat memiliki tipe data yang bervariasi seperti strings, numbers, booleans, arrays, dan objek. Karena variasi tipe data pada nilai label dan bahasa *query* yang kuat, *document databases* sangat bagus untuk berbagai penggunaan dan dapat digunakan

sebagai database untuk tujuan yang lebih umum. *Document database*s dapat dilakukan *vertical scaling* untuk mengakomodasi volume data yang besar yang berarti kedua tipe penskalaan dapat dilakukan.

2. *Key-Value databases*

   Merupakan jenis yang lebih sederhana dimana setiap *item* berisi kunci *(key)* dan nilai *(value).* Nilai pada umumnya hanya bisa diambil dengan mereferensikan kuncinya.

3. *Wide-column sotres*

   Jenis ini menyimpan data dalam bentuk tabel, kolom, dan baris yang dinamis. *Wide-column stores* lebih fleksibel dibandingkan database relasional karena setiap baris tidak diharuskan memiliki kolom yang sama.

4. *Graph databases*

   Jenis ini menyimpan data dalam bentuk *nodes* dan *edges. Node* biasanya akan menyimpan informasi tentang orang, tempat, dan lain sebagainya sedangkan *edge* menyimpan informasi tentang hubungan antara *nodes. Graph databases* cocok untuk kasus penggunaan mencari pola tertentu seperti pola pada media sosial, deteksi penipuan, dan lain-lain.

Dalam pengembangan Ubaform dari empat (4) jenis *database* NoSQL paling cocok menggunakan jenis *document databases.* Salah satu *document databases* yang paling populer saat ini adalah Mongo DB. Keputusan penggunaan *document databases* berdasarkan pada fleksibilitas dalam hal variasi skema *database* dan penskalaan *(scaling). Database* NoSQL cenderung memiliki kinerja *(performance)* yang lebih cepat untuk operasi baca tulis dengan tidak memerlukan gabungan untuk menghubungkan data dan dengan hal tersebut akan menghilangkan *overhead* dalam operasi SQL. Meskipun begitu dalam hal ini *database* NoSQL memang memliki peningkatan dibandingkan *databases* SQL akan tetapi membutuhkan pengorbanan pada fitur integritas data tertentu yang ditawarkan oleh SQL. Pengorbanan ini terjadi pada jaminan integrasi data yang hanya menawarkan BASE *(Basically, Available, Soft state, Eventually consistent)* sedangkan untuk *database* NoSQL selalu sesuai dengan jaminan ACID *(Atomicity, Consistency, Isolation, Durability).* Dengan pengorbanan ini merupakan salah satu kelemahan yang paling sering dikutip karena tidak mendukung ACID di banyak dokumen seperti *database* SQL. Namun untuk mengatasi hal tersebut Mongo DB menambahkan dukungan untuk transaksi ACID multi-dokumen pada Mongo DB versi 4.0.

Selain itu model data dalam *database* NoSQL biasanya dioptimalkan untuk *query* dan bukan untuk mengurangi duplikasi data dan *database* NoSQL bisa lebih besar kapasitasnya dan murah dari pada *database* SQL. Penyimpanan saat ini sangat murah sehingga sebagian besar menganggap ini sebagai kelemahan kecil dan beberapa *database* NoSQL juga mendukung kompresi untuk menghemat penyimpanan (Schaefer Lauren, 2021).

Untuk lebih menguatkan pemilihan Mongo DB sebagai *database* yang digunakan dalam pengembangan *startup* Ubaform, berikut empat (4) keunggulan *database* NoSQL menurut (Schaefer Lauren, 2021):

1. Model data yang fleksibel *(Flexible data models)*

   Database NoSQL biasanya memiliki skema yang sangat fleksibel. Skema fleksibel memudahkan untuk membuat perubahan pada database saat kondisi berubah.

2. Penskalaan Horizontal *(Horizontal scaling)*

   Sebagian besar *database* SQL mengharuskan penskalaan secara vertical *(vertical scaling)* yang mana membutuhkan biaya yang lebih mahal ketika kapasitas sudah melebihi batas maksimum. Akan tetapi *database* NoSQL memungkinkan penskalaan horizontal. Ini berarti dapat menambahkan server yang lebih murah kapan pun dibutuhkan.

3. Kecepatan Kueri *(Fast queries)*

   Kueri pada *database* NoSQL bisa lebih cepat daripada *database* SQL. Seringkali, data dari database SQL dinormalisasi, sehingga satu objek atau kueri entitas harus menggabungkan data dari beberapa tabel. Semakin besar tabel, semakin besar gabungannya. Namun, data dalam database NoSQL sering disimpan dengan cara yang dioptimalkan untuk kueri. Aturan praktis saat menggunakan MongoDB adalah *"Data is that is accessed together should be stored together"* atau kurang lebih maknanya adalah data yang diakses bersama harus disimpan bersama. Kueri biasanya sangat cepat karena tidak memerlukan penggabungan.

4. Mudah bagi pengembang *(Easy for developers)*

   Beberapa database NoSQL seperti MongoDB memetakan struktur data mereka ke bahasa pemrograman populer. Pemetaan ini memungkinkan pengembang untuk menyimpan data mereka dengan cara yang sama seperti mereka menggunakannya dalam kode aplikasi mereka. Ini mungkin tampak seperti keuntungan kecil, tetapi pemetaan ini memungkinkan pengembang untuk menulis lebih sedikit kode, menghemat waktu pengembangan, dan memiliki lebih sedikit *bug*.

Selain itu terdapat beberapa perbedaan antara database NoSQL dengan database SQL (Schaefer Lauren, 2021). Berikut beberapa perbedaan yang ditampilkan pada tabel 2.1:

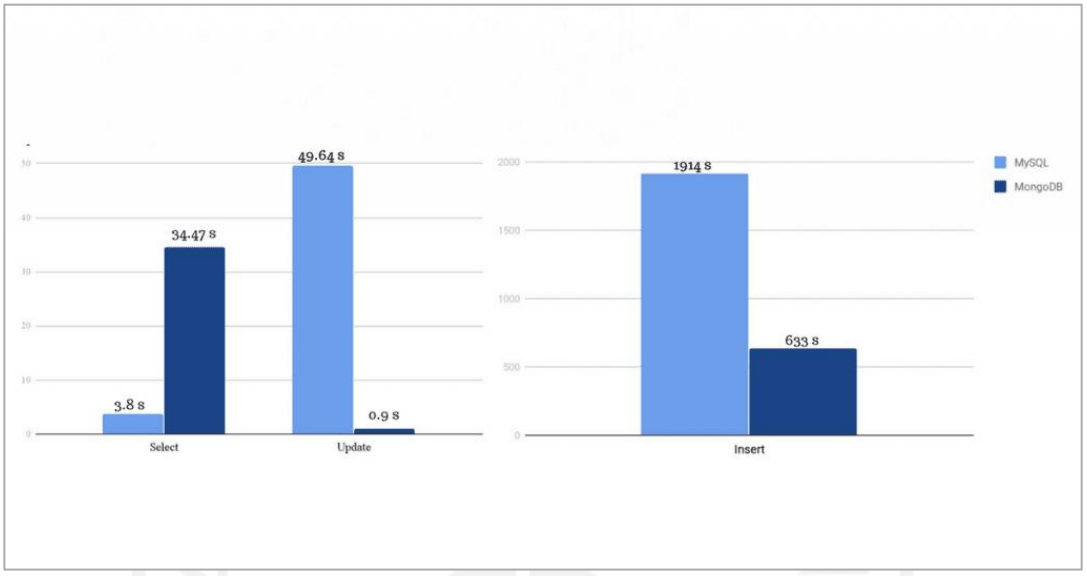
Tabel 2.1 Perbandigan database NoSQL vs database SQL

Variabel Perbandingan	SQL Database	NoSQL Database
Model Penyimpanan Data	Tabel dengan baris dan kolom tetap	*Document*: Dokumen JSON, *Key-value*: pasangan nilai kunci, *Wide-column*: tabel dengan baris dan kolom dinamis, *Graph*: simpul dan tepi
Sejarah Perkembangan	Dikembangkan pada 1970-an dengan fokus pada pengurangan duplikasi data	Dikembangkan pada akhir 2000-an dengan fokus pada penskalaan dan memungkinkan perubahan aplikasi yang cepat yang didorong oleh praktik gesit dan DevOps.
Contoh	Oracle, MySQL, Microsoft SQL Server, and PostgreSQL	*Document*: MongoDB and CouchDB, *Key-value*: Redis and DynamoDB, *Wide-column*: Cassandra and HBase, *Graph*: Neo4j and Amazon Neptune
Tujuan utama	Untuk tujuan umum	*Document*: untuk tujuan umum, *Key-value*: data dalam jumlah besar dengan kueri pencarian sederhana, *Wide-column*: data dalam jumlah besar dengan pola kueri yang dapat diprediksi, *Graph*: menganalisis dan melintasi hubungan antara data yang terhubung
Skema *(Scheme)*	Kaku *(kaku)*	Flexibel *(Flexible)*
Penskalaan *(Scaling)*	Vertikal (peningkatan skala dengan server yang lebih besar)	Horizontal (penskalaan di seluruh server komoditas)
Transaksi ACID Multi-Rekam *(Multi-Record ACID Transactions)*	Didukung	Sebagian besar tidak mendukung transaksi ACID multi-rekam. Namun, beberapa seperti MongoDB mendukung ACID.
Penggabungan *(join)*	Biasanya diperlukan	Biasanya tidak diperlukan

Pemetaan Data ke Objek	Membutuhkan ORM (pemetaan objek-relasional)	Banyak yang tidak memerlukan ORM. Dokumen MongoDB dipetakan langsung ke struktur data dalam sebagian besar bahasa pemrograman populer.

2.3.2 Kinerja Mongo DB *(*Mongo DB *Performance)*

Salah satu keunggulan Mongo DB dibandingkan dengan database MySQL adalah performa yang lebih cepat terutama dalam menangani data dalam jumlah besar yang terstruktur. Dalam pengembangan web Ubaform penggunaan data membutuhkan strukturisasi yang tepat karena setiap data yang disimpan bersifat fleksibel sesuai dengan apa yang pengguna masukan. Hal ini mengacu pada fleksibilitas pembuatan formulir, kuis dan survei dengan beberapa bentuk data yang digunakan dan tetap harus terstruktur. Oleh karena itu pemilihan Mongo DB dibandingkan MySQL untuk digunakan dalam pengembangan web Ubaform sudah sangat tepat karena dua poin tersebut sudah menjadi spesifikasi dari Mongo DB. Namun demikian untuk lebih menguatkan perlu adanya pertimbangan performa diantara keduanya, perbandingan ini dilakukan pada Mongo DB versi 3.2.0 dengan MySQL 5.7.9. Masing-masing telah diuji pada instansi Amazon m4.xlarge terpisah dengan ubuntu 14.4 x64 dan konfigurasi *default*, pengujian ini menghasilkan 1000000 *record* (Shah, 2017). Untuk perbandingan dari hasil pengujian dapat dilihat pada gambar 2.3 berikut



Gambar 2.4 Kecepatan Mongo DB (Waktu paling sedikit lebih baik)

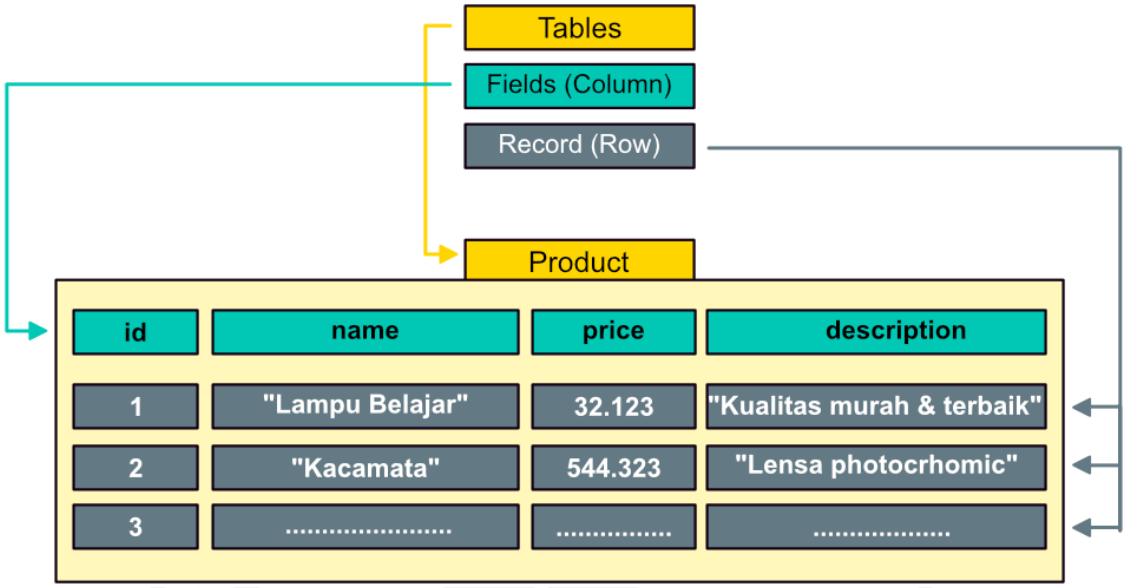
Pada gambar 2.4 dapat dilihat bahwasanya performa Mongo DB dibandingkan MySQL lebih baik. Hal ini menjadi poin penting dalam pengembangan Ubaform yang membutuhkan performa yang cepat.

2.3.3 Cara Kerja Mongo DB (NoSQL) vs MySQL (SQL)

Perbedaan selanjutnya antara Mongo DB dengan MySQL adalah dari aspek cara kerja kedua database tersebut. Terdapat dua (2) variabel untuk perbandingan ini yaitu skema dan relasi (Schwarzmüller, 2018). Skema bermakna bentuk pemodelan pada database seperti penetapan tipe data atau bagaimana data disimpan sedangkan relasi bermakna hubungan antara tabel atau antara dokumen yang satu dengan yang lain. Berikut penjelasan kedua perbedaan cara kerja antara database:

1. Penulisan Skema

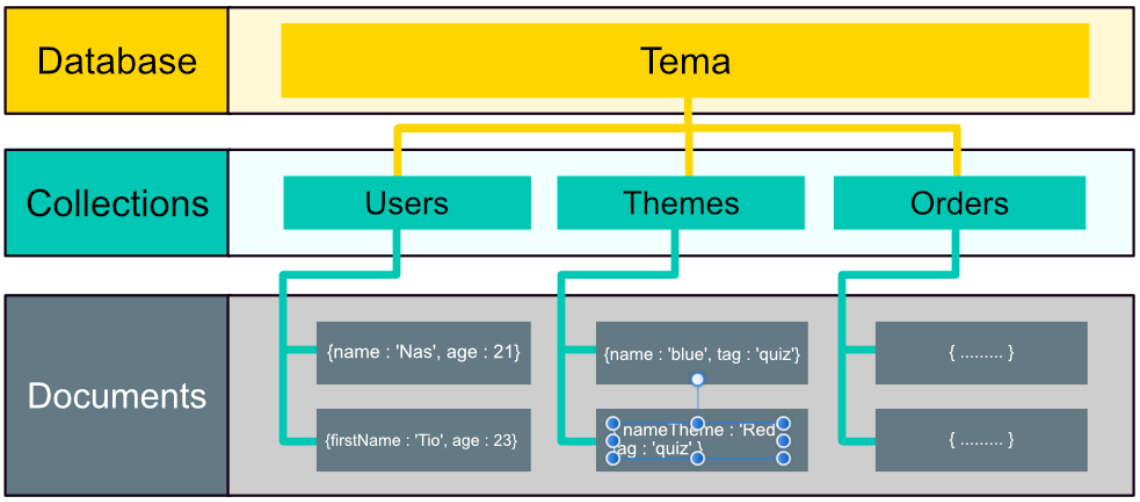
   Pada database MySQL (SQL) data disimpan sebagai satu atau kumpulan catatan *(record)* dalam tabel dan setiap tabel memiliki struktur yang jelas dimana *fields* berbentuk kolom *(columns)* atau dapat dianalogikan dalam sebuah tabel ada yang dinamakan judul kolom atau nama kolom. Fields yang menentukan data mana yang boleh masuk ke tabel atau *record* dan data mana yang tidak boleh masuk. Struktur ini didefinisikan *fields* beserta tipe data nya sehingga *field* yang tipe data nya integer menerima record dengan tipe integer dan nama *field* nya juga harus sama. Kira-kira skema penulisan struktur pada MySQL seperti pada gambar 2.5 berikut



Gambar 2.5 Contoh Skema Penulisan Struktur MySQL

Pada skema ini jika ingin menambah *record* harus sesuai skema yang sudah didefinisikan, selain itu jika ingin menambah *field* tambahan misalnya *field* "discount" harus memodifikasi skema atau membuat *tables* tersendiri yang berbeda.

Sedangkan pada database NoSQL tidak memiliki skema akan tetapi penulisan atau penyusunan data dilakukan dalam koleksi *(collections)*. Pada NoSQL *collections* sama dengan *tables* pada SQL sedangakan *documents* pada NoSQL sama dengan *records* pada SQL. Tentu saja perbedaan bukan hanya pada nama atau penyebutan dari kedua database akan tetapi pada NoSQL data dapat dimasukan dari struktur yang berbeda ke dalam koleksi yang sama sedangkan pada SQL tidak dapat melakukan hal ini dikarenakan setiap tabel memiliki skema yang tetap. Kira-kira penulisan atau penyusunan koleksi pada NoSQL seperti pada gambar 2.8 berikut

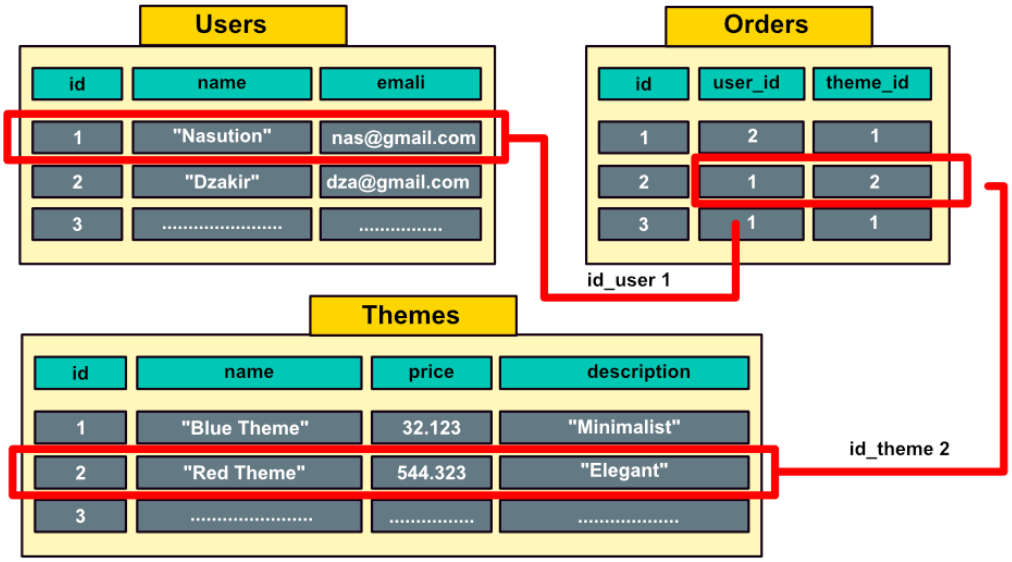


Gambar 2.6 Contoh Penyusunan koleksi pada NoSQL

Untuk mengetahui jelas perbedaan kedua *database* dapat diperhatikan pada kedua gambar di bagian *records* untuk SQL dan *documents* untuk NoSQL. pada gambar 2.5 *records* hanya dapat ditambah jika sesuai dengan definisi *field* yang tipe datanya sudah ditentukan. Sedangkan pada gambar 2.6 *documents* bisa ditambahkan tanpa terikat tipe data pada *collections.*

2. Relasi

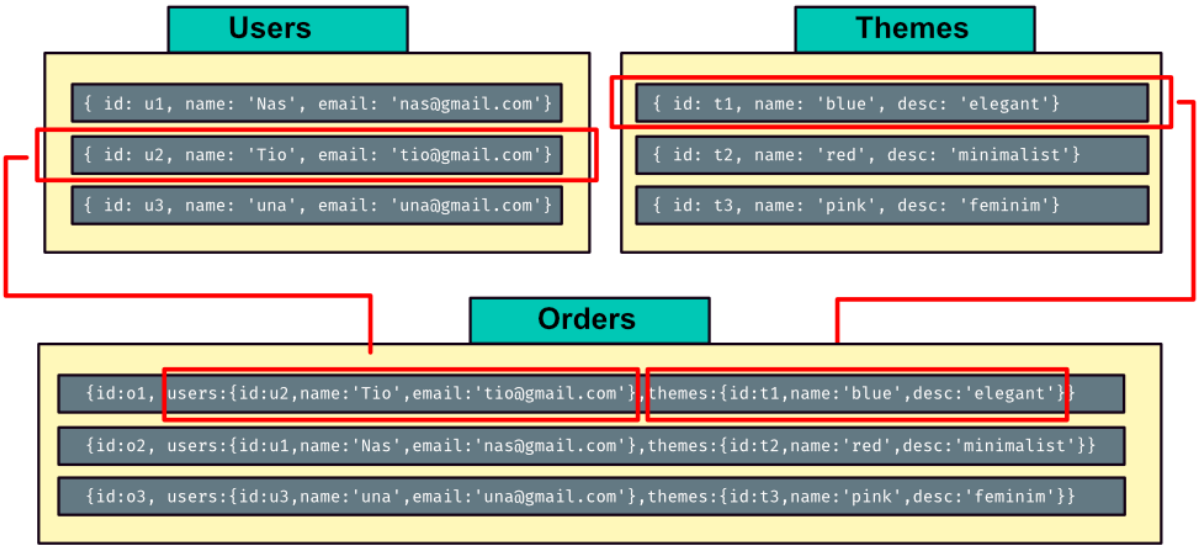
Pada database MySQL (SQL) tentu saja terdapat relasi antara tabel yang satu dengan yang lainnya. Relasi pada MySQL merupakan bagian penting karena pada SQL banyak tabel yang dibuat agar terhindar dari duplikasi data, sehingga perlu untuk menghubungkan data dari tabel yang berbeda. Untuk menggambarkan pemahaman terhadap relasi antara tabel pada kasus ini dibuat beberapa simulasi relasi tabel. Ada tiga (3) tabel antara lain tabel Users, Themes, dan Orders. Berikut relasi antara tabel dijelaskan dalam gambar 2.7



Gambar 2.7 Relasi antar table SQL

Dapat dilihat pada gambar 2.7 tabel Orders memiliki hubungan atau relasi dengan tabel Users dan Themes. Data pada setiap tabel yang di *records* tidak memiliki kesamaan atau tidak ada duplikasi diantara tabel. Relasi pada tabel dilakukan dengan menulis id dari tabel yang direlasikan ke tabel baru dalam kasus ini tabel Orders. Sehingga pada tabel Orders berisi id dari tabel lain dengan data dari id tersebut, id pada tabel Orders disebut sebagai id tamu atau bias dikenal dengan *foreign_key*. Dengan relasi ini memberikan keuntungan tidak ada data yang terduplikasi karena data dikelola dalam satu tabel masing-masing. Namun dilain sisi jika penggabungan data antara tabel banyak akan membuat proses lambat.

Sedangkan untuk database NoSQL yang tidak memiliki relasi jika dalam kondisi yang sama maka aka ada duplikasi data. Artinya ketika NoSQL membutuhkan data dari kedua dokumen maka data tersebut akan di duplikasikan ke dokumen baru. Untuk menggambarkan pemahaman pada kasus ini dibuat simulasi yang sama pada gambar 2.7, namun pada simulasi ini bukan tabel tetapi dokumen, ada tiga (3) dokumen yaitu Users, Themes, dan Orders. Untuk lebih lengkapnya dapat dilihat pada gambar 2.8



Gambar 2.8 Konsep penggabungan antar dokumen NoSQL

Dapat dilihat pada gambar 2.8 dimana dokumen Orders menampung data dari dua dokumen lain sehingga digabung menjadi satu pada dokumen Orders. Konsep duplikasi ini memang memiliki bahaya seperti akan menimbulkan masalah pada pembaruan data yang mana jika dilakukan pembaruan pada salah satu dokumen tanpa menyesuaikan data pada dokumen lain. Jadi disini peran *developer* untuk memastikan data terkait juga dilakukan pembaruan. Namun terlepas dari masalah tersebut dengan konsep seperti ini memliki keuntungan yang lebih besar karena tidak ada nya relasi yang menyebabkan proses lambat.

2.4 Express JS

Express merupakan *a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications* (StrongLoop, IBM, and other expressjs.com contributors., 2021) jika diartikan maka kurang lebih merupakan kerangka kerja aplikasi web Node.js yang minimal dan fleksibel yang menyediakan serangkaian fitur tangguh untuk aplikasi web dan seluler. Ada beberapa hal yang menjadi pertimbangan pemilihan express dalam pengembangan *startup* Ubaform, berikut beberapa hal kenapa menggunakan Express:

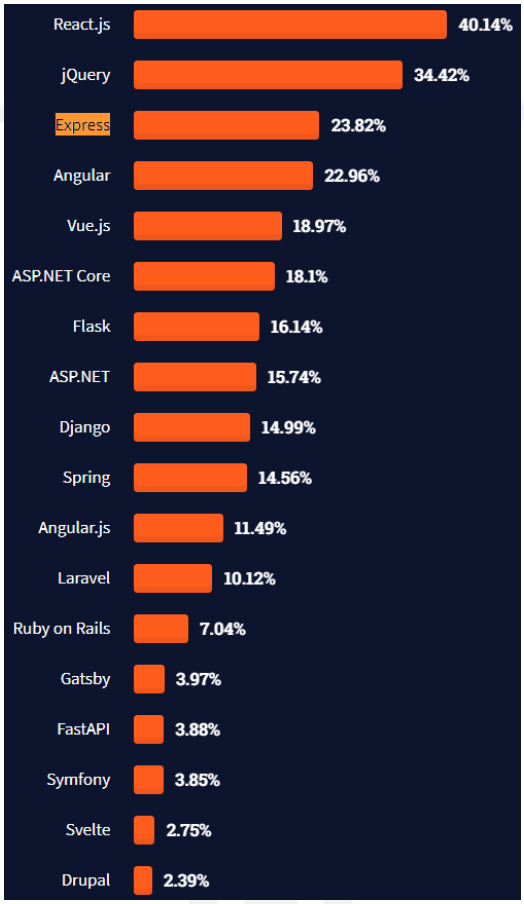
2.4.1 Express merupakan *framework* populer

Express JS sangat *framework* yang populer saat ini, banyak mekanisme yang tersedia ketika menggunakannya. Express JS menyediakan empat (4) mekanisme (Developer Mozila, 2021), berikut mekanisme yang ada pada Express JS:

1. Penulisan penanganan *(handler)* untuk melakukan permintaan dengan kata kerja HTTP yang berbeda di jalur URL atau rute *(routes)* yang berbeda
2. Terintegrasi dengan mesin rendering *'view'* atau terintegrasi dengan bagian *frontend* sehingga dapat menghasilkan *response* dengan memasukan data ke dalam tampilan atau *user interfaces template*.
3. Mengatur pengaturan web aplikasi secara umum seperti port yang digunakan untuk menghubungkan server dengan lokasi dalam tampilan atau *user interfaces template* yang digunakan untuk merender *response*
4. Menambahkan pemrosesan permintaan tambahan "*middleware*" pada setiap titik dalam rute *(route)* permintaan (*request)*

Express JS merupakan *framework* yang cukup minimalis, *developer* telah membuat *package* Middleware yang kompatibel untuk mengatasi hampir semua masalah dalam proses pengembangan web. Selain itu ada beberapa *library* untuk membantu *developer* bekerja pada *cookie*, *session, user login,* URL *parameters*, POST data, *security header* dan lain sebagainya. Selain dari beberapa hal tersebut alasan penggunaan Express pada pengembangan web Ubaform adalah popularitas dari Express itu sendiri. Popularitas sebuah *framework* sangat penting karena dapat dijadikan indikator apakah *framework* tersebut akan terus ada dan dipertahankan atau sebaliknya kemudian semakin populer sebuah *framework* juga akan memiliki sumber dokumentasi yang baik, *library* tambahan dan dukungan teknis sehingga akan memudahkan dalam pengembangan dan mencari sumber daya manusia yang kompatibel. Express awalnya dirilis pada November 2010 dan saat ini sudah ada sampai versi 4.17.1 API (dengan 5.0 dalam "alfa") (Developer Mozila, 2021). Survei yang dilakukan oleh Stackoverflow pada tahun 2021 dengan pertanyaan *"Which web frameworks and libraries have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the framework and want to continue to do so, please check both boxes in that row)"* yang jika diartikan kurang lebih "*Framework* web dan *library* mana yang telah Anda kerjakan pengembangan ekstensifnya selama setahun terakhir,

dan yang mana yang ingin Anda kerjakan selama tahun depan? (Jika Anda berdua bekerja dengan *framework* dan ingin terus melakukannya, centang kedua kotak di baris itu)". Data hasil survei dapat dilihat pada gambar 2.9 berikut



Gambar 2.9 Hasil Survei Stackoverflow 2021

Dapat dilihat bahwa Express menempati peringkat ketiga *framework* yang paling umum digunakan dengan hasil 23.82%. Selain itu jumlah perusahaan besar yang menggunakan Express (StrongLoop, IBM, and other expressjs.com contributors, 2021), jumlah orang yang berkontribusi pada basis kode, dan jumlah orang yang memberi dukungan secara gratis dan berbayar, dengan hal-hal ini dapat disimpulkan bahwa express merupakan *framework* yang populer (Developer Mozila, 2021).

2.4.2 Express merupakan *framework unopinionated*

Seperti yang kita ketahui *framework* sering menyebut diri mereka sebagai *"opinionated"* atau *"unopinionated"*. Kerangka kerja beropini *(opinionated framework)* merupakan *framework* yang memliki opini (ketentuan) tentang "cara yang tepat" untuk menangani tugas

tertentu. *Framework* jenis ini sering membantu *developer* pada domain tertentu atau dalam memecahkan masalah dengan jenis tertentu karena memiliki “cara yang tepat” untuk melakukan sesuatu. Hal ini biasanya cepat dipahami karena sudah memiliki dokumentasi yang sudah ada tentang bagaimana cara yang tepat menyelesaikan masalah yang ada. Namun dengan hal tersebut *framework opinionated* kurang fleksibel dalam memecahkan masalah diluar domain utama mereka dan cenderung menawarkan lebih sedikit pilihan untuk komponen serta pendekatan yang dapat digunakan dalam *framework* tersebut (Developer Mozila, 2021).

Sedangkan kerangka kerja yang tidak beropini *(unopinionated framework)* merupakan *framework* yang tidak memiliki opini (ketentuan) dan memiliki batasan yang jauh lebih sedikit tentang cara terbaik untuk menggunakan komponen bersama-sama untuk mencapai tujuan atau bahkan komponen apa yang digunakan. *Framework* ini memudahkan *developer* untuk menggunakan *tools* yang cocok untuk menyelesaikan tugas tertentu meskipun dengan *effort* tersendiri untuk menemukan komponen yang dibutuhkan tersebut (Developer Mozila, 2021).

Untuk Express termasuk *unopinionated framework* sehingga lebih fleksibel dan memasukan beberapa komponen seperti Middleware untuk menangani permintaan tertentu. Selain itu dapat menyusun aplikasi dalam satu *file* atau beberapa *file* dan menggunakan struktur direktori apapun (Developer Mozila, 2021).

2.5 React JS

React adalah *open-source library* JavaScript deklaratif, efisien dan fleksibel untuk membangun antarmuka pengguna (HANRY HAM., 2019). React memungkinkan untuk membuat user interface yang kompleks dengan set kode kecil yang terisolasi yang disebut "komponen". React JS ini digunakan untuk menangani lapisan tampilan dalam aplikasi satu halaman atau SPA dan pengembangan *mobile application*. React JS dikelola oleh facebook, instagram, komunitas pengembang dan korporasi. React berusaha untuk memberikan kecepatan, kesederhanaan, dan skalabilitas.

2.5.1 Keunggulan React JS

Terdapat tiga (3) fitur yang diunggulkan oleh React JS (Facebook, 2021). antara lain:

1. Deklaratif

React JS bersifat deklaratif artinya React dapat membuat *User Interfaces* yang interaktif sehingga dapat dengan mudah membuat desain yang sederhana untuk setiap *state* di dalam aplikasi.

2. Berbasis komponen

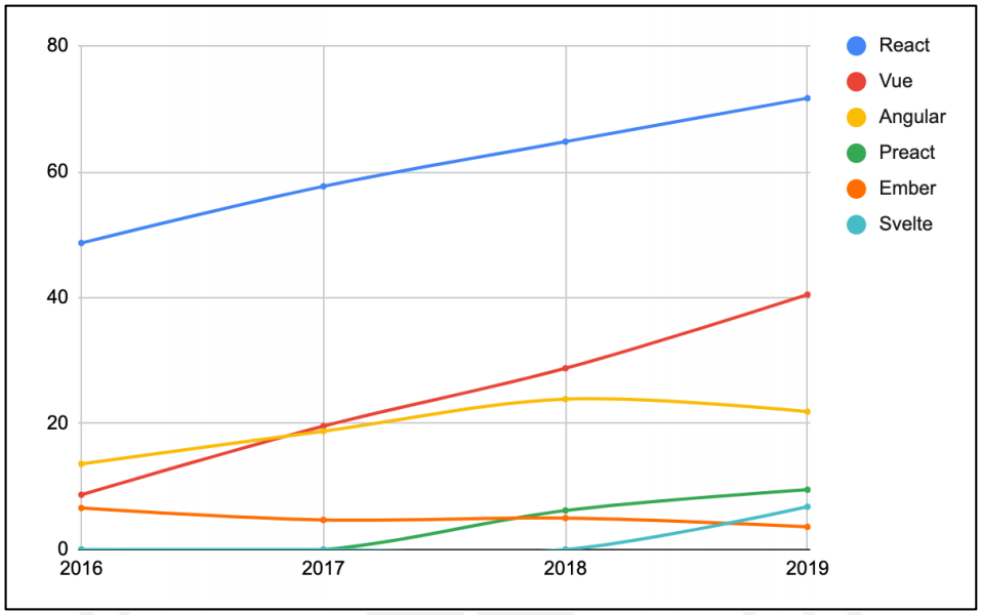
   React JS dapat memebuat komponen-komponen berskala kecil atau besar secara terpisah atau komponen terenkapsulasi *(Encapsulated Component)* sehingga dapat memudahkan pengembang dalam melakukan *debug* atau menemukan *error*, memodifikasi serta memperbaiki kode tanpa mengganggu komponen lain yang tidak saling berhubungan. Komponen-komponen kecil ini dapat digabungkan menjadi suatu komponen yang lebih komplek sehingga mampu menjadi satu kesatuan yang utuh sesuai fungsi dan kebutuhan.

3. *Learn Once, Write Anywhere*

   Sebagai pengembang aplikasi tentunya mempelajari bahasa pemrograman atau *library* pendukung menghabiskan banyak waktu, tenaga serta uang. Namun ketika belajar React JS pengembang bisa menulis kode dimana saja, artinya React JS dapat digunakan untuk membuat sebuah platform berbasis web dan juga mobile.

2.5.2 Perbandingan React JS dengan *framework* atau *library* Javascript lain

Selain React JS Terdapat banyak *framework* atau *library* javascript diantaranya ada Angular JS dan Vue JS. Namun dalam pengembangan *startup* Ubaform dipilih React JS dengan beberapa pertimbangan. Pertimbangan tersebut termasuk dari tiga (3) fitur yang diunggulkan oleh React JS, selain itu ada beberapa pertimbangan lain diantaranya berdasarkan survei *The State of javaScript*, yang diterbitkan setiap tahun sejak 2016 Edisi 2019 memiliki 21.717 responden pengembang (Sacha Greif, 2019). Untuk memilih kerangka kerja untuk evaluasi, variabel dari survei ini dipertimbangkan yaitu berapa banyak pengembang yang aktif menggunakan kerangka kerja yang dimaksud. Data kualitatif penggunaan tahunan, selama rentang waktu 2016 hingga 2019, untuk enam teratas framework pada tahun 2019 digambarkan pada Gambar 2.10 berikut



Gambar 2.10 Persentase responden yang setuju dengan pernyataan "Saya pernah menggunakannya dan akan menggunakannya lagi", *State of JavaScript Surveys* 2016 – 2019

Dapat dilihat pada gambar 2.10 React menjadi *library* atau *framework* yang populer dan banyak digunakan dibandingkan *library* atau *framework* lainnya. Kemudian selain banyak nya pengguna dan populer digunakan juga terdapat komparasi atau perbandingan DOM *benchmarks* untuk beberapa *framework* atau *library* javascript. Tes DOM *benchmarks* ini berdasarkan tesis yang dilakukan oleh Mattias Levlin di Åbo Akademi *University* (DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte, 2020). Pada tesis tersebut terdapat beberapa parameter yang dijadikan sebagai variabel perbandingan diantaranya adalah CRUD *(Create, Read, Update, Delete)* dan *compile*. Ringkasan perbandingan dapat dilihat pada gambar 2.11 berikut

TECHNICAL METRICS	React v16.12.0	Vue v2.6.11	Angular v8.2.14	Svelte v3.20.0
Add 10,000 (ms)	30.96	25.36	52.75	31.26
Edit one (ms)	16.58	22.23	6.08	0.11
Edit 10,000 (ms)	17.86	20.64	896.76	885.03
Remove one (ms)	16.54	24.51	0.09	0.53
Remove 10,000 (ms)	7.39	33.33	23.83	22.97
Compilation (s)	3.96	3.07	8.70	1.61
Minified size (kb)	6.4	63.5	187.6	3.5
POPULARITY	React v16.12.0	Vue v2.6.11	Angular v8.2.14	Svelte v3.20.0
Documentation in number of languages	16	8	4	2
Positive interest in StateOfJs 2019 (%)	83.7	74.7	31.6	51.7
Number of Github stars, January 2020	142,850	159,091	29,756	56,789
OTHER METRICS	React v16.12.0	Vue v2.6.11	Angular v8.2.14	Svelte v3.20.0
Virtual DOM	Yes	Yes	No	No
TypeScript	No	No	Yes	Yes
Release date	2013	2014	2016	2016
OVERALL RATING	React v16.12.0	Vue v2.6.11	Angular v8.2.14	Svelte v3.20.0
Placement	1	2	4	3

Gambar 2.11 Performa terbaik ditandai dengan warna biru. Paling lambat yang ditandai dengan warna merah

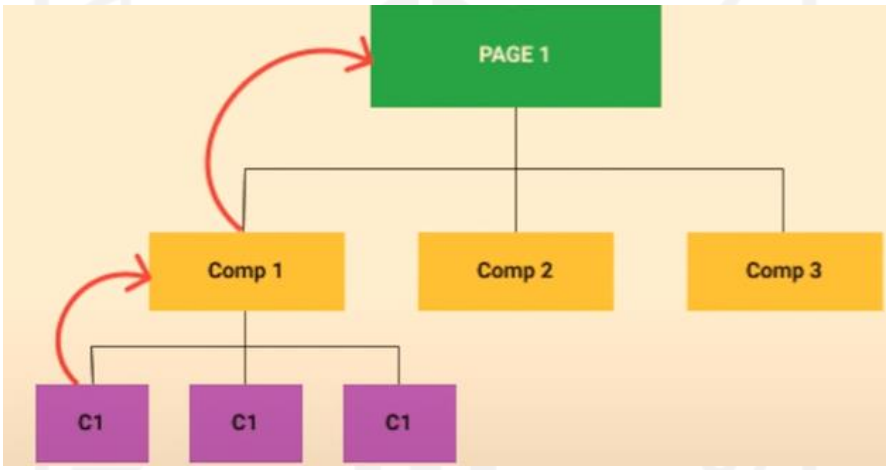
Dapat dilihat pada gambar 2.11 bahwa React lebih stabil, populer dan lebih baik performanya dibandingkan dengan *library* atau *framework* lain.

2.5.3 Pengikatan Data satu arah *(One-way data binding)*

Pengikatan data satu arah *(One-way data binding)* merupakan salah satu fitur react dimana *flow* data searah dari komponen tingkat yang lebih tinggi *(parent)* ke tingkat yang lebih rendah *(child)* (Facebook, 2021). Terdapat dua (2) hal yang tidak dapat dipisahkan dalam komponen react, yaitu data dan UI *(user interfaces)*, dimana data biasanya direpresentasikan dalam format javascript berupa objek. Dua komponen pada react tersebut memiliki cara kerja yang sederhana dimana ketika data berubah maka UI otomatis juga berubah, namun ketika UI berubah maka data tidak berubah secara otomatis kecuali ketika ada sebuah pemicu untuk mengubah *state*

data bisa berubah, inilah yang disebut pengikatan data satu arah *(One-way data binding).* Sebelum react lahir ada konsep yang sudah lebih dulu populer yaitu pengikatan data dua arah *(two-way data binding)* yang mana ketika data berubah maka UI juga otomatis berubah dan begitupun sebaliknya ketika UI berubah maka data juga berubah. Konsep *two-way data binding* dipopulerkan oleh *framework* seperti angular dan ember.

Selain itu dibandingkan pengikatan data dua arah, pengikatan data satu arah memiliki keunggulan dalam hal performa, skalabilitas dan *(single source of truth)* yang dapat memudahkan pelacakan pemicu perubahan pada data. Ilustrasi pengikatan data satu arah dapat dilihat pada gambar 2.12 berikut



Gambar 2.12 *One-way data binding*

Pada gambar 2.12 terdapat tiga (3) tingkatan level komponen dimana dari tinggi ke rendah terdapat komponen dengan nama PAGE 1 sebagai *parent,* komponen dengan nama Comp 1 sebagai *child* pertama dan komponen dengan nama C1 sebagai *child* kedua. Ketiga komponen tersebut mengirim data dari *child* ke *parent* atau sebaliknya tanpa mengubah UI diantaranya.

Berdasarkan beberapa keunggulan yang ditawarkan React dan survei tentang perbandingan beberapa *library* atau *framework* dari berbagai sumber maka disimpulkan dan ditetapkan pada pengembangan *startup* Ubaform menggunakan React JS sebagai teknologinya.

2.6 Node JS

Node JS merupakan *runtime* JavaScript berbasis peristiwa asinkron, Node.js dirancang untuk membangun jaringan aplikasi yang skalabel (OpenJS Foundation, 2021). Node JS sama hal nya dengan JRE *(*Java *Runtime Environment),* perbedaan kedua nya Node JS untuk bahasa

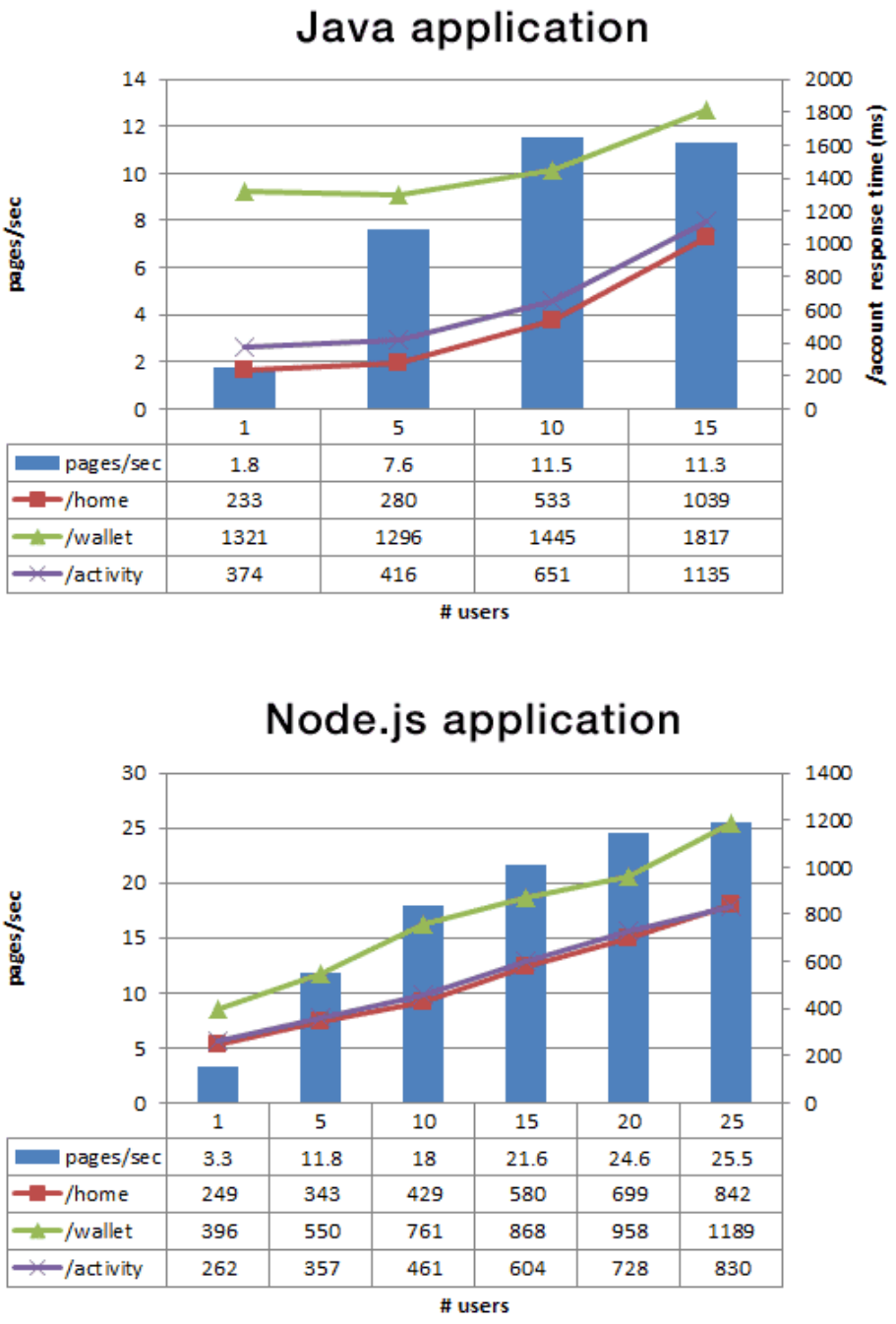
javascript sedangkan JRE untuk bahasa Java. Node JS mengkompilasi dan mengeksekusi kode javascript untuk dijalankan di sisi server. Hal ini juga yang menjadikan javascript sebagai bahasa yang bisa digunakan selain pada sisi *client* juga sebagai alternatif yang layak dijadikan pilihan dalam menangani sisi server. Node JS hadir dengan beberapa modul bawaan seperti NGINX yang terkait dengan persyaratan paling umum seperti manajemen file, HTTP *(Hypertext Transfer Protocol)*, SSL *(Secure Sockets Layer)*, DNS *(Domain Name System)*, kompresi dan kriptografi (Nodejs.org, 2021). Penggunaan Node JS juga didukung oleh Node Package Manager (NPM) yang merupakan kumpulan kode *open-source* yang dapat memungkinkan pengembang dalam menginstal *package* untuk digunakan dalam projek yang dikerjakan sehingga selain memudahkan juga mempercepat proses pengembangan. Berdasarkan kinerja yang kuat dalam hal kecepatan dan penggunaan sumber daya, Node JS telah digunakan oleh banyak perusahaan ternama seperti Netflix, LinkedIn, NASA, Paypal, Medium dan lain sebagainya. Untuk lebih detailnya berikut beberapa keunggulan dari Node JS diantaranya:

2.6.1 NPM (Node Package Manager)

NPM merupakan *package manager* untuk NodeJS yang dibuat pada tahun 2009 sebagai projek *open source* untuk membantu JavaScript *developer* dengan mudah berbagi modul kode. Kumpulan *open source* kode ini disebut juga NPM *registry*, selain untuk pengembangan web kumpulan modul kode ini juga bisa digunakan untuk pengembangan aplikasi seluler, robot, router dan juga kebutuhan komunitas *developer* JavaScript lainnya (npm, inc, 2021). NPM juga mudah diimplementasikan ketika ingin menginstal *package* cukup dengan melakukan CLI *(Command Line Interface)* `npm install <package-name>` atau bisa menggunakan alternatif lain seperti `yarn add <package-name>` (OpenJS Foundation, 2021). Saat ini NPM merupakan perangkat lunak terbesar di dunia dengan lebih dari 800.000 paket kode dan 3 juta unduhan perhari (Vorbach, 2021). Hal ini menjadikan NodeJS populer selain itu juga banyak perusahaan ternama menggunakan NodeJS seperti slack, Microsoft, Netflix, adobe dan lain sebagainya. Dengan adanya NPM proses pengembangan perangkat lunak bisa menjadi lebih mudah dan cepat karena *developer* tidak perlu membuat kode mulai dari nol melainkan hanya menginstal *package* yang dibutuhkan dan memodifikasi kode pada *package* tersebut jika diperlukan.

2.6.2 Performa Node JS

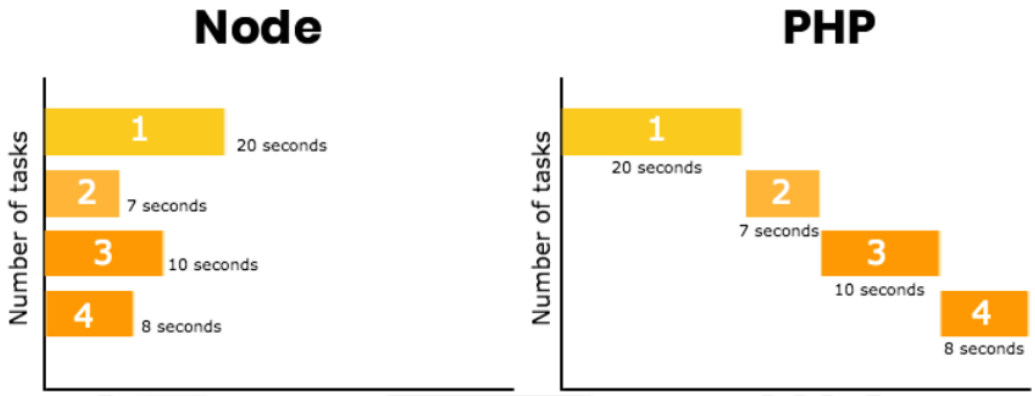
Pembahasan performa untuk Node JS tidak terlepas dari dilakukannya tes *benchmark*. Tes ini dilakukan oleh perusahaan ternama Paypal dimana mereka membandingkan antara Node JS dengan bahasa Java yang notabene sebagai bahasa pemrograman paling populer secara keseluruhan selama lebih dari 2 dekade sebelum akhirnya disalip oleh bahasa Phyton. Meskipun Node JS lebih baru dibandingkan dari beberapa alternatif sejenisnya Node JS tetap tidak hadir sebagai pengganti keseluruhan dari semua alternatif yang ada, melainkan sebuah solusi alternatif dengan performa yang lebih cepat. Tes *benchmark* yang dilakukan oleh Paypal untuk beralih dari Java ke Node JS untuk layanan mereka, untuk hasil tes dapat dilihat pada gambar 2.13 berikut (Kunwar, 2021).



Gambar 2.13 Perbandingan tes *benchmark* Node JS dengan Java

Dapat dilihat pada gambar 2.13 bahwasanya performa Node JS lebih baik dengan *response time* lebih rendah dari pada Java. Bahkan Paypal mengklaim ada penurunan 35% dalam waktu *respons* rata-rata untuk halaman yang sama. Ini menghasilkan halaman yang disajikan 200 ms lebih cepat serta dua kali lebih banyak *request* dari Java (Kunwar, 2021). Selain itu jika Node JS dibandingkan performanya dengan bahasa lain seperti PHP, Node JS lebih unggul karena

PHP berbasis *synchronous code* sedangkan Node JS berbasis *asynchronous code.* Operasi *synchronous* dapat dimisalkan seperti antrean pada sebuah tempat perbelanjaan dimana kostumer baru bisa membayar belanjaan ketika kostumer yang lebih dulu antre selesai. Sedangkan untuk operasi *asynchronous* kebalikannya dimana kostumer tidak perlu menunggu kostumer lain selesai akan tetapi bisa ditangani secara bersamaan. Permisalan ini seperti halnya ketika eksekusi kode terjadi dimana Node JS akan melakukan eksekusi secara bersamaan tanpa harus menunggu kode lain selesai dieksekusi sedangkan PHP harus menunggu kode pada baris pertama selesai baru bisa baris berikutnya dieksekusi (Patel, 2019). Untuk lebih memahami konsep *synchronous* dan *asynchronous* dapat dilihat pada gambar 2.14 berikut



Gambar 2.14 Eksekusi kode *asynchronous* vs *synchronous*

Dapat dilihat pada 2.14 Node JS melakukan eksekusi *task* secara bersamaan sedangkan PHP harus menunggu *task* pertama baru *task* kedua dieksekusi. Ini membuat Node JS lebih unggul dalam kecepatan mengkesekusi kode.

Implementasi MERN-*stack* pada pengembangan startup Ubaform tidak terlepas dari hasil landasan teori ini. Dimana alasan pemilihan teknologi berdasarkan faktor keunggulan teknologi itu sendiri baik dalam hal performa, kemudahan, popularitas dan keberlanjutan teknologi untuk beberapa tahun kedepan dalam hal ini kemampuan skalabilitas. Semua keunggulan ini sudah dijabarkan secara terperinci pada poin 2.3 sampai 2.6.

2.7 *Single Page Aplication* (SPA)

Single Page Application (SPA) merupakan aplikasi yang berjalan di browser dan tidak memerlukan pemuatan ulang halaman saat digunakan. Ini berarti bahwa pengguna tidak berpindah halaman saat user mengirim setiap permintaan ke server. Menerapkan *single page application* memiliki dua manfaat utama yaitu mengurangi penggunaan *bandwidth* jaringan dan mempercepat penjelajahan. SPA ini didukung secara luas oleh pustaka Javascript yang mengurangi bandwidth jaringan. Data yang diterima dari server dalam format JSON *(JavaScript Object Notation)* dan dapat dilihat secara asinkron menggunakan JavaScript, sehingga penjelajahan menjadi lebih cepat.

2.7.1 Performa SPA

Dalam hal performa SPA lebih baik dibandingkan *Multi page application* (MPA). Hal ini berdasarkan pada pemuatan halaman yang terjadi pada SPA dilakukan sekali dan setiap perubahan data yang terjadi hanya pada yang bersangkutan alias tidak memuat ulang keseluruhan halaman web. Sedangkan MPA setiap kali data mengalami perubahan maka keseluruhan halaman web akan dimuat ulang, hal ini akan membuat transaksi serta pengambilan data yang sering bahkan data yang tidak dibutuhkan setiap kali *request* terjadi ke server, oleh karena itu SPA lebih cepat dibandingkan MPA.

2.7.2 Konsep dan Pengembangan SPA

Dalam konsep SPA sangat dipisahkan antara sisi *frontend* dan *backend*, aplikasi yang berbasis SPA menggunakan API untuk melakukan interaksi seperti *request* dan *response* antara kedua sisi tersebut. Selain itu keunggulan pengembangan SPA setiap komponen yang dibuat dapat digunakan secara berulang-ulang seperti dalam keunggulan *library* React JS dinamakan *reusable-component*. Hal ini sangat membantu mempercepat serta mempermudah proses pengembangan. Oleh karena itu pemilihan React JS sebagai *library* untuk mengembangkan web Ubaform berbasis SPA sudah sesuai denga napa yang dibutuhkan oleh pengembang saat ini.

BAB III
ANALISIS, PERANCANGAN DAN PENGEMBANGAN

Pada bagian ini berisi penjelasan metodologi yang digunakan dalam proses implementasi MERN pada pengembangan aplikasi *startup* Ubaform. Menurut (Mistry, 2021) selaku CTO di Monocubed salah satu perusahaan dibidang layanan pembuatan website dalam artikelnya, ada tujuh (7) Fase Siklus hidup pengembangan web *(7 Phases of Web Development Life Cycle)* antara lain sebagai berikut:

1. Analisis dan penelitian *(Research and Analysis)*
2. Rencana dan strategi *(Planning and Strategy)*
3. Desain *(Designing and Wireframing)*
4. Pembuatan konten *(Content Creation)*
5. Koding dan pengembangan *(Code and Development)*
6. Pengujian *(Testing)*
7. Peluncuran dan pemeliharaan *(Deployment and Maintenance)*

3.1 Analisis dan penelitian *(Research and Analysis)*

Proses analisis dan penelitian perlu dilakukan untuk tujuan dapat mengetahui pemahaman yang jelas tentang apa yang dikembangkan bahkan menurut (Mistry, 2021) ada tiga (3) jenis pertanyaaan untuk ditanyakan pada tahap ini antara lain sebagai berikut:

1. Analisis Tujuan dari website *(purpose)*
2. Analisis Persyaratan *(requirement)*
3. Ekspetasi *(expectation)*

Ketiga jenis pertanyaan ini sudah kami lakukan teliti dan analisis jauh-jauh hari sebelum proses pengembangan dimulai bersama anggota tim lainya. Untuk itu kami melakukan wawancara dengan beberapa calon *stakeholder* secara *online* dan juga meneliti beberapa kompetitor untuk mengetahui proses bisnis, fitur, alur sistem dan lain sebagainya.

3.1.1 Analisis Tujuan dari website *(purpose)*

Untuk tujan dari website ada beberapa poin yang kami diskusikan sehingga menghasilkan hasil akhir seperti yang ada pada tabel 3.1 berikut:

Tabel 3.1 Tujuan *startup* Ubaform

Tujuan Website	Tipe Website	Target *Audiens* / *Stakeholder*
Membuat sebuah platform yang memudahkan orang untuk membuat formulir, kuis, DAN SURVEI dalam satu platform/aplikasi.	*Software as Service*	Peneliti, HRD, guru/dosen, mahasiswa atau pengguna internet pada umumnya.

Setelah menentukan tujuan dari pengembangan *startup* Ubaform selanjutnya adalah mengumpulkan informasi terkait sistem yang dikembangkan. Pengumpulan informasi dilakukan dengan metode wawancara secara daring kebeberapa *stakeholder*. Adapun beberapa gambaran pertanyaan yang diberikan saat wawancara kurang lebih seperti pada tabel 3.2 berikut:

Tabel 3.2 Pertanyaan-pertanyaan saat wawancara

No	Pertanyaan	*Stakeholders*
1.	Kesusahan terbesar apa yang anda rasakan ketika membuat kuis secara *online* atau *offline*?	Guru / Dosen
2.	Kesusahan terbesar apa yang anda rasakan ketika membuat formulir secara *online* atau *offline*?	HRD
3.	Kesusahan terbesar apa yang anda rasakan ketika membuat survei secara *online* atau *offline*?	Peneliti
4.	Seberapa sering anda merasakan kesusahan tersebut?	Peneliti, HRD, guru/dosen, mahasiswa atau pengguna internet pada umumnya.
5.	Apa yang anda coba lakukan untuk menyelesaikan kesusahan tersebut?	
6.	Sudahkah kesusahan tersebut terselesaikan Jika belum, mengapa?	
7.	Apakah sudah pernah coba melakukan komparasi cara lain dalam menyelesaikan kesusahan tersebut Jika sudah, apa yang terjadi sebelum dan setelah anda mencoba alternatif solusi itu?	
8.	Apakah Anda membayar untuk solusi itu, berapa?	

3.1.2 Analisis Persyaratan *(requirement)*

Tahap selanjutnya adalah menganalisis kebutuhan pengguna *(User requirement)* yang mana hal ini merupakan suatu hal yang penting untuk dilakukan, pasalnya dalam pengumpulan informasi sebelumnya juga bertujuan untuk memahami kebutuhan pengguna seperti apa dan bagaimana. Oleh karena itu pengembangan sistem tidak akan jauh dari kebutuhan yang diinginkan oleh pengguna.

Untuk menganalisis *user requirement* perlu memahami dua jenis *user requirement*, yaitu *User Requirement Functional* dan *User Requirement Non-Functional.* Terdapat perbedaan diantara keduanya, untuk *User Requirement Functional* merupakan kebutuhan terhadap fungsi sistem, sedangkan untuk *User Requirement Non-Functional* merupakan kebutuhan yang tidak berkaitan dengan fungsi sistem. Berikut hasil analisis *User Requirement* berdasarkan informasi yang dikumpulkan.

A. *User Requirement Functional*

Bagi Admin atau pemilik

- Dapat menambah dan mengelola *template* dan tema pada website Ubaform
- Dapat mengatur manajemen admin atau *employee*
- Dapat melihat informasi pengguna

Bagi *Customer*

- Dapat membuat formulir, kuis, atau survei secara *online*
- Dapat mengedit formulir, kuis, atau survei secara *online*
- Dapat menghapus formulir, kuis, atau survei secara *online*
- Dapat melihat, memilih dan menggunakan tema saat membuat formulir, quiz, atau survei secara *online*
- Dapat menggunakan *template* sesuai keinginan

B. *User Requirement Non-Functional*

Bagi Admin atau Pemilik

- Penyajian data yang mudah dipahami
- Tampilan menarik dan mudah dioperasikan
- Tampilan situs web *responsive (Desktop / Mobile)*

Bagi *Customer*

- Tampilan menarik dan mudah dioperasikan

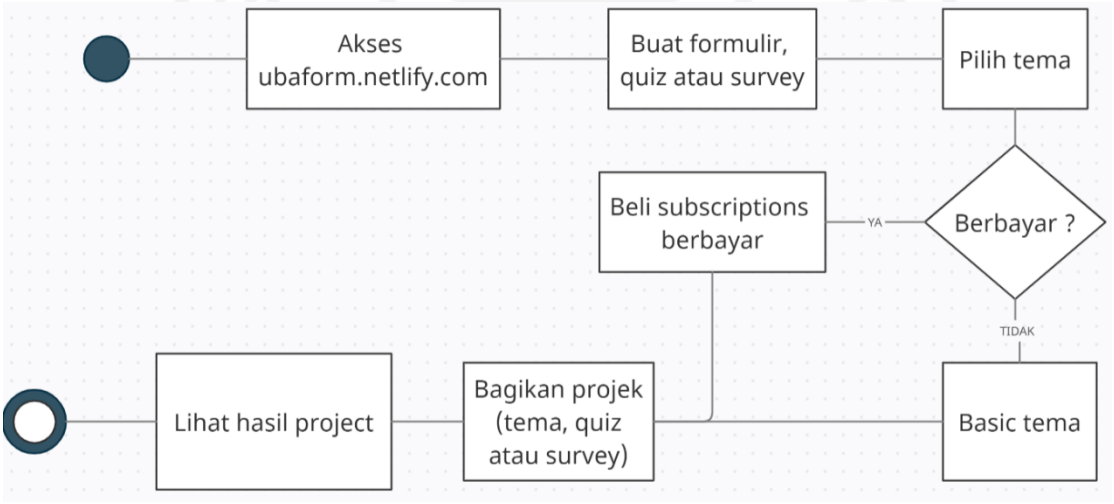
- Tampilan situs web *responsive (Desktop / Mobile)*
- Bisa diakses dimanapun dan kapanpun

3.1.3 Ekspetasi *(expectation)*

Proses selanjutnya adalah membuat diagram aktivitas dan *use case* untuk memberi gambaran alur situs web agar tujuan bisa benar-benar sesuai selain itu juga memberi gambaran atau ekspektasi bagaimana interaksi setiap aktor yang memiiki akses terhadap web *startup* Ubaform serta ekspektasi proses bisnis sistem yang dikembangkan.

Diagram Aktivitas *(Activity Diagram)*

Setelah menetukan tujuan dan melakukan proses wawancara, didapatkan kesimpulan sebuah proses bisnis sebagai berikut:

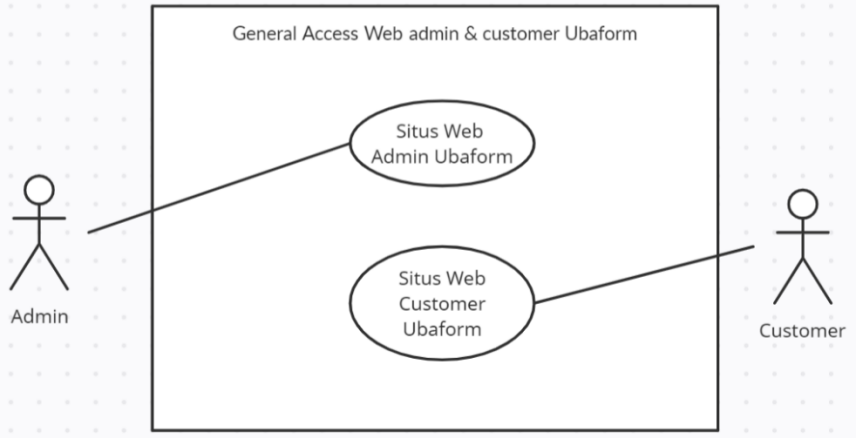


Gambar 3.1 Diagram aktivitas atau Proses bisnis *customer* Ubaform

***Use Case* Diagram**

Selanjutnya adalah pembuatan *use case diagram* berdasarkan analisis dan informasi yang telah dikumpulkan dari hasil wawancara sebelumnya. *Use case* diagram merupakan satu dari berbagai jenis diagram UML *(Unified Modelling Language)* yang menggambarkan hubungan interaksi antara sistem dan aktor (Dicoding, 2021). Dalam pengembangan starup Ubaform terdapat dua sistem berbeda yaitu untuk admin (web admin) dan untuk *customer* (web customer). Dari analisis dan infromasi tersebut didapatkan interaksi antara aktor yang memiliki

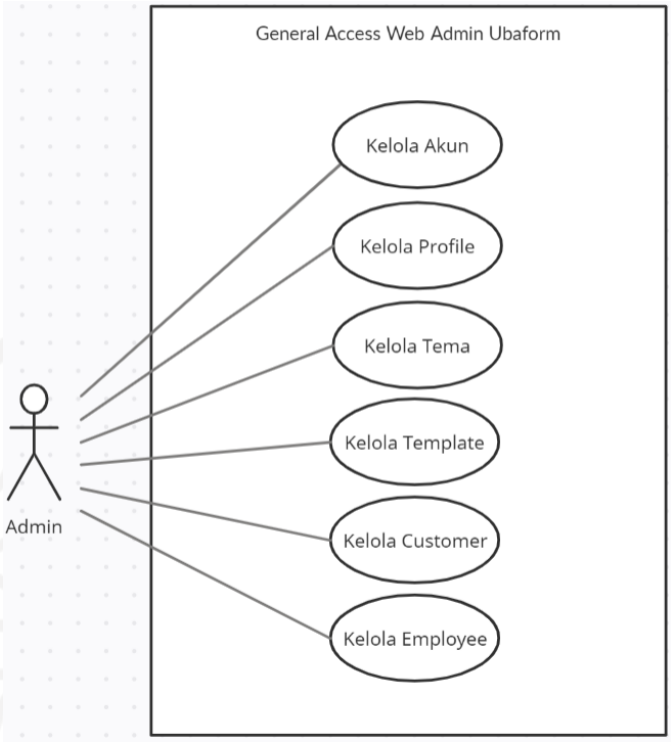
akses terhadap dua situs Ubaform tersebut. Secara umum akses aktor masing-masing pada dua sistem tersebut dapat dilihat di *use case* pada gambar 3.2 berikut:



Gambar 3.2 *Use case* secara umum untuk web admin dan *customer*

Gambar *use case* 3.2 merupakan gambaran umum terhadap aktor yang memiliki akses terhadap situs web admin dan *customer* Ubaform. Dari gambar 3.2 dapat dipahami bahwa, secara umum hanya ada satu aktor yang dapat mengakses web admin Ubaform. Sedangkan untuk gambaran umum aktor yang memiliki akses terhadap situs web *customer* Ubaform juga terdapat satu aktor. Untuk detail akses yang dapat dilakukan oleh aktor pada masing-masing situs web dapat dilihat sebagai berikut

A. Admin



Gambar 3.3 *Use case* pada Web Admin Ubaform

Use case pada gambar 3.3 merupakan gambaran akses yang dapat dilakukan oleh admin untuk mengelola web admin Ubaform. Akses tersebut secara detail dijelaskan sebagai berikut:

- Kelola Akun

  Merupakan gambaran akses yang dapat dilakukan oleh admin untuk mengelola akun. Dapat dilihat bahwa ada tiga hal yang dapat dilakukan oleh admin untuk mengelola akun, yaitu mendaftarkan akun, melakukan login apabila sudah diberi akses dan keluar dari sistem atau logout.
- Kelola *Profile*

  Merupakan gambaran akses yang dapat dilakukan oleh admin untuk mengelola *profile.* Dalam mengelola *profile* admin dapat melihat *profile* dan melakukan pengeditan pada *profile*
- Kelola Tema

  Merupakan gambaran akses yang dapat dilakukan oleh admin untuk mengelola tema untuk digunakan oleh *customer*. Tema yang dikeloa oleh admin nantinya akan digunakan oleh *customer* untuk pembuatan *forms, quiz* atau *survey*.
- Kelola *Tempalte*

Merupakan gambaran akses yang dapat dilakukan oleh admin untuk mengelola template. Template yang dikeloa oleh admin nantinya akan digunakan oleh *customer* untuk pembuatan *forms, quiz* atau *survey*

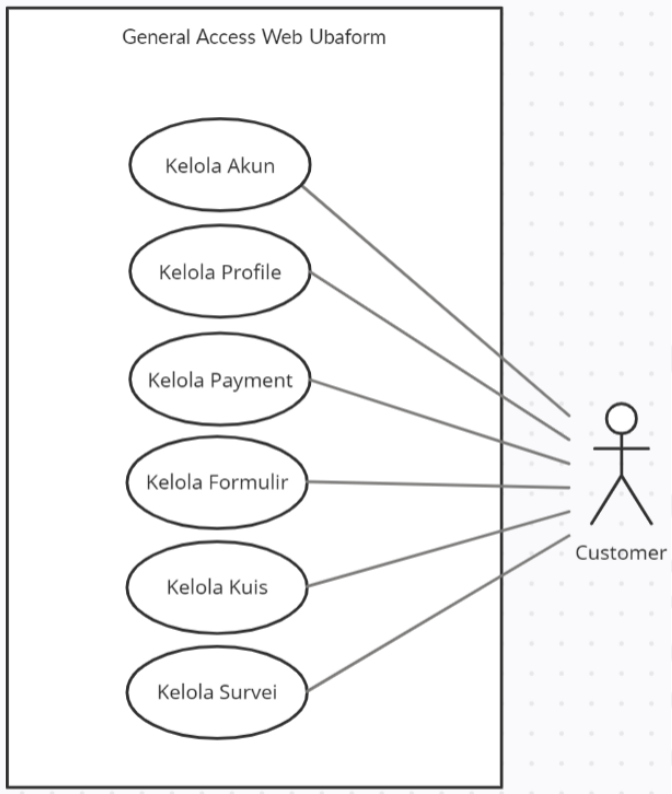
- Kelola *Customer*

  Merupakan gambaran akses yang dapat dilakukan oleh admin untuk mengelola akun *customer* yang telah terdaftar pada sistem Ubaform. Dapat dilihat bahwasanya admin dapat melihat *customer* terdaftar, melihat detail *customer* dan menghapus *customer*.

- Kelola *Employee*

  Merupakan gambaran akses yang dapat dilakukan oleh admin untuk mengelola admin terdaftar seperti memberi akses pada akun terdaftar yang belum memiliki akses pada sistem admin untuk memiliki akses serta admin dapat melihat daftar admin serta menghapus admin.

B. Customer



Gambar 3.4 *Use case* pada Web *Customer* Ubaform

Use case pada gambar 3.4 merupakan gambaran akses yang dapat dilakukan oleh admin untuk mengelola web admin Ubaform. Akses tersebut secara detail dijelaskan sebagai berikut:

- Kelola Akun

  Merupakan gambaran yang dapat dilakukan oleh *customer* untuk mengelola akun. Pengelolaan akun diantaranya customer dapat melakukan pendaftaran pada sistem, melakukan *login* serta *logout*
- Kelola *Profile*

  merupakan gambaran yang dapat dilakukan oleh *customer* untuk mengelola profile. Pengelolaan ini berupa informasi *customer* mulai dari melihat *profile* serta mengedit *profile*
- Kelola *Payment*

  Merupakan gambaran yang dapat dilakukan oleh *customer* untuk mengelola *subscription plan*. *Customer* dapat membeli *subscription* berbayar untuk dapat mengakses tema yang berbayar

- Kelola Formulir

  Merupakan gambaran yang dapat dilakukan oleh *customer* untuk mengelola projek formulir. *Customer* dapat membuat, mengedit, menghapus, melihat formulir.
- Kelola Kuis

  Merupakan gambaran yang dapat dilakukan oleh *customer* untuk mengelola projek formulir. *Customer* dapat membuat, mengedit, menghapus, melihat kuis.
- Kelola Survei

  Merupakan gambaran yang dapat dilakukan oleh *customer* untuk mengelola projek survei. *Customer* dapat membuat, mengedit, menghapus, melihat survei.
- Kelola Tema

  Merupakan gambaran yang dapat dilakukan oleh *customer* untuk mengelola tema. *Customer* dapat melihat daftar tema yang ada sesuai dengan *subscription plan*.
- Kelola *Template*

  Merupakan gambaran yang dapat dilakukan oleh *customer* untuk mengelola *template*. *Customer* dapat melihat daftar template yang ada kemudian menggunakan nya pada saat melakukan pembuatan projek formulir, kuis ataupun survei.

3.2 Rencana dan strategi *(Planning and Strategy)*

Langkah selanjutnya adalah proses perencanaan dan strategi, dimana pada langkah ini penyusunan strategi dilakukan pada semua aspek situs web seperti desain, teknologi, konten, dan pemasaran. Namun pada tahap ini pembahasan akan berfokus pada strategi perencanaan teknologi yang digunakan serta perencanan perancangan peta situs *(sitemap)* pada proses pengembangan *startup* Ubaform. Tahap ini tentunya sangat penting untuk mendapatkan hasil yang maksimal dan mengurangi kesalahan-kesalahan pada saat proses implementasi atau pengembangan nantinya. Perencanaan teknologi yang digunakan dalam pengembangan seperti pemilihan bahasa pemrograman, *library* atau *framework*, metodologi pengembangan, *text editor* dan pemilihan *database*.

Perancangan Peta situs harus mendeskripsikan hubungan antar halaman satu dengan halaman lainnya atau antar halaman induk dengan halaman-halaman kecil yang terdapat pada halaman induk. Pembuatan peta situs memiliki dua tujuan yaitu selain untuk membangun situs

web yang ramah pengguna juga untuk memudahkan navigasi antar halaman. Kemudian untuk proses perencanaan pada teknologi yang digunakan pada saat proses pengembangan baik yang membantu proses pengembangan maupun yang terlibat langsung dalam pengembangan, penjelasan dapat dilihat pada tabel 3.3 berikut

Tabel 3.3 Komponen Perencanaan

Komponen	Penjelasan	Teknologi *(tools)* yang dipilih
Metodologi pengembangan	Dalam pengembangan *startup* Ubaform dipilih metodologi pengembangan SCRUM. Dengan metodologi pengembangan ini proses pengembangan dilakukan dengan menetapkan estimasi waktu dalam pengembangan untuk setiap fiturnya. Setiap hari akan ada laporan dan diskusi bersama dengan *developer* lainnya terkait apa yang akan dikerjakan, sedang dikerjakan, belum dikerjakan serta kesulitan apa saja pada saat pengerjaan. Dengan dilakukannya hal-hal tersebut diharapkan pengembangan akan lebih terfokus dan cepat.	SCRUM *methodology*
Text Editor	Dengan penggunaan *text editor* Visual Studio Code dapat memudahkan pengembangan. Hal ini dikarenakan terdapat banyak *extension* yang ada dan fitur yang disediakan seperti dapat melakukan penulisan kode bersama dengan anggota *developer* lain secara *real time*, fitur komunikasi dan lain sebagainya.	Visual Studio Code
Bahasa Pemrograman	Pada pengembangan *startup* Ubaform bahasa pemrograman yang digunakan adalah JavaScript secara keseluruhan baik itu untuk sisi klien maupun sisi server. Dengan basis satu bahasa diharapkan pengembangan dapat dipermudah karena tidak perlu banyak mempelajari bahasa lain.	JavaScript
Framework / *Library*	Penggunaan React dan Express JS memberikan banyak dampak mulai dari kemudahan dalam implementasi serta mampu mempercepat proses pengembangan dengan bantuan intalasi kode eksternal untuk digunakan.	React JS dan Express JS
Database	Pemilihan Mongo DB sebagai database untuk digunakan dalam sistem Ubaform adalah karena kemampuan skalabilitasnya yang bagus serta berbasis SQL.	Mongo DB

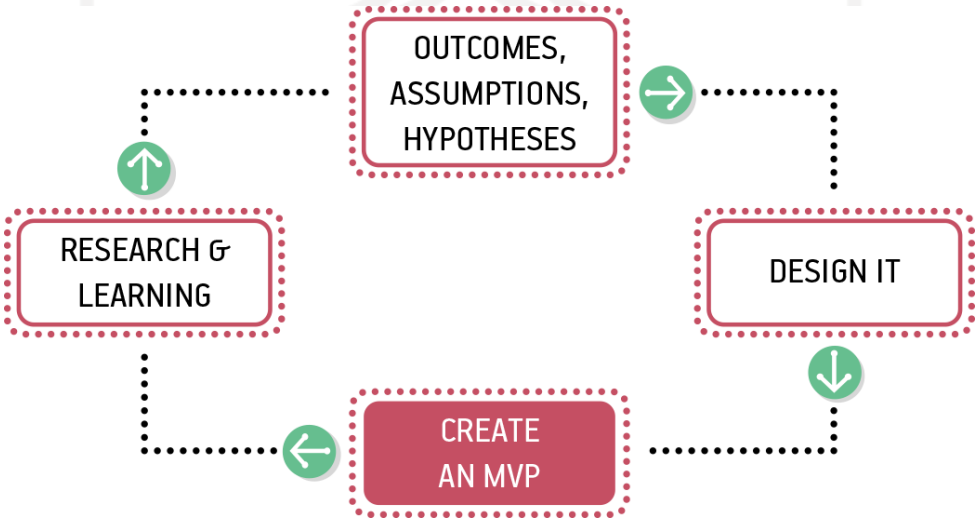
API Testing	Postman sangat berguna untuk melakukan pengujian terhadap API yang telah dibuat. Pengujian ini sangat penting dilakukan untuk memastikan semua *endpoint* berjalan sesuai dengan yang diharapkan. Pengujian dengan postman yang dapat dilakukan adalah CRUD.	Postman

3.3 Desain *(Designing and Wireframing)*

Tahap selanjutnya adalah desain, pada tahap ini proses desain dibagi menjadi dua proses, yaitu proses desain tampilan website UI/UX *(user interface* - antarmuka / *user experience* - pengalaman*)* dan desain rancangan database website.

3.3.1 Desain UI/UX

Proses desain UI/UX utamanya dilakukan oleh tim hipster pada umumnya, namun pada saat pengembangan dilakukan secara bersama dalam rangka saling memberi masukan dan bantuan agar sistem yang dikembangkan dapat selesai dengan cepat. Desain UI/UX tidak terlepas dari informasi awal yang didapatkan pada tahap pengumpulan informasi. Meski begitu tetap ada beberapa hal yang harus disesuaikan baik itu dikurangi atau ditambahkan untuk menciptakan *user experience* yang baik. Untuk sebuah *startup* yang membutuhkan pengembangan cepat, proses desain UI/UX menggunakan metode lean UX. Lean UX berfokus untuk mencari cara tercepat untuk mencapai tujuan akhir dibanding proses desain UX tradisional. Menurut (Jeff Gothelf, 2021) proses lean UX terdiri dari empat (4) seperti pada gambar 3.5 berikut



Gambar 3.5 Lean UX *process*

3.3.2 Desain *Database* atau *Database Modeling*

Setelah desain UI/UX tentunya sudah tergambar dengan jelas bagaimana tampilan sistem, alur bisnis dan juga fitur-fitur yang akan ada pada sistem yang dikembangkan. Oleh karena itu selanjutnya adalah tahap desain *database.* Pada tahap ini desain *database* disesuaikan dengan model database yang digunakan dalam hal ini adalah mongo DB, karena Mongo DB merupakan jenis database NoSQL maka desain atau modeling antara dokumen tidak seperti pemodelan atau desain database MySQL yang menggunakan tabel untuk pemodelannya, Menurut developer Mongo DB, pemodelan Mongo DB berbasis pada data berbentuk JSON yang mana pemodelan hanya menggunakan baris kode hal ini sesuai dengan pemodelan Mongo DB (Karlsson, 2020).

Rancangan *database* Ubaform telah dirancang sesuai kebutuhan pengguna serta kebutuhan dari pihak Ubaform sendiri untuk mendukung proses komersial layanan Ubaform itu sendiri dan juga memudahkan pengelolaan terhadap pengguna. Desain *database* Ubaform dilakukan untuk kedua sistem admin dan *customer*, meskipun begitu kedua database disatukan karena memiliki hampir kesamaan desain kecuali pada desain dokumen user. Untuk desain atau modeling *schema* dari *database* tersebut dapat dilihat sebagai berikut:

- *User Customer Schema*

```
  name: {
    type: String,
    required: true,
    min: 3,
  },
  email: {
    type: String,
    unique: true,
    required: true,
  },
  username: {
    type: String,
    unique: true,
    required: true,
  },
  hashed_password: {
    type: String,
    min: 8,
  },
  salt: {
     type: String
  },
  subscription: [],
  stripe_customer_id: {
```

```
    type: String
  },
  profile: {
    Type: String
  },
  {timestamp}
```

Gambar 3.6 *Schema User Customer*

Desain *schema* pada gambar 3.6 merupakan desain untuk *user customer*. Desain *user* terdiri dari beberapa atribut untuk menampung data *customer* pada saat melakukan registrasi pada web *customer* Ubaform.

- *User* Admin *Schema*

```
  name: {
    type: String,
    required: true,
    min: 3,
  },
  email: {
    type: String,
    unique: true,
    required: true,
  },
  username: {
    type: String,
    unique: true,
    required: true,
  },
  hashed_password: {
    type: String,
    min: 8,
  },
  salt: {
    type: String
  },
  role: {
    type: Number
  },
  profile: {
    Type: String
  },
  {timestamp}
```

Gambar 3.7 *Schema User* Admin

Desain *schema* pada gambar 3.7 merupakan desain untuk *user* Admin. Desain *user* terdiri dari beberapa atribut untuk menampung data admin *(employee)* pada saat melakukan registrasi

pada web admin Ubaform. Perbedaan untuk kedua *user schema* antara admin dan *customer* terletak pada atribut `role` yang mana atribut ini akan memvalidasi apakah *user* yang mengakses web admin Ubaform adalah *user* yang sudah diberikan hak akses atau belum.

- *Schema* Desain Formulir

```
{
  id: {
    type: String,
    required: true,
    index: true,
    unique: true,
    trim: true,
  },
  formName: {
    type: String,
    required: true,
    default: 'Untitled Form',
    max: '32',
  },
  docName: {
    type: String,
    required: true,
    default: 'Untitled Document',
    max: '32',
  },
  docDesc: {
    type: String,
    required: true,
    default: 'Add Description',
    max: '32',
  },
  questions: {
    type: Array,
    default: [
       {questionText: ''},
       {questionType: ''},
       {options: []},
       {answer: ''}
    ],
  },
  theme: {
    fontType: {
      type: String,
      default: '',
    },
    paperColor: {
      type: String,
      default: '',
    },
    backgroundPaperColor: {
      type: String,
      default: '',
    },
```

```
      backgroundImagePaperColor: {
        type: String,
        default: '',
      },
      questionColor: {
        type: String,
        default: '',
      },
      answerColor: {
        type: String,
        default: '',
      },
      stripColor: {
        type: String,
        default: '',
      },
      backgroundButtonColor: {
        type: String,
        default: '',
      },
    },
    createdBy: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Customer',
    },
    respondens: {
      type: [String],
      default: [],
    },
  },
  {timestamps: true},
```

Gambar 3.8 *Schema* Desain *database* Formulir.

Desain *schema* pada gambar 3.8 merupakan desain untuk formulir. Desain formulir terdiri dari beberapa atribut untuk menampung data projek formulir pada saat melakukan pembuatan formulir pada web *customer* Ubaform.

- *Schema* Desain Kuis

```
  {
    id: {
      type: String,
      required: true,
      index: true,
      unique: true,
      trim: true,
    },
    quizName: {
      type: String,
      required: true,
      default: 'Untitled Quiz,
      max: '32',
```

```
  },
  docName: {
    type: String,
    required: true,
    default: 'Untitled Document',
    max: '32',
  },
  docDesc: {
    type: String,
    required: true,
    default: 'Add Description',
    max: '32',
  },
  questions: {
    type: Array,
    default: [
       {questionText: ''},
       {questionType: ''},
       {options: []},
       {answer: ''},
       {point: ''},
       {require: ''},
       {times: ''},
    ],
  },
  theme: {
    fontType: {
      type: String,
      default: '',
    },
    paperColor: {
      type: String,
      default: '',
    },
    backgroundPaperColor: {
      type: String,
      default: '',
    },
    backgroundImagePaperColor: {
      type: String,
      default: '',
    },
    questionColor: {
      type: String,
      default: '',
    },
    answerColor: {
      type: String,
      default: '',
    },
    stripColor: {
      type: String,
      default: '',
    },
    backgroundButtonColor: {
      type: String,
      default: '',
    },
  },
```

```
    createdBy: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Customer',
    },
    respondens: {
      type: [String],
      default: [],
    },
  },
  {timestamps: true},
```

Gambar 3.9 *Schema* Desain *database* Kuis

Desain *schema* pada gambar 3.9 merupakan desain untuk kuis. Desain kuis terdiri dari beberapa atribut untuk menampung data projek kuis pada saat melakukan pembuatan kuis pada web *customer* Ubaform.

- *Schema* Desain Survei

```
{
  id: {
    type: String,
    required: true,
    index: true,
    unique: true,
    trim: true,
  },
  surveiName: {
    type: String,
    required: true,
    default: 'Untitled Survei,
    max: '32',
  },
  docName: {
    type: String,
    required: true,
    default: 'Untitled Document',
    max: '32',
  },
  docDesc: {
    type: String,
    required: true,
    default: 'Add Description',
    max: '32',
  },
  questions: {
    type: Array,
    default: [
       {questionText: ''},
       {questionType: ''},
       {options: []},
       {answer: ''}
```

```
      ],
    },
    theme: {
      fontType: {
        type: String,
        default: '',
      },
      paperColor: {
        type: String,
        default: '',
      },
      backgroundPaperColor: {
        type: String,
        default: '',
      },
      backgroundImagePaperColor: {
        type: String,
        default: '',
      },
      questionColor: {
        type: String,
        default: '',
      },
      answerColor: {
        type: String,
        default: '',
      },
      stripColor: {
        type: String,
        default: '',
      },
      backgroundButtonColor: {
        type: String,
        default: '',
      },
    },
    createdBy: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Customer',
    },
    respondens: {
      type: [String],
      default: [],
    },
  },
  {timestamps: true},
```

Gambar 3.10 *Schema* Desain *database* Survei.

Desain *schema* pada gambar 3.10 merupakan desain untuk survei. Desain survei terdiri dari beberapa atribut untuk menampung data projek survei pada saat melakukan pembuatan survei pada web *customer* Ubaform

- *Schema* Desain *Template*

```
{
  templateName: {
    type: String,
    required: true,
    default: 'Untitled Form',
    max: '32',
  },
  docName: {
    type: String,
    required: true,
    default: 'Untitled Document',
    max: '32',
  },
  docDesc: {
    type: String,
    required: true,
    default: 'Add Description',
    max: '32',
  },
  questions: {
    type: Array,
    default: [
       {questionText: ''},
       {questionType: ''},
       {options: []},
       {answer: ''}
    ],
  },
  type: {
    type: String,
    enum: ['FORMS', 'QUIZ', 'SURVEY'],
    required: true,
  },
  selectedFile: String,
  theme: {
    fontType: {
      type: String,
      default: '',
    },
    paperColor: {
      type: String,
      default: '',
    },
    backgroundPaperColor: {
      type: String,
      default: '',
    },
    backgroundImagePaperColor: {
      type: String,
      default: '',
    },
    questionColor: {
      type: String,
      default: '',
    },
    answerColor: {
      type: String,
```

```
      default: '',
    },
    stripColor: {
      type: String,
      default: '',
    },
    backgroundButtonColor: {
      type: String,
      default: '',
    },
  },
  createdBy: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Customer',
  },
},
{timestamps: true},
```

Gambar 3.11 *Schema* Desain *database Template*.

Desain *schema* pada gambar 3.11 merupakan desain untuk *template*. Desain *template* terdiri dari beberapa atribut untuk menampung data *template* pada saat melakukan pembuatan *template* pada web Admin Ubaform. Template dibuat oleh admin untuk digunakan oleh *customer* pada saat membuat projek.

- *Schema* Desain Tema

```
name: {
  type: String,
  required: true,
},
slug: {
  type: String,
  slug: 'name',
},
description: {
  type: String,
  required: true,
},
type: {
  type: String,
  enum: ['PRO', 'BASIC'],
  required: true,
},
selectedFile: String,
style: {
  fontType: {
    type: String,
  },
  paperColor: {
    type: String,
  },
  backgroundPaperColor: {
```

```
    type: String,
  },
  backgroundImagePaperColor: {
    type: String,
  },
  questionColor: {
    type: String,
  },
  answerColor: {
    type: String,
  },
  stripColor: {
    type: String,
  },
  backgroundButtonColor: {
    type: String,
  },
```

Gambar 3.12 *Schema* Desain *database* Tema.

Desain *schema* pada gambar 3.12 merupakan desain untuk tema. Desain tema terdiri dari beberapa atribut untuk menampung data tema pada saat melakukan pembuatan tema pada web Admin Ubaform. Tema dibuat oleh admin untuk digunakan oleh *customer* pada saat membuat projek.

3.4 Pembuatan konten *(Content Creation)*

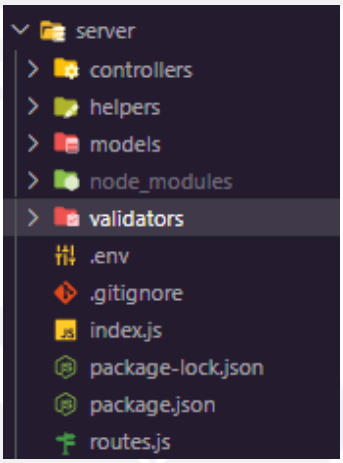
Tahap pembuatan konten penting dilakukan agar saat tahap pengembangan tidak terlalu menggunakan data *dummy* karena itu akan memakan waktu lagi untuk mengubah nya kembali. Oleh karena itu pembuatan konten dilakukan secara terpisah dari tahap lainnya sebelum implementasi atau koding dilakukan.

Pada tahap ini pembuatan konten dilakukan secara bersama dengan anggota tim lain nya berdasarkan hasil analisis atau pengumpulan informasi yang dilakukan sebelumnya. Selain dikomunikasikan dengan kostumer atau pengguna tahap pembuatan konten juga dilakukan dengan mengumpulkan informasi tambahan melalui proses penelitian terhadap kompetitor. Tahap ini konten yang dibuat dapat berupa gambar, text, audio dan lain sebagainya yang memang harus ada dalam website Ubaform.

3.5 Koding dan pengembangan *(Code and Development)*

Tahap penulisan kode atau pengembangan, pada tahap ini web *startup* Ubaform mulai diimplementasikan MERN dan dikembangkan berdasarkan desain yang telah dibuat sebelumnya, baik itu desain UI/UX maupun desain atau modeling *database.* Tentu saja tujuan

dari proses koding adalah untuk merealisasikan semua hasil desain tersebut menjadi sebuah situs web yang dinamis. Pada tahap ini implementasi Express JS, Node JS dan MongoDB dimulai, alur koding dimulai dengan membuat API terlebih dahulu dan melakukan inisiasi *backend* sehingga menghasilkan kerangka kerja atau struktur folder seperti pada gambar 3.13 berikut



Gambar 3.13 Struktur folder atau kerangka kerja sisi *backend*

Pada gambar 3.13 kerangka kerja terdiri dari beberapa *file* dan folder utama serta kongfigurasi lainnya. Pada kerangka kerja ini alur kode mulai dieksekusi dari *file* `index.js` yang berfungsi sebagai dokumen yang pertama kali dijalankan ketika server diaktifkan dan pada dokumen ini server akan di *create*, menentukan port mana yang digunakan dan *endpoint* serta *base* URL. *Source code* dari dokumen `index.js` dapat dilihat pada gambar 3.14 berikut

```
const express = require('express');
var cors = require('cors');
const morgan = require('morgan');
const bodyParser = require('body-parser');
const cookieParser = require('cookie-parser');
const mongoose = require('mongoose');
require('dotenv').config();

// IMPORT FILES
const routes = require('./routes');

// app
const app = express();

app.use(express.json({ limit: '50mb' }));
app.use(express.urlencoded({ limit: '50mb' }));
```

```
// middleware
app.use(morgan('dev'));
app.use(bodyParser.json());
app.use(cookieParser());

app.use(
  cors({
    credentials: true,
    origin: true,
  })
);
app.options('*', cors());

// ROUTES
app.get('/', (req, res) => res.send('Work, Check /health'));
app.get('/health', (req, res) =>
  res.json({ message: 'Hello, Ubaform API!!!' })
);
app.use('/api/v1', routes);

mongoose
  .connect(process.env.MONGO_URL_ONLINE)
  .then(() => console.log('Mongo DB Connected'));

const PORT = process.env.PORT || 9000;

app.listen(PORT, function () {
  console.log(`Server Running On Port ${PORT}`);
  console.log('Connecting to Database...');
});
```

Gambar 3.14 *Source Code* pada dokumen index.js

Pada gambar 3.14 dapat dilihat ada beberapa baris kode, baris kode tersebut menginisiasi server dan menentukan port mana yang digunakan untuk menjalankan server yang telah di *create* kemudian menyambungkan ke database melalui mongo URL *connection.* Pada pengembangan *startup* Ubaform Mongo DB yang digunakan adalah yang versi *online* atau versi atlas.

Alur selanjutnya kode yang akan dieksekusi adalah *file* `routes.js`. *File* ini berisi *endpoint* API yang digunakan untuk menghubungkan server *(backend)* dengan *frontend.* dapat dilihat pada gambar 3.30 berikut

```
const express = require('express');
const router = express.Router();

const {
  preRegister,
  register,
  login,
```

```
      googleLogin,
      logout,
      requireLogin,
      authMiddleware,
      adminMiddleware,
      canUpdateAndDelete,
      resetPassword,
      forgotPassword,
    } = require('./controllers/conAuth');

    const {
      getAllThemes,
      // list,
      getTheme,
      addTheme,
      editTheme,
      deleteTheme,
    } = require('./controllers/conTheme.js');
    const {
      getAllTemplates,
      getSingleTemplate,
      createTemplate,
      editTemplate,
      deleteTemplate,
    } = require('./controllers/conTemplate.js');

    const {
      getAllForms,
      getFormByUser,
      createForm,
      updateForm,
      getSingleForm,
      deleteForm,
    } = require('./controllers/conForm.js');
    // const {
    //   getAllCustomer,
    //   getSingleCustomer,
    // } = require('./controllers/conCustomer.js');

    // VALIDATORS
    const { runValidation } = require('./validators');
    const {
      userSignupValidator,
      userSigninValidator,
      forgotPasswordValidation,
      resetPasswordValidation,
    } = require('./validators/valAuth');

    // AUTH
    router.post('/pre-
register',userSignupValidator,runValidation,preRegister);
    router.post('/register', register);
    router.post('/login', userSigninValidator, runValidation, login);
    router.post('/google-login', googleLogin);
    router.post('/logout', logout);
    router.put(
      '/forgot-password',
      forgotPasswordValidation,
      runValidation,
```

```
  forgotPassword
);
router.put(
  '/reset-password',
  resetPasswordValidation,
  runValidation,
  resetPassword
);

router.get('/themes', getAllThemes);
// router.post('/add-theme', addTheme);
// router.get('/theme/:id', getTheme);
// router.patch('/edit/:id', editTheme);
// router.delete('/delete/:id', deleteTheme);

router.get('/templates', getAllTemplates);

router.get('/forms', getAllForms);
router.get('/forms/:id', getFormByUser);
router.get('/form/:id', getSingleForm);
router.post('/form', requireLogin, authMiddleware, createForm);
router.patch('/form/:id', requireLogin, authMiddleware, updateForm);
router.delete('/form/:id', requireLogin, authMiddleware, deleteForm);
// router.post('/add-template', createTemplate);
// router.get('/template/:id', getSingleTemplate);
// router.patch('/edit-template/:id', editTemplate);
// router.delete('/delete-template/:id', deleteTemplate);

// router.get('/customers', getAllCustomer);
// router.get('/customer/:id', getSingleCustomer);

module.exports = router;
```

Gambar 3.15 *Source Code* pada dokumen routes.js

Pada *source code* gambar 3.15 berisi *endpoint* yang menjadi jembatan penghubung untuk sisi *frontend* dan *backend* yang mana penggunaan REST API yang memungkinkan setiap *request* dari sisi *frontend* memerlukan endpoint sehingga sisi *backend* dapat memberikan respon sesuai *request client*. Endpoint juga membawa satu atau lebih parameter untuk diteruskan ke `controller` sebagai bagian dari sebuah *request*. Selain itu untuk endpoint tertentu yang memerlukan penanganan khusus seperti *endpoint* yang bersifat *private* atau sebaginya dapat ditambahkan sebuah `middleware` yang mengatur aturan penggunaan *endpoint* tersebut. Adapun `middleware` dapat dilihat pada gambar 3.31 berikut

```
const { check } = require('express-validator');

exports.userSignupValidator = [
  check('name').not().isEmpty().withMessage('Name is required'),
  check('email').isEmail().withMessage('Must be a valid email address'),
  check(
```

```
      'password',
      'Password must include one lowercase character, one uppercase
       character, a number, and a special character.'
    ).matches(/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])[0-9a-zA-Z]{8,}$/, 'i'),
  ];

  exports.userSigninValidator = [
    check('email').isEmail().withMessage('Must be a valid email address'),
    check('password').not().isEmpty().withMessage('Password is required'),
  ];

  exports.forgotPasswordValidation = [
    check('email')
      .not()
      .isEmpty()
      .isEmail()
      .withMessage('Must be a valid email address'),
  ];

  exports.resetPasswordValidation = [
    check(
      'newPassword',
      'Password must include one lowercase character, one uppercase
       character, a number, and a special character.'
    )
      .not()
      .isEmpty()
      .matches(/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])[0-9a-zA-Z]{8,}$/, 'i'),

    check('confirmPassword').custom(async (confirmPassword, { req }) => {
      const password = req.body.newPassword;

      console.log('ini newPassword', password);
      console.log('ini newPassword Confirm', confirmPassword);

      // If password and confirm password not same
      // don't allow to sign up and throw error
      if (password !== confirmPassword) {
        throw new Error('Passwords must be same');
      }
    }),
  ];
```

Gambar 3.16 *Source Code* pada dokumen middleware

Pada *source code* gambar 3.16 middleware ini berfungsi untuk melakukan pengecekan atau melakukan validasi penggunaan endpoint registrasi. `Middleware` ini akan menentukan data registrasi yang dilakukan oleh *user* sesuai dengan aturan sistem atau tidak. `Middleware` sendiri jika disederhanakan adalah sekumpulan aturan yang dibuat untuk mengatur penggunaan *endpoint*.

Alur eksekusi kode selanjutnya sampai pada bagian `controller`, pada bagian ini berisi beberapa fungsi yang mengatur serta melakukan eksekusi pada setiap *request* dan *response* berdasarkan *endpoint* sebelumnya. Adapun salah satu `controller` pada sistem Ubaform dapat dilihat pada gambar 3.17 berikut

```
 const FormModel = require('../models/modForm');
const CustomerModel = require('../models/modUser');
const mongoose = require('mongoose');

exports.getAllForms = (req, res) => {
  FormModel.find().exec((err, forms) => {
    if (err) {
      res.status(400).json({ error: 'Something wrong' });
    }

    res.status(200).json(forms);
  });
};

exports.getFormByUser = (req, res) => {
  CustomerModel.findById({ _id: req.params.id }).exec((err, user) => {
    if (err) {
      return res.status(400).json({ error: 'User Not Found' });
    }

    let userId = user._id;

    FormModel.find({ createdBy: userId })
      // .populate('createdBy', '_id name email')
      // .select(
      //   '_id formName docName docDesc createdBy questions theme
          createdAt updatedAt'
      // )
      .exec((err, data) => {
        if (err) {
          return res.status(400).json({ error: 'Form not found' });
        }
        res.json(data);
      });
  });
};

exports.createForm = async (req, res, next) => {
  const {
    // id,
    formName,
    docName,
    docDesc,
    questions,
    theme,
  } = req.body;

  const _id = mongoose.Types.ObjectId();
   console.log('ini body BE',  formName,
```

```
    docName,
    docDesc,
    questions,
    theme);

  let newForm = new FormModel({
    id: _id,
    formName,
    docName,
    docDesc,
    questions,
    theme,
    createdBy: req.user._id,
  });

  console.log('ini new Form BE', newForm);

  newForm.save((err, success) => {
    if (err) {
      return res.status(400).json({ error: err });
    }
    res.status(201).json({
      message: `Form Created !`,
      newForm,
    });
  });
};

exports.updateForm = async (req, res) => {
  const { id } = req.params;
  console.log('ini id', id);
  const {
    // id,
    formName,
    docName,
    docDesc,
    questions,
    theme,
  } = req.body;
  console.log('ini body request', req.body.formName, req.body.docName,
  req.body.docDesc);
  if (!mongoose.Types.ObjectId.isValid(id))
    return res.status(404).send(`No post with id: ${id}`);

  const updateForm = { formName, docName, docDesc, questions, theme, _id:
                       id };
  console.log('ini updateForm', updateForm);
  await FormModel.findByIdAndUpdate(id, updateForm, { new: true });
  res.json({
    message: `Form Updated !`,
    updateForm,
  });
};

exports.getSingleForm = async (req, res) => {
  const { id } = req.params;
  try {
    const form = await FormModel.findById(id);
```

```
    res.status(200).json(form);
  } catch (error) {
    res.status(404).json({ message: error.message });
  }
};

exports.deleteForm = async (req, res) => {
  const { id } = req.params;

  if (!mongoose.Types.ObjectId.isValid(id))
    return res.status(404).send(`No Form with Id ${id}`);

  await FormModel.findByIdAndRemove(id);

  res.json({ message: 'Form Deleted Successfully' });
};
```

Gambar 3.17 *Source code forms* controller.

Pada *source code* gambar 3.17 `controller` formulir terdapat beberapa fungsi CRUD *(Create, Read, Update, Delete)* yang mana akan digunakan untuk membuat formulir, membaca formulir, mengedit formulir dan menghapus formulir. Fungsi-fungsi tersebut akan dijalankan sesuai dengan *endpoint* yang diakses oleh sisi *client.*

Alur eksekusi kode selanjutnya sampai pada bagian `model`, pada bagian ini berisi beberapa baris kode yang memodelkan penyimpanan data pada database. Pemodelan atau desain database ini telah dijelaskan sebelumnya pada poin 3.3. Adapun salah satu `model` pada sistem Ubaform dapat dilihat pada gambar 3.18 berikut

```
const mongoose = require('mongoose');

const formSchema = new mongoose.Schema(
  {
    id: {
      type: String,
      required: true,
      index: true,
      unique: true,
      trim: true,
    },
    formName: {
      type: String,
      required: true,
      default: 'Untitled Form',
      max: '32',
    },
    docName: {
      type: String,
      required: true,
```

```
      default: 'Untitled Document',
      max: '32',
    },
    docDesc: {
      type: String,
      required: true,
      default: 'Add Description',
      max: '32',
    },
    questions: {
      type: Array,
      default: [{ questionText: '' }, { questionType: '' }, { options:
               [] }, {answer : ''}],
    },
    theme: {
      fontType: {
        type: String,
        default: '',
      },
      paperColor: {
        type: String,
        default: '',
      },
      backgroundPaperColor: {
        type: String,
        default: '',
      },
      backgroundImagePaperColor: {
        type: String,
        default: '',
      },
      questionColor: {
        type: String,
        default: '',
      },
      answerColor: {
        type: String,
        default: '',
      },
      stripColor: {
        type: String,
        default: '',
      },
      backgroundButtonColor: {
        type: String,
        default: '',
      },
    },
    createdBy: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Customer',
    },
    // respondens: {
    //   type: [String],
    //   default: [],
    // },
  },
  { timestamps: true },
  // { _id: false }
```

```
    );

    module.exports = mongoose.model('Form', formSchema);
```

Gambar 3.18 *Source code* forms model.

Selain beberapa *file* utama tersebut pada bagian server atau *backend* terdapat *file* kongfigurasi lainnya. *File* kongfigurasi ini sebagai pembantu *file* utama untuk mendapatkan *resource* mulai baris kode yang sensitif atau perlu dirahasiakan sampai pada informasi umum yang memuat cara untuk menjalankan server dan *library* tambahan yang digunakan dalam sistem.

File pendukung pertama adalah `package.json`. *File* `package.json` berisi informasi tentang semua informasi *backend* mulai nama server, nama *library* yang digunakan, cara mengaktifkan server dan lain sebagainya. Untuk lebih lengkap *source code* pada *file* `package.json` dapat dilihat pada gambar 3.19 berikut

```
{
  "name": "backend-api-ubaform",
  "version": "1.0.0",
  "description": "",
  "author": {
    "name" : "nasution",
    "email" : "nasutioncode@gmail.com",
    "git" : "github.com/nasutioncode"
  },
  "main": "index.js",
  "scripts": {
    "start": "NODE_ENV=prod node index.js",
    "dev": "NODE_ENV=dev nodemon index.js",
    "test": "NODE_ENV=test jest --verbose --runInBand",
    "lint": "eslint .",
    "seed": "node ./app/cli/seeder.js"
  },
  "repository": {
    "type": "git"
  },
  "keywords": [],
  "license": "ISC",
  "dependencies": {
    "@meanie/mongoose-to-json": "^2.5.0",
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^8.2.0",
    "email-templates": "^8.0.8",
    "email-validator": "^2.0.4",
    "express": "^4.17.1",
    "express-async-errors": "^3.1.1",
    "google-auth-library": "^6.1.0",
```

```
    "helmet": "^4.1.1",
    "jsonwebtoken": "^8.5.1",
    "moment": "^2.29.1",
    "mongoose": "^6.0.7",
    "mongoose-autopopulate": "^0.12.2",
    "mongoose-slug-generator": "^1.0.4",
    "mongoose-timestamp": "^0.6.0",
    "mongoose-unique-validator": "^2.0.3",
    "morgan": "^1.10.0",
    "nodemailer": "^6.7.0",
    "pug": "^3.0.2",
    "randomstring": "^1.1.5",
    "socket.io": "^3.0.3",
    "validator": "^13.5.1",
    "validatorjs": "^3.20.0"
  },
  "devDependencies": {
    "eslint": "^7.10.0",
    "jest": "^26.4.2",
    "nodemon": "^2.0.4",
    "supertest": "^5.0.0"
  }
}
```

Gambar 3.19 *Source Code* pada Package.json.

File `package.json` seperti sebuah konfigurasi untuk keseluruhan sisi *backend*, seperti yang terlihat pada gambar 3.19 dokumen ini juga mencatat *library* apa saja yang digunakan pada sisi *backend* di dalam *scope* `dependencies` dan `devDependencies`. Kedua nya memiliki perbedaan yang mana `dependencies` *library* yang juga dipakai pada tahap pengembangan dan produksi sedangkan `devDependencies` untuk tahap pengembangan saja.

File kedua adalah `.env`. File `.env` merupakan sebuah dokumen kongfigurasi yang bersifat rahasia yang mana dokumen ini akan berisi port yang digunakan oleh server, URI mongo DB, dan lain sebagainya. *Source code* dari dokumen `.env` dapat dilihat pada gambar 3.20 berikut

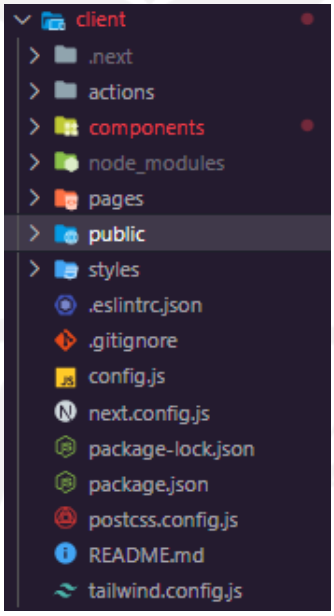
```
PORT=7071
MONGODB_URI='
mongodb+srv://<username>:<password>@cluster0.zqvtm.mongodb.net/myFirstDatab
ase?retryWrites=true&w=majority'

JWT_AUTH_TOKEN='secreet'
JWT_AUTH_REFRESH_TOKEN='secreet'
JWT_EMAIL_VERIFY_TOKEN='secreet'
JWT_PASS_RESET_TOKEN='secreet'
```

```
GOOGLE_CLIENT_ID='secreet'
GOOGLE_CLIENT_SECRET='secreet'

GOOGLE_CLIENT_ID_EMAIL='secreet'
GOOGLE_CLIENT_SECRET_EMAIL='secreet'

CLIENT_URL_RESET_PASS='http://localhost:7070/auth/reset-password'
```

Gambar 3.20 *Source Code* pada .env

Seperti yang terlihat pada gambar 3.20 ada beberapa baris kode dengan *value* `secret` yang mana ini bukan *value* sesungguhnya dari kode yang asli akan tetapi tujuan dokumen ini adalah untuk menyimpan *value* yang bersifat rahasia yang bahkan ketika projek di letakan di github atau tempat penyimpanan sejenis tidak dianjurkan, hal ini dikarenakan alasan keamanan dari server itu sendiri.

Setelah membuat API server atau *backend*, eksekusi kode selanjutnya adalah membuat sisi *frontend*. Untuk kerangka kerja atau struktur folder dapat dilihat pada gambar 3.21 berikut



Gambar 3.21 Struktur folder atau kerangka kerja sisi *frontend*

Pada gambar 3.21 kerangka kerja sisi *frontend* terdapat beberapa *file* dan folder yang tersusun. Sama hal nya dengan sisi *backend*, pada sisi ini juga terdapat beberapa *file* atau folder utama dan kongfigurasi. Untuk mengkases API yang telah dibuat sebelumnya pada sisi

backend, sisi *frontend* terdapat folder `actions` yang berfungsi menjalankan *request* ke *endpoint* pada sisi *backend*. Untuk lebih lengkapnya *source code* folder `actions` pada *file* `actForms.js` dapat dilihat pada gambar 3.22 berikut

```
import fetch from 'isomorphic-fetch';
import { API } from '../config';

export const listFormsByUser = async (id) => {
  try {
    const response = await fetch(`${API}/forms/${id}`, {
      method: 'GET',
      headers: {
        Accept: 'application/json',
        'Content-Type': 'application/json',
      },
    });
    return await response.json();
  } catch (err) {
    return console.log(err);
  }
};


export const createForm = async (formData, token) => {
  console.log('ini data form', formData, token);

  try {
    const response = await fetch(`${API}/form`, {
      method: 'POST',
      headers: {
        Accept: 'application/json',
        Authorization: `Bearer ${token}`,
      },
      body: formData,
    });
    return await response.json();
  } catch (err) {
    return console.log(err);
  }
};

export const singleForm = async (id) => {
  try {
    const response = await fetch(`${API}/form/${id}`, {
      method: 'GET',
    });
    return await response.json();
  } catch (err) {
    return console.log(err);
  }
}

export const updateForm = async (formData, token, id) => {
  console.log('ini formData update form', formData, token, id);
```

```
  try {
    const response = await fetch(`${API}/form/${id}`, {
      method: 'PATCH',
      headers: {
        Accept: 'application/json',
        Authorization: `Bearer ${token}`,
      },
      body: formData,
    });
    return await response.json();
  } catch (err) {
    return console.log(err);
  }
};

export const removeForm = async (id, token) => {
  console.log("ini pass data delete", id, token)
  try {
    const response = await fetch(`${API}/form/${id}`, {
      method: 'DELETE',
      headers: {
        Accept: 'application/json',
        'Content-Type': 'application/json',
        Authorization: `Bearer ${token}`,
      },
    });
    // handleResponse(response);
    return await response.json();
  } catch (err) {
    return console.log(err);
  }
};
```

Gambar 3.22 *Source Code* pada folder *actions file* actForm.js

Pada gambar 3.22 *source code* tersebut akan melakukan request ke sisi *backend* sesuai dengan *endpoint* sehingga *backend* akan mengembalikan *response* tertentu. Pada gambar 3.22 berisi beberapa request pada umumnya seperti *Create, Read, Update, Delete* data. Dalam konteks file pada gambar 3.22 tersebut fungsi CRUD digunakan untuk menangani *request* pengelolaan formulir oleh *customer*.

Alur selanjutnya adalah menampilkan data yang telah didapat dari *request* tersebut ke tampilan halaman web. Untuk lebih lengkapnya *source code* dapat dilihat pada gambar 3.23 berikut

```
   const loadAllThemes = () => {
    return listThemes().then((data) => {
      console.log("ini data", data)
      if (data.error) {
        console.log('Error get Themes');
```

```
      } else {
        setThemes(data);
      }
    });
  };

  useEffect(() => {
    loadAllThemes();
  }, []);

  const showAllThemes = () => {
    return <ListThemes themes={themes} />;
  };


  // <div class='flex flex-wrap -m-4'>{showAllThemes()}</div>
```

Gambar 3.23 *Source code* algoritma menampilkan data

Pada gambar 3.23 fungsi tersebut akan memuat data dengan bantuan `useEffect` dan setelah data dimuat, data tersebut akan ditampilkan dengan fungsi `showAllThemes`. Dengan cara tersebut maka kedua sisi antara *backend* dan *frontend* terhubung dan membentuk satu website dinamis.

Selain *file* dan folder utama tersebut pada sisi *frontend* juga terdapat beberapa *file* pendukung yang berisi *file* kongfigurasi, *reusable component, library* tambahan, informasi terkait cara menjalankan port dan lain sebagainya.

Pertama *file* kongfigurasi `package.json`, *File* ini berisi nama *package* atau library yang terinstal pada sistem untuk membantu pengembangan sisi frontend. Adapaun file `package.json` dapat dilihat pada gambar 3.25 berikut

```
{
  "name": "client",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "@headlessui/react": "^1.4.1",
    "@heroicons/react": "^1.0.5",
    "@material-ui/core": "^4.12.3",
    "@material-ui/icons": "^4.11.2",
    "@popperjs/core": "^2.10.2",
    "axios": "^0.24.0",
    "font-awesome": "^4.7.0",
    "formik": "^2.2.9",
    "isomorphic-fetch": "^3.0.0",
```

```
    "js-cookie": "^3.0.1",
    "moment": "^2.29.1",
    "next": "12.0.5",
    "nprogress": "^0.2.0",
    "react": "17.0.2",
    "react-beautiful-dnd": "^13.1.0",
    "react-color": "^2.19.3",
    "react-confirm-alert": "^2.7.0",
    "react-dom": "17.0.2",
    "react-google-login": "^5.2.2",
    "react-scroll": "^1.8.4",
    "react-slick": "^0.28.1",
    "react-toastify": "^8.1.0",
    "react-uuid": "^1.0.2",
    "reactcss": "^1.2.3",
    "sweetalert": "^2.1.2",
    "uuid": "^8.3.2",
    "yup": "^0.32.11",
    "yup-password": "^0.2.2"
  },
  "devDependencies": {
    "autoprefixer": "^10.4.0",
    "eslint": "8.4.0",
    "eslint-config-next": "12.0.5",
    "postcss": "^8.4.4",
    "tailwindcss": "^2.2.19"
  }
}
```

Gambar 3.24 *Source Code* pada Package.json *frontend*.

File `package.json` seperti sebuah konfigurasi untuk keseluruhan sisi *frontend*, seperti yang terlihat pada gambar 3.24 dokumen ini juga mencatat *library* apa saja yang digunakan pada sisi *backend* di dalam *scope* `dependencies` dan `devDependencies`. Kedua nya memiliki perbedaan yang mana `dependencies` *library* yang juga dipakai pada tahap pengembangan dan produksi sedangkan `devDependencies` untuk tahap pengembangan saja. Pada `package.json` juga ditentukan CLI *command* yang digunakan untuk menjalankan projek atau sistem.

File kongfigurasi selanjutnya adalah `next.config.js`. *File config* ini hampir mirip dengan file `.env` yang ada pada sisi *backend*, namun pada sisi *frontend* dinamakan `next.config.js`. Adapun file `next.config.js` dapat dilihat pada gambar 3.37 berikut

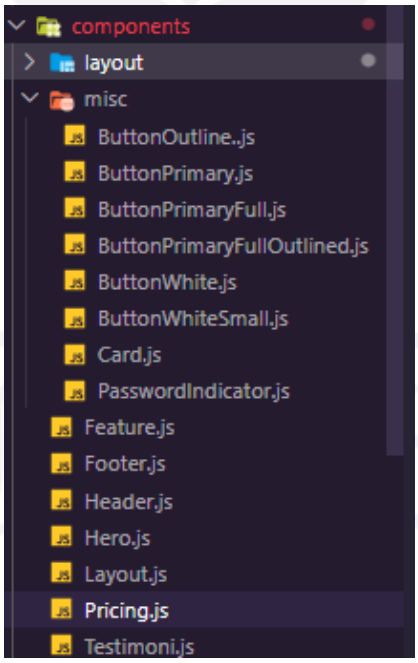
```
module.exports = {
  publicRuntimeConfig: {
    APP_NAME: 'UBAFORM',
    API_DEVELOPMENT: 'http://localhost:9001/api/v1',
```

```
    API_PRODUCTION: 'https://ubaform.com/api/v1',
    PRODUCTION: false,
    DOMAIN_DEVELOPMENT: 'http://localhost:secret',
    DOMAIN_PRODUCTION: 'https://ubaform.com',
    GOOGLE_CLIENT_ID:
      secret'',
  },
};
```

Gambar 3.25 *Source Code* pada next.config.json

Pada gambar 3.25 berisi inisiasi terhadap beberapa baris kode rahasia seperti URL pihak ketiga, base URL dan lain sebagainya.

Selanjutnya beberapa *file* komponen yang bersifat *reusable*. Komponen yang dibuat tentu saja untuk membantu mempermudah dan mempercepat proses pengembanga sistem karena bisa digunakan berulang-ulang sesuai dengan keunggulan dari react JS itu sendiri. Ada beberapa *file reusable component* pada projek Ubaform, *file-file* tersebut dapat dilihat pada gambar 3.26 berikut



Gambar 3.26 *File reusable component*

Dapat dilihat pada gambar 3.26, terdapat banyak *file* yang bisa digunakan secara berulang-ulang dengan dinamis. Hal inilah yang membuat implementasi MERN dapat mempermudah

dan mempercepat waktu pengembangan sistem. Adapun contoh *file* salah satu komponen yaitu komponen `button` dapat dilihat pada gambar 3.39 berikut

```
import React from 'react';

const ButtonPrimary = ({ children, addClass }) => {
  return (
    <button
      className={
        'py-3 lg:py-4 px-12 lg:px-16 text-white-500 font-semibold rounded-
lg bg-primary hover:shadow-orange-md transition-all outline-none ' +
        addClass
      }
    >
      {children}
    </button>
  );
};

export default ButtonPrimary;
```

Gambar 3.27 *Source Code* komponen ButtonPrimary.js

Pada gambar 3.27 komponen `ButtonPrimary.js` dapat digunakan kapan saja dikomponen lain atau halaman lain. Begitu pula dengan pembuatan komponen lain seperti `Header`, `Footer`, `Card` dan lain sebagainya.

3.6 Pengujian *(Testing)*

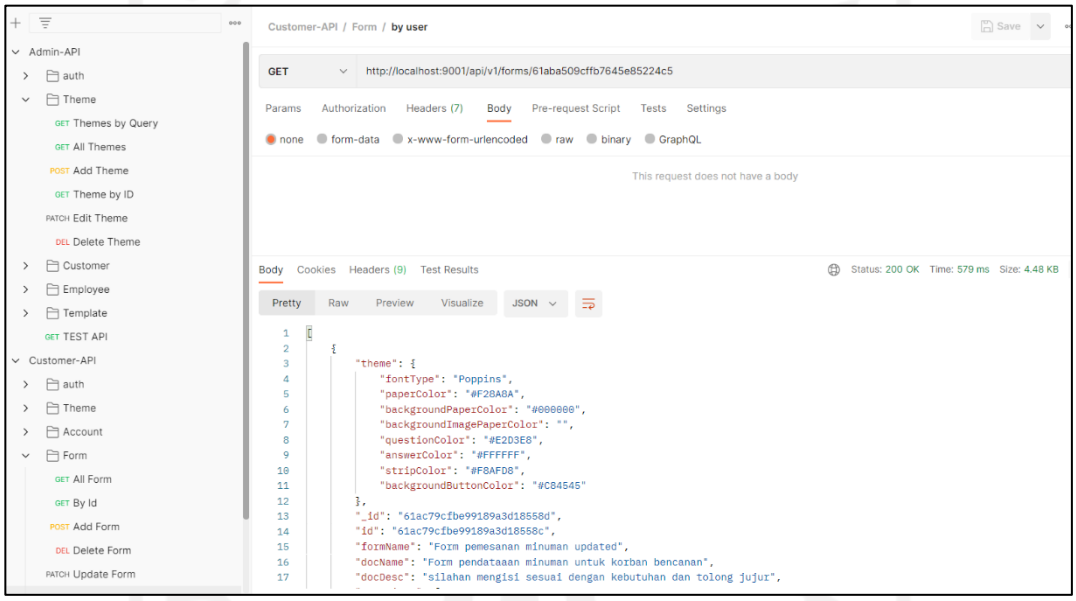
Tahap pengujian dilakukan untuk mengetahui beberapa hal apakah sistem yang dikembangkan sudah sesuai dengan ekspektasi yang seharusnya dan untuk mengetahui sistem berjalan dengan lancar. Tahapan ini mencakup seluruh bagian mulai dari UI/UX dan fungsionalitas sistem. Berikut beberapa pengujian yang dilakukan untuk mengetahui kekurangan, *bug* dan hal yang tidak diinginkan pada saat sistem Ubaform dijalankan

3.6.1 Pengujian UI/UX

Pengujian UI/UX dilakukan untuk mengetahui apakah pengguna bisa mengoperasikan sistem yang dibuat atau tidak. Pengujian ini secara detail akan dibahas oleh bagian hustler nantinya. Namun secara garis besar pengujian ini dilakukan dengan membuat *prototype* sederhana dan dibagikan kepada stakeholder untuk mendapatkan feedback sehingga pengembang dapat mengetahui hal apa saja yang akan diperbaiki.

3.6.2 Pengujian Fungsionalitas sistem

Pengujian fungsionalitas sistem dilakukan untuk mengetahui apakah sistem berjalan sesui dengan fungsinya atau tidak. Pada pengujian ini dilakukan hanya pada sisi *backend* dengan metode manual yaitu melakukan pengecekan terhadap *endpoint* yang dibuat. Pengujian ini dilakukan dengan memanfaatkan salah satu software API yaitu postman. Pengujian yang dilakukan dikatakan berhasil jika mengembalikan respon atau status sesuai dengan apa yang diharapkan. Adapun contoh yang pengujian dengan postman dapat dilihat pada gambar 3.42 berikut



Gambar 3.28 Pengujian API GET data form

Pada gambar 3.28 dilakukan pengujian untuk mengambil data form dengan *method* `GET`. Pengambilan data dilakukan dengan melakukan *request* terhadap endpoint `http://localhost:9001/api/v1/forms/61aba509cffb7645e85224c5` dengan paramater. *Request* ini menghasilkan respon berupa JSON data formulir dengan respon status 200.

Pengujian yang sama dilakukan terhadap semua *endpoint* yang telah dibuat baik itu dengan *method* `GET`, `PUT`, `PATCH`, `DELETE` dan `POST` tergantung *endpoint*.

3.7 Peluncuran dan pemeliharaan *(Deployment and Maintenance)*

Pada tahap peluncuran dilakukan peluncuran Ubaform versi pertama, yang mana tidak diluncurkan keseluruhan fitur yang telah dirancang melainkan hanya MVP dari *startup* Ubaform. Hal ini selain untuk merebut pasar dengan cepat juga sebagai bentuk untuk mencari *feedback* terhadap sistem yang kembangkan. Jika *feedback* dari pengguna memuaskan, sistem dapat lanjut untuk dikembangkan akan tetapi jika *feedback* terkait fitur atau tampilan kurang maka pengembangan sistem dapat diberhentikan sementara untuk memperbaiki tampilan atau fitur terlebih dahulu.

Untuk tahap pemeliharaan juga sangat penting dilakukan dan ini merupakan tahap akhir dari pengembangan sistem. Tahap ini untuk terus memastikan sistem berjalan tanpa kendala serta juga bisa dijadikan tahap untuk menambah fitur atau memperbaiki hal-hal yang tidak diinginkan selama pengguna mengoperasikan sistem.

BAB IV
HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan hasil serta implementasi empat teknologi yang digunakan yaitu Mongo DB, Express JS, React JS, dan Node JS dalam pengembangan web Ubaform. Implementasi tersebut akan dijelaskan dengan hasil dan pembahasan masing-masing. Adapun untuk hasil dan pembahasan dari implementasi MERN adalah sebagai berikut

4.1 Hasil dan Pembahasan Implementasi Mongo DB

Pada implementasi Mongo DB menggunakan versi 4.4.11, implementasi ini dibantu dengan melakukan instalasi *package* pendukung menggunakan NPM. *Package* tersebut yaitu `mongoose versi 6.0.12`, Adapun untuk CLI *(Command Line Interface)* yang digunakan untuk instalasi dapat dilihat pada gambar 4.1 berikut

```
npm install mongoose@6.0.12 --save
```

Gambar 4.1 CLI instalasi *package* mongoose

Package mongoose akan membantu pengembang dalam beberapa hal diantaranya seperti menghubungkan *backend* ke server atau database Mongo DB, membuat skema database dan lain sebagainya. Untuk implementasi skema database dapat dilihat pada bab 3, sub bab ke 3.3.2, sedangkan untuk menghubungkan *backend* ke Mongo DB dapat dilihat pada gambar 4.2 berikut

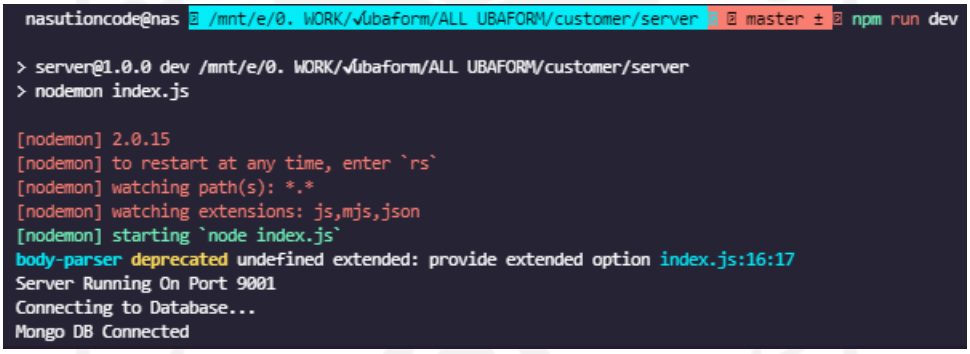
```
mongoose
  .connect(process.env.MONGO_URL_ONLINE)
  .then(() => console.log('Mongo DB Connected'));


const PORT = process.env.PORT || 9000;
app.listen(PORT, function () {
  console.log(`Server Running On Port ${PORT}`);
  console.log('Connecting to Database...');
});
```

Gambar 4.2 *Connecting backend* ke Mongo DB

Pada gambar 4.2 dapat dilihat terdapat dua fungsi utama dari source code tersebut, untuk menghubungkan ke database melalui Mongo URL dan juga menentukan port mana yang digunakan untuk menjalankan server.

4.1.1 Hasil

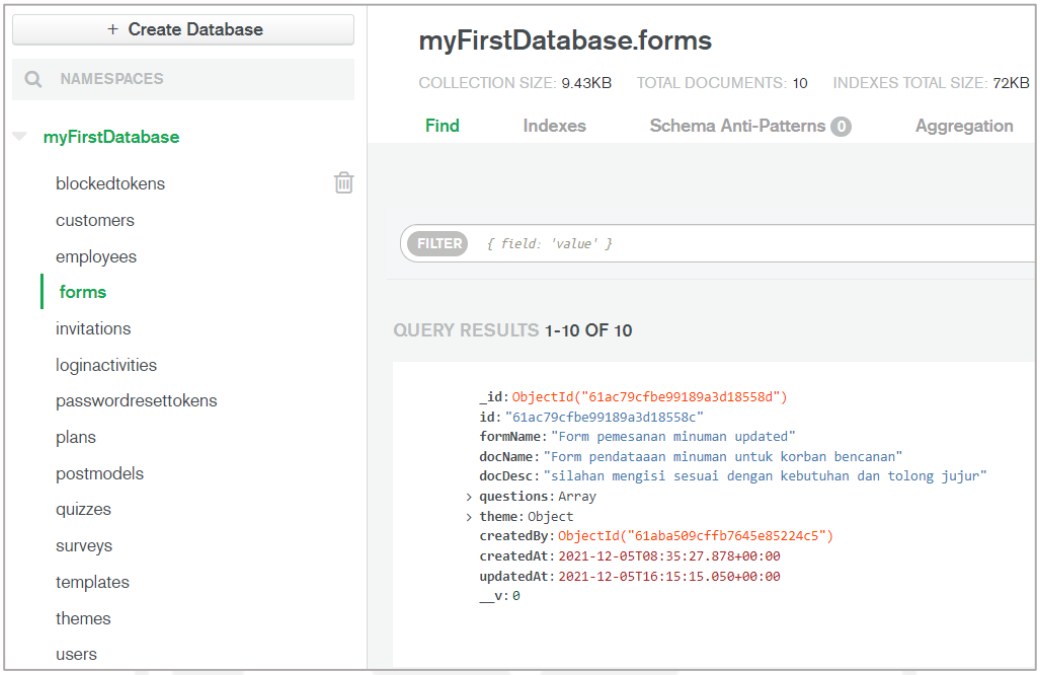
Untuk hasil implementasi dapat dilihat setelah program dijalankan, dalam kasus Mongo DB hasil implementasi dapat dilihat melalui terminal dan juga pada database atlas *user interface* yang disediakan oleh pihak Mongo DB secara *online*. Adapun hasil implementasi pada terminal ketika program dijalankan secara umum dapat memantau kinerja server seperti apakah server terhubung atau mengalami error, meskipun begitu transaksi data juga dapat dilihat menggunakan CLI tertentu namun dalam praktik pengembagan web Ubaform transaksi data dilihat langsung menggunakan Mongo DB atlas *user interfaces*. Adapun hasil implementasi Mongo DB pada terminal dapat dilihat pada gambar 4.3 berikut



Gambar 4.3 Informasi *connecting* Mongo DB

Pada gambar 4.3 dapat dilihat ketika program dijalankan dengan CLI `npm run dev` server langsung dihubungkan ke database Mongo DB. Pada terminal gambar 4.3 akan memantau setiap transaksi CRUD yang dilakukan dan juga informasi terkait program *error* dan lain sebagainya.

Kemudian pada Mongo DB atlas *user interfaces* transaksi data yang disimpan dapat dilihat pada gambar 4.4 berikut

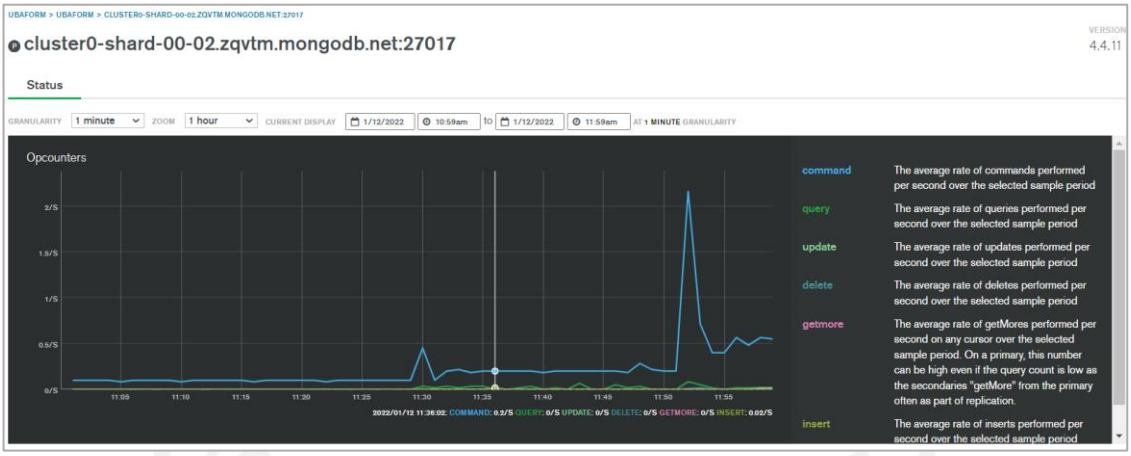


Gambar 4.4 Hasil transaksi data pada Mongo DB atlas UI

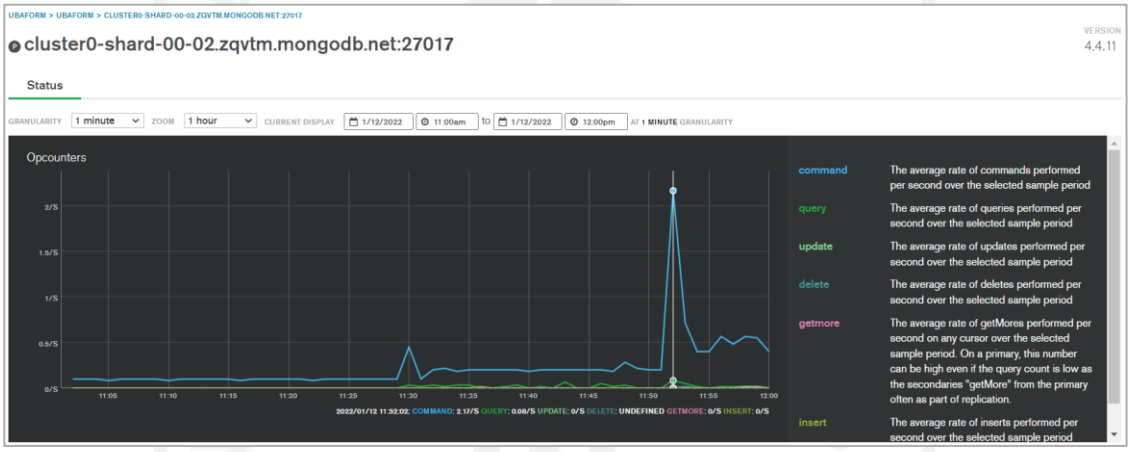
Dapat dilihat pada gambar 4.4 data disimpan berdasarkan *collections* yang ada pada sidebar kiri gambar sedangkan data dari *collections* terdapat pada kanan gambar. Penamaan *collection* didasarkan pada penamaan skema database yang ditulis pada *source code*, namun oleh Mongo DB dibuat plural misalnya yang awalnya *collection* dengan nama form namun secara otomatis diubah menjadi forms.

4.1.2 Pembahasan

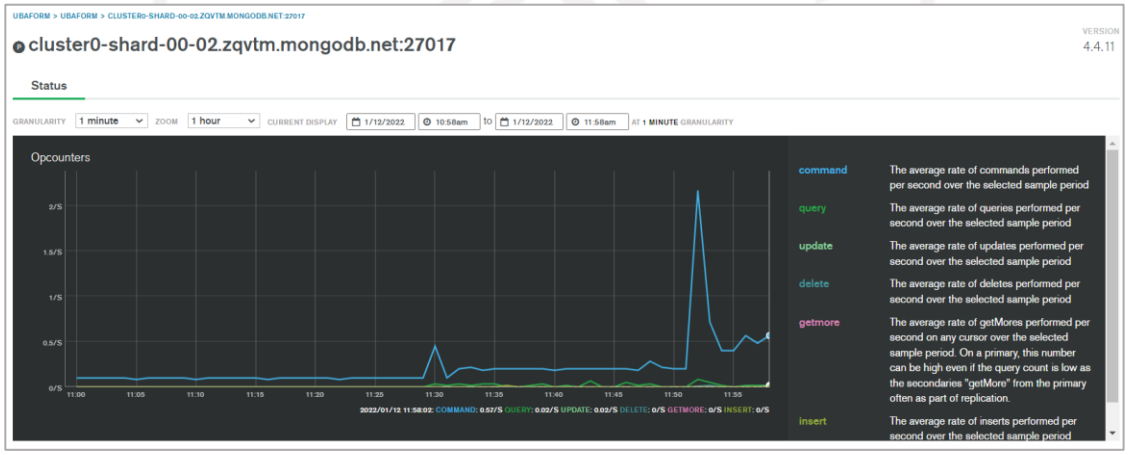
Penggunaan Mongo DB dalam pengembangan web Ubaform yang memiliki fitur atau layanan pembuatan formulir, kuis dan survei dikarenakan dua hal yaitu adalah performa dan skema penyimpanan data yang memudahkan pengembangan. Untuk performa Mongo DB sendiri lebih baik dibandingkan database MySQL dikarenakan tidak ada nya penggabungan data antara tabel yang memperpanjang proses query data pada database. Untuk lebih lengkap performa Mongo DB dapat dilihat pada data pemantauan status transaksi data yang didapat pada Mongo DB atlas UI. Data ini berdasarkan status transaksi data dengan melakukan CRUD pada web Ubaform. Adapun hasil data tersebut dapat dilihat pada gambar 4.5 sampai 4.8 berikut



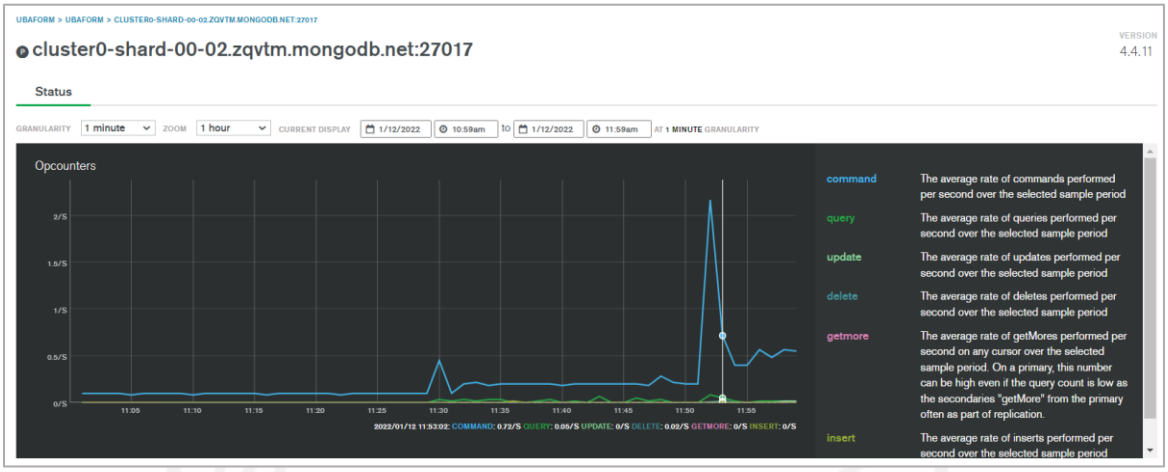
Gambar 4.5 Performa Create/Insert data pada Mongo DB



Gambar 4.6 Performa Read/Select data pada Mongo DB



Gambar 4.7 Performa Update data pada Mongo DB



Gambar 4.8 Performa Delete data pada Mongo DB

Dapat dilihat pada gambar 4.5 sampai 4.8 waktu yang dibutuhkan oleh Mongo DB untuk melakukan INSERT, UPDATE, DELETE data hanya 0.02/s dan query SELECT data hanya 0.08/s paling lama. Hal ini tentunya sangat cepat karena berdasarkan penelitian oleh Kinsta sebelumnya yang juga disebutkan pada latar belakang penelitian ini mengatakan bahwa 1 dari 4 pengunjung akan meninggalkan website jika memuat lebih dari 4 detik dan 46% pengguna tidak mengunjungi kembali situs web dengan performa yang buruk (KINSTA, 2021). Sedangkan dalam performa yang ditujukan oleh Mongo DB bahkan tidak sampai 1 detik.

Selain performa yang cepat pemilihan Mongo DB dikarenakan skema penyimpanan data yang memudahkan dan mendukung dalam pengembangan web Ubaform. Hal ini dikarenakan fitur yang ada pada web Ubaform adalah pembuatan formulir, kuis dan survei yang membutuhkan data yang fleksibel dan terstruktur karena data akan diinput oleh pengguna sesuai kebutuhan, misalnya ketika pengguna akan membuat sebuah formulir tentunya jumlah dan jenis pertanyaan berbeda-beda. Hal ini yang sulit untuk dibuat skema nya jika menggunakan database MySQL karena tidak mendukung fleksibilitas yang terstruktur. Namun ketika menggunakan Mongo DB data tersebut akan dibuat skemanya dalam bentuk data JSON baik secara bersarang *(nested)* maupun tidak. Adapun lebih detailnya dapat dilihat pada gambar 4.9 berikut

Gambar 4.9 Skema Mongo DB fleksibel dan terstruktur

Pada gambar 4.9 dapat dilihat skema penyimpanan data dalam bentuk JSON. Selain itu terdapat data yang disimpan dalam bentuk objek tertentu dalam tipe array. Data tersebut disusun secara terstruktur untuk memudahkan pengembang dalam melakukan query. Kemudian penyimpanan data seperti pada gambar tersebut akan memudahkan pengembang menyimpan data secara fleksibel dikarenakan ketika pengguna akan menambah sebuah atau beberapa pertanyaan dalam pembuatan formulir, kuis atau survei data akan langsung disimpan atau ditambahkan dalam bentuk objek selanjutnya. Hal ini seperti menambah data pada sebuah variabel array yang mana setiap data yang ditambahkan akan mengisi index selanjutnya. Berdasarkan hal tersebut kemudahan dalam mengola data seperti query SELECT, INSERT, DELETE dan UPDATE akan lebih mudah.

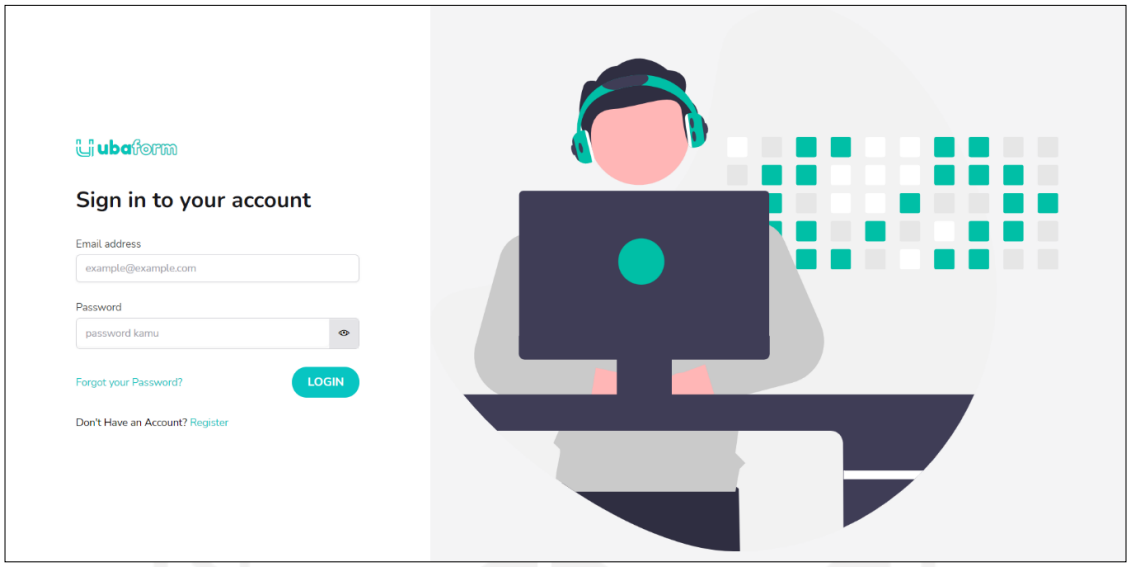
4.2 Hasil dan Pembahasan Implementasi React JS

Pada implementasi React JS untuk pengembangan web Ubaform tentunya tidak terlepas dari kemudahan dan performa yang diinginkan. Pengembangan web Ubaform yang berbasis pada SPA dipercaya lebih cepat dibandingkan dengan web konvensional Muilti page. Hal ini dikarenakan SPA akan melakukan pemuatan halaman web hanya sekali dan jika terjadi perubahan data maka yang dilakukan pemuatan hanya data yang bersangkutan saja tanpa memuat *resource* lainnya yang tidak diperlukan. Hal ini tentu akan menghemat transaksi data seperti *request* dan *response* yang dilakukan ke server sehingga bandwith berkurang. Selain itu kemudahan pada impelementasi React JS yang sangat membantu karena dapat menambah *package* lain menggunakan NPM. Tentu hal ini dapat dilakukan dengan menggunakan *library* lain seperti angular JS, namun React JS lebih fleksibel menambah *library* lain dibandingkan dengan angular JS. Hal ini dikarenakan React JS merupakan *library* sedangkan angular JS merupakan sebuah framework yang mana aturan penggunaan seperti susunan direktori folder sudah ditentukan, hal ini akan membuat penggabungan dengan kode atau *library* lain lebih sulit.

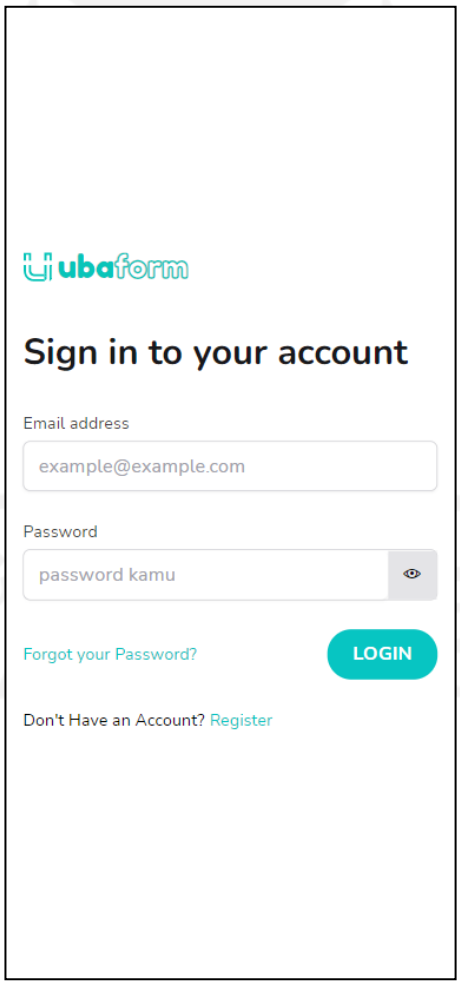
4.2.1 Hasil

Untuk hasil dari implementasi React JS pada pengembangan aplikasi *startup* Ubaform secara keseluruhan yang telah dijelaskan secara lengkap pada laporan ini baik untuk web admin Ubaform dan web *customer.* Web admin digunakan oleh admin *(employee)* untuk melakukan manajemen data-data terkait *startup* Ubaform. Data-data yang dibuat oleh admin akan digunakan pada web *customer*. Web admin dapat diakses pada link https://adminubaform.netlify.app sedangkan untuk web *customer* menjadi tempat layanan yang dapat diakses oleh pengguna pada umumnya untuk membuat formulir, kuis dan survei. Adapun untuk hasil dari web admin dan web *customer* adalah sebagai berikut:

- Halaman *Login* Admin



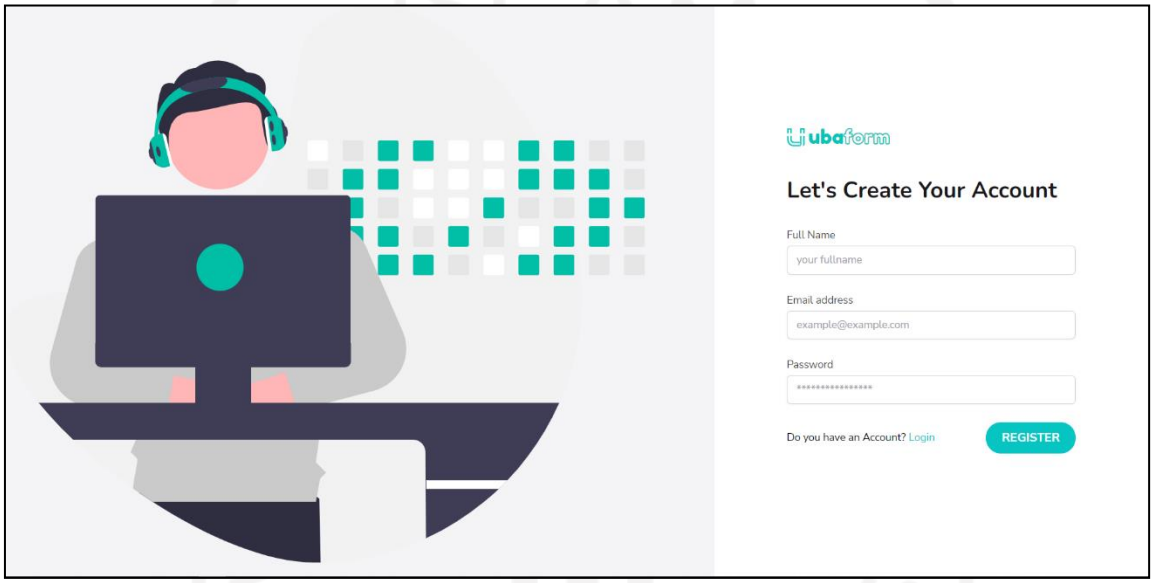
Gambar 4.10 Halaman *Login* Admin Ubaform *(Desktop View)*



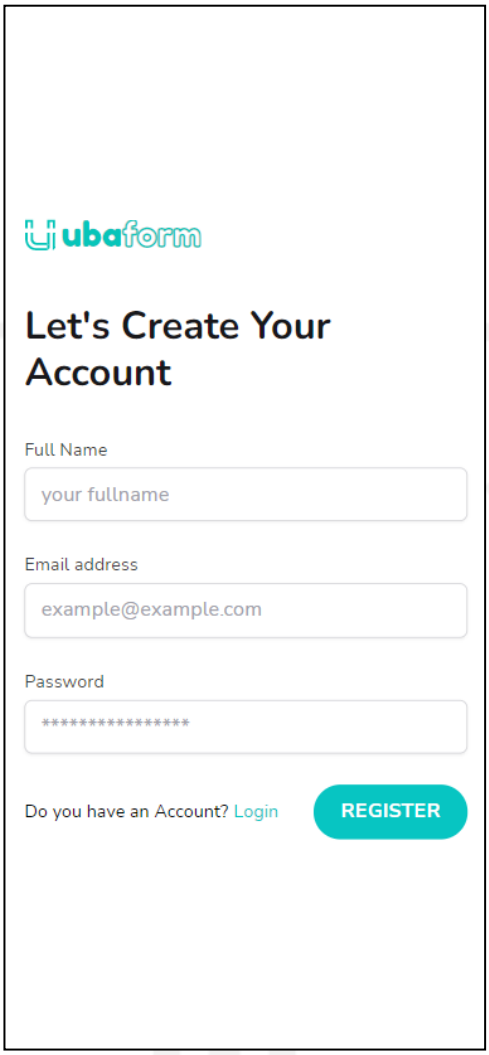
Gambar 4.11 Halaman *Login* Admin Ubaform *(Mobile View)*

Halaman *Login* admin Ubaform adalah yang pertama kali tampil ketika admin *user* mengakses admin Ubaform jika belum melakukan *login* sebelumnya. Halaman ini akan berisi dua *field* yang harus diisi untuk dapat mengakses dashboard Ubaform. Jika admin *user* tidak memiliki akun maka bisa melakukan pendaftaran.

- Halaman *Regsiter* Admin



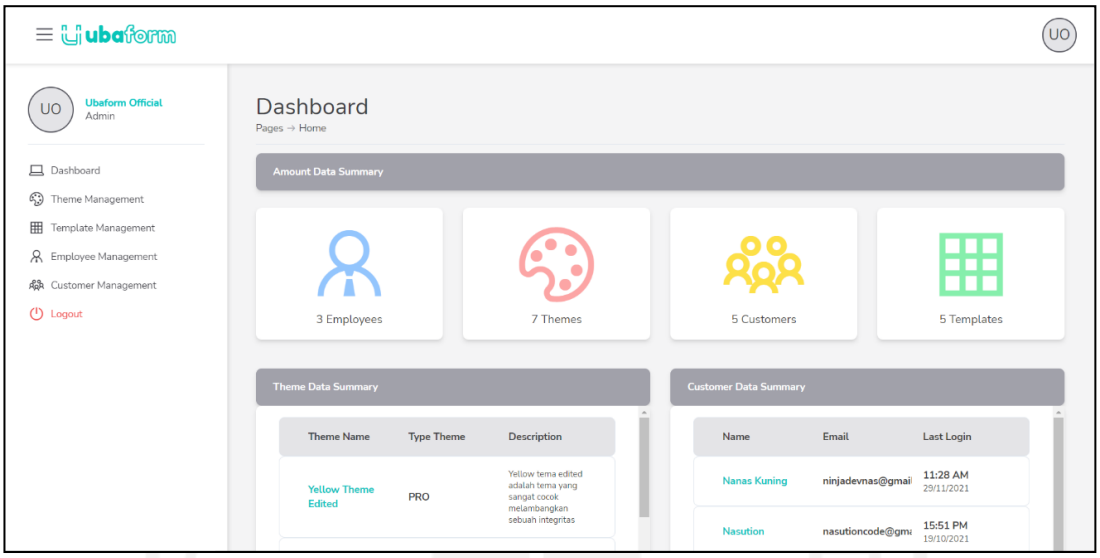
Gambar 4.12 Halaman *Regsiter* Admin Ubaform *(Desktop View)*



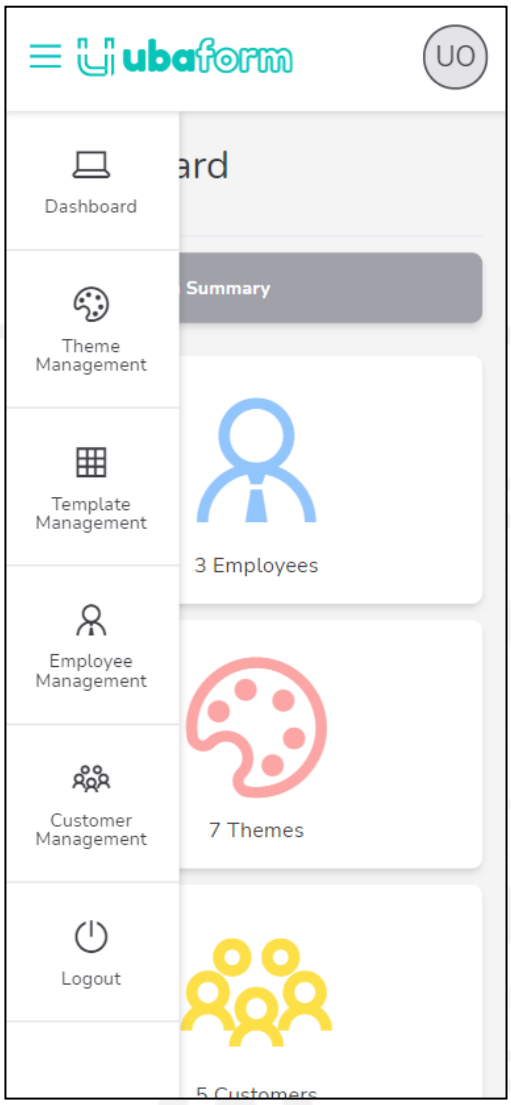
Gambar 4.13 Halaman *Regsiter* Admin Ubaform *(Mobile View)*

Halaman *Register* admin Ubaform adalah halaman untuk calon admin melakukan pendaftaran. Halaman ini terdiri dari tiga *fields* yang harus diisi oleh *user* saat melakukan pendaftaran. Setelah regsitrasi *user* tidak langsung bisa melakukan *login* karena meski data registrasi sudah ada pada database akan tetapi belum dilakukan aktivasi akun oleh admin yang telah aktif sebelumnya.

- Halaman *Home Dashboard* Admin



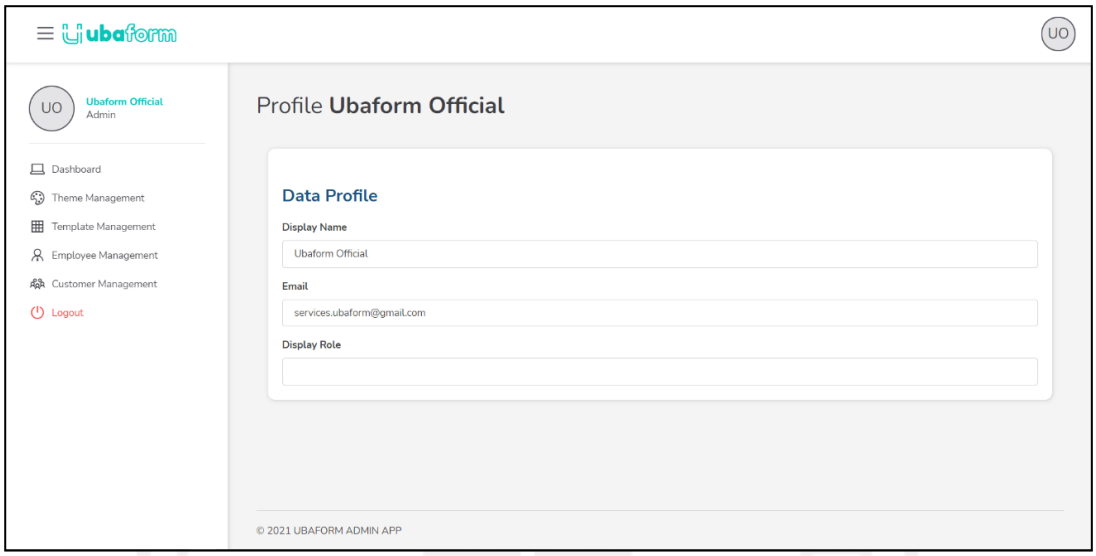
Gambar 4.14 Halaman *Home dahsboard* admin Ubaform *(Desktop View)*



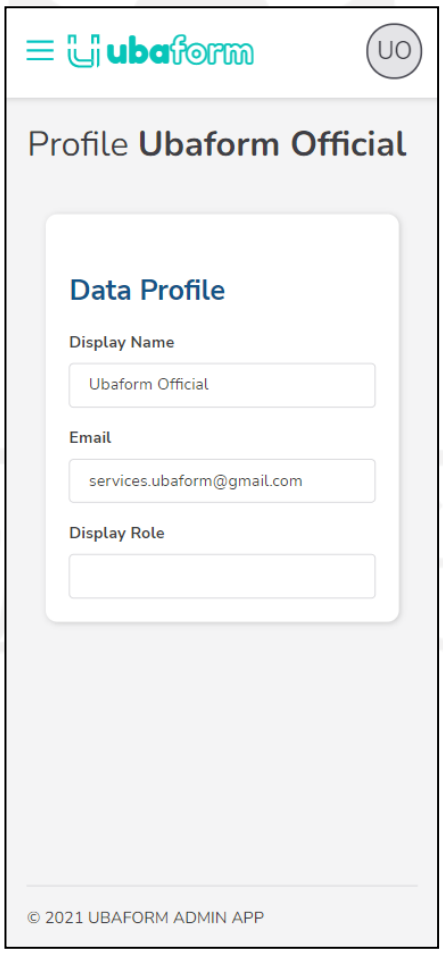
Gambar 4.15 Halaman *Home dashboard* Admin Ubaform *(Mobile View)*

Halaman *home dashboard* admin Ubaform adalah halaman yang dapat diakses oleh admin terdaftar yang telah aktif. Pada halaman *home dashboard* ini admin dapat melihat secara ringkas data yang ada seperti jumlah tema, template, *customer* terdaftar dan lain sebagainya

- Halaman *Profile* Admin



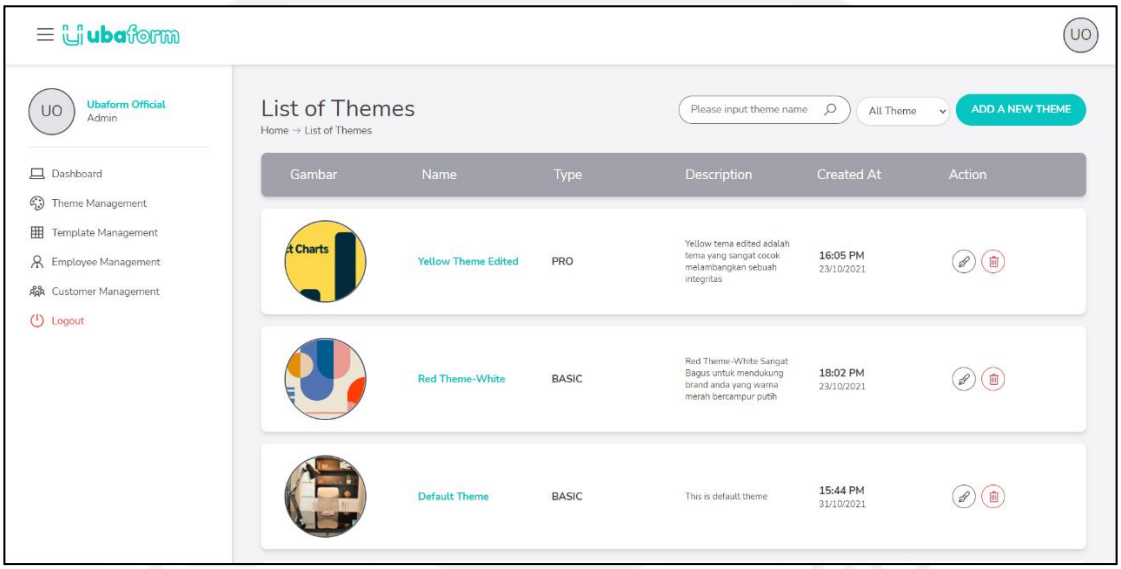
Gambar 4.16 Halaman *Profile* admin Ubaform *(Desktop View)*



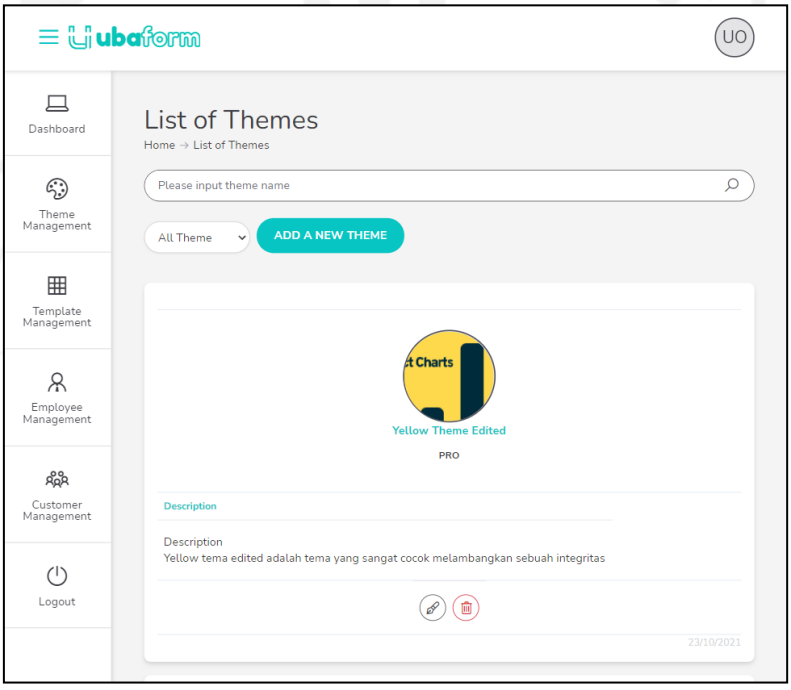
Gambar 4.17 Halaman *profile* Admin Ubaform *(Mobile View)*

Halaman *profile* admin Ubaform adalah halaman yang dapat diakses oleh admin user yang telah melakukan login. Pada halaman *profile* ini admin dapat melihat informasi diri sendiri mulai dai nama, email dan role.

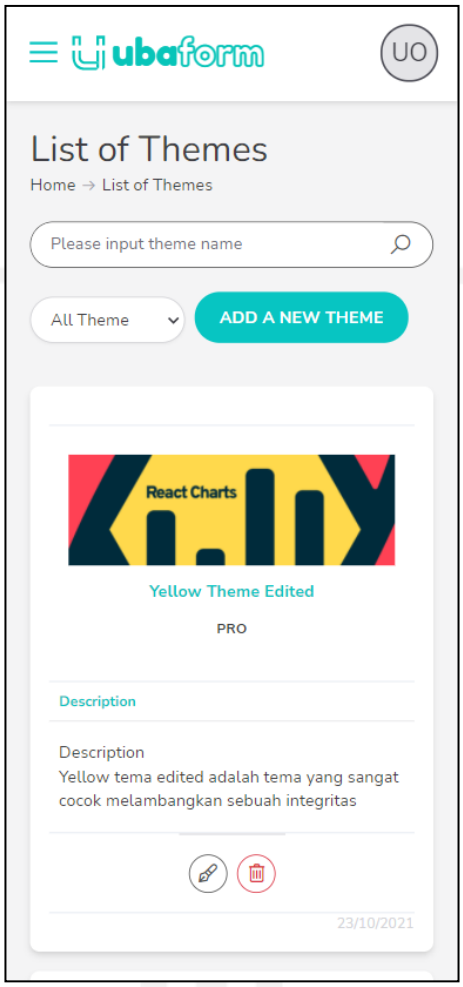
- Halaman *Theme Management (List Theme)*



Gambar 4.18 Halaman *Theme management list theme* admin Ubaform *(Desktop View)*



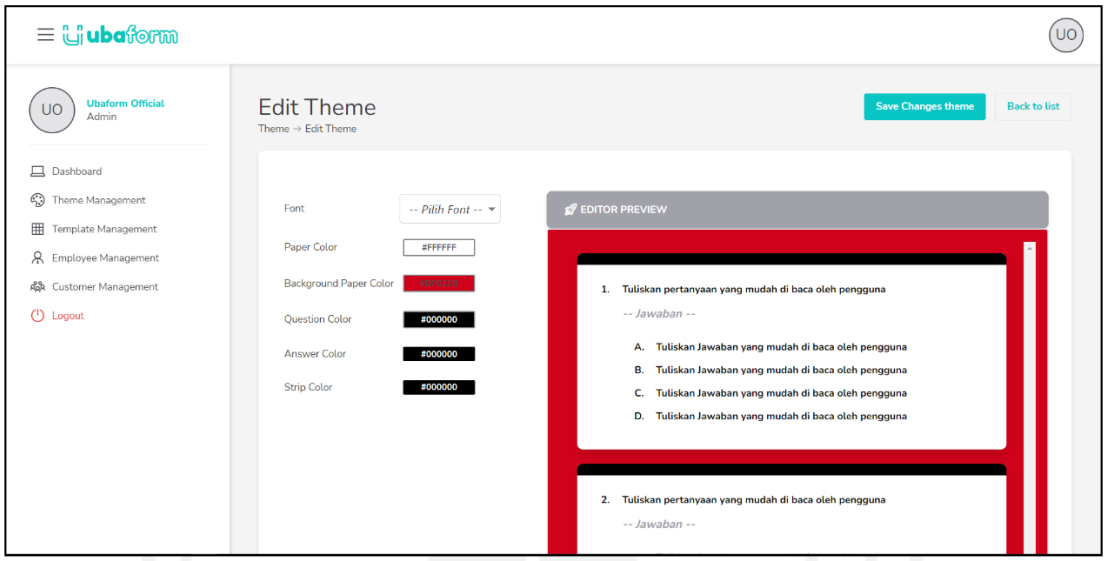
Gambar 4.19 Halaman *Theme management list theme* admin Ubaform *(Tablet View)*



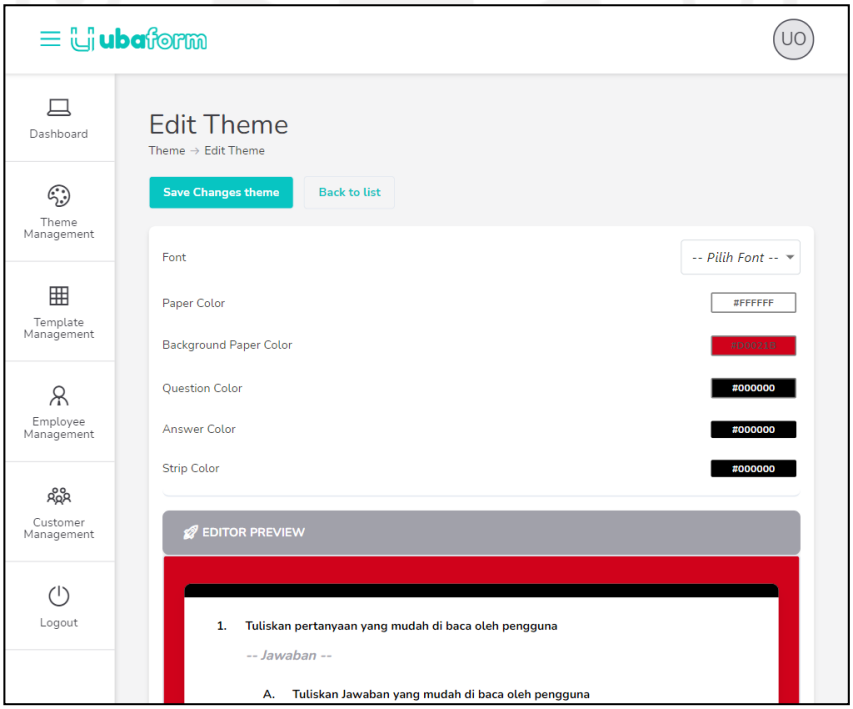
Gambar 4.20 Halaman *Theme management list theme* admin Ubaform *(Mobile View)*

Halaman *theme management list theme* admin Ubaform adalah halaman yang menampilkan daftar keseluruhan tema yang telah dibuat. Atribut tema yang ditampilkan seperti gambar, nama tema, tipe tema dan deskripsi tema. Halaman ini memiliki fitur cari tema, filter tema, edit tema, hapus tema dan buat tema.

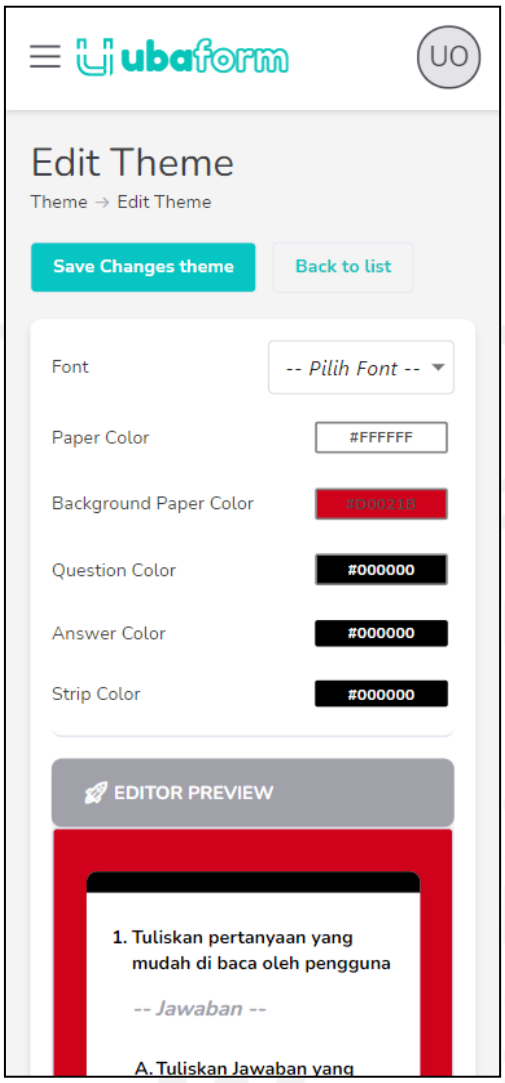
- Halaman *Theme Management (Edit Theme)*



Gambar 4.21 Halaman *Theme management edit theme* admin Ubaform *(Desktop View)*



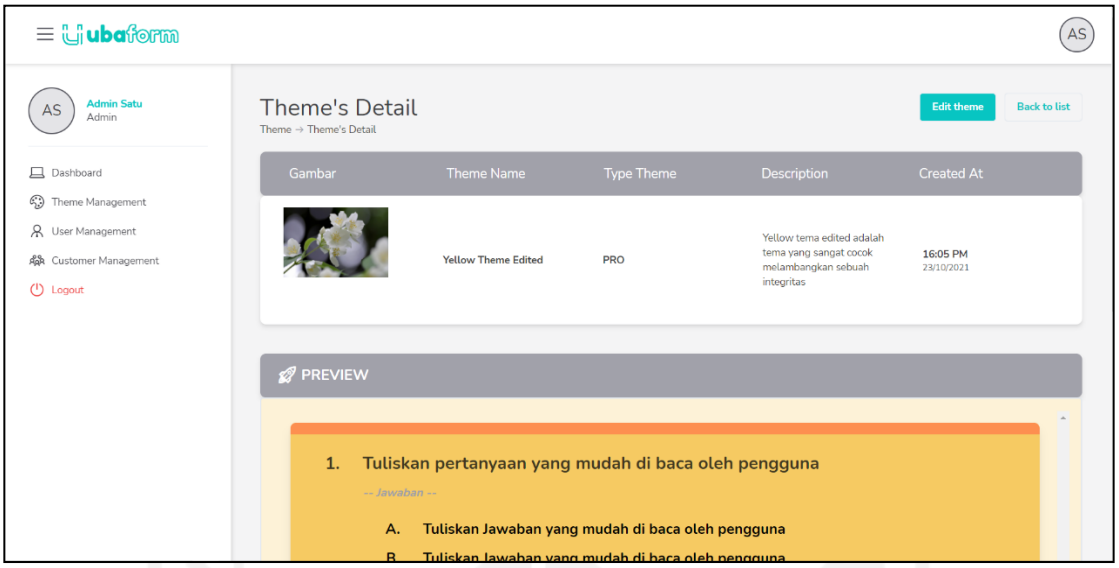
Gambar 4.22 Halaman *Theme management edit theme* admin Ubaform *(Tablet View)*



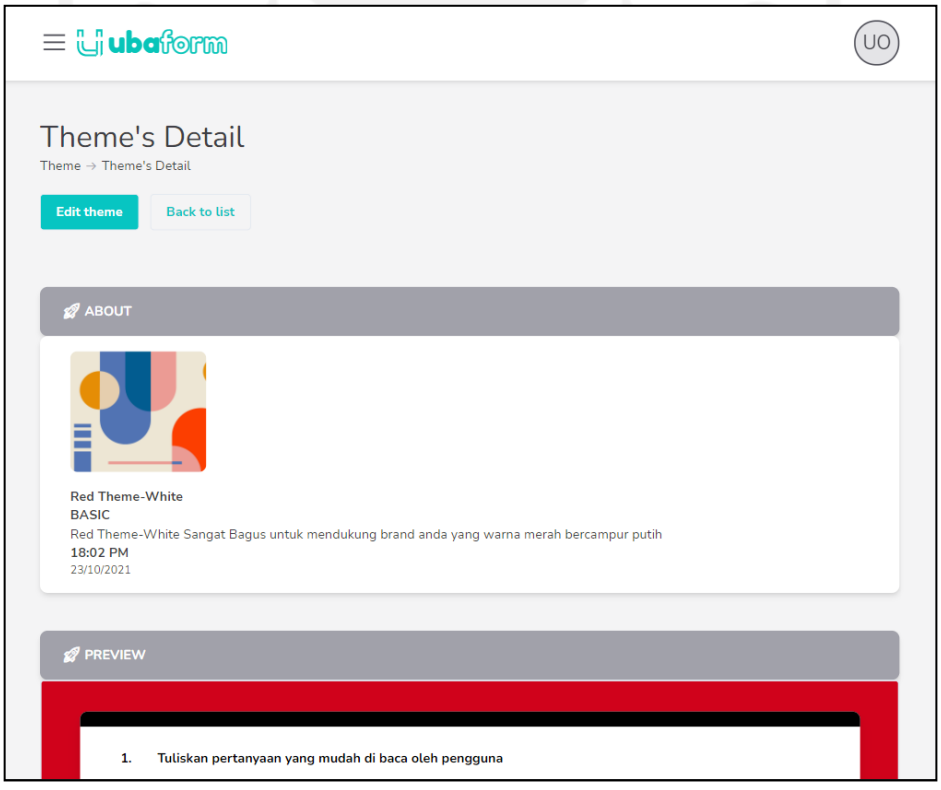
Gambar 4.23 Halaman *Theme management edit theme* admin Ubaform *(Mobile View)*

Halaman *theme management edit theme* admin Ubaform adalah halaman yang digunakan untuk melakukan pengeditan pada suatu tema. Secara umum ada dua komponen yang ditampilkan pada tampilan ini yaitu komponen editor dan preview temanya.

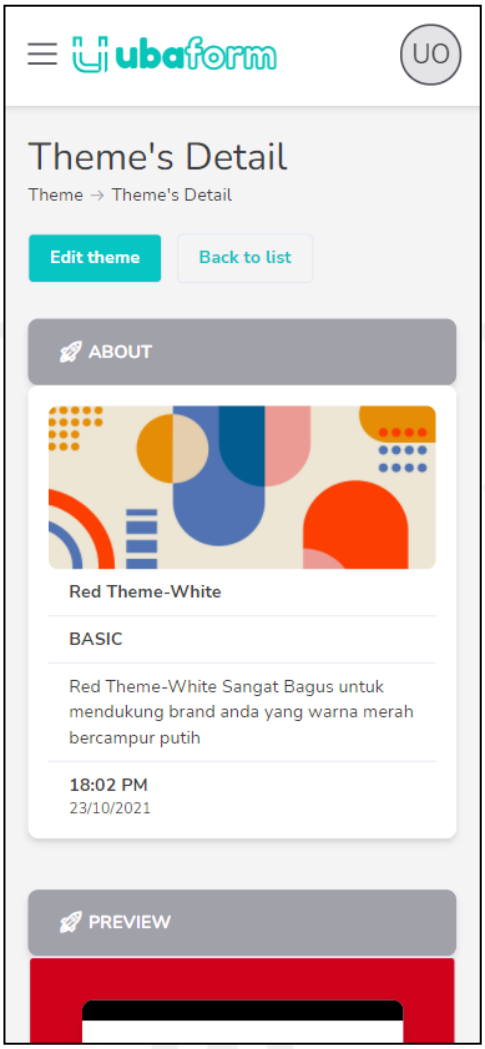
- Halaman *Theme Management (Detail Theme)*



Gambar 4.24 Halaman *Theme management detail theme* admin Ubaform *(Desktop View)*



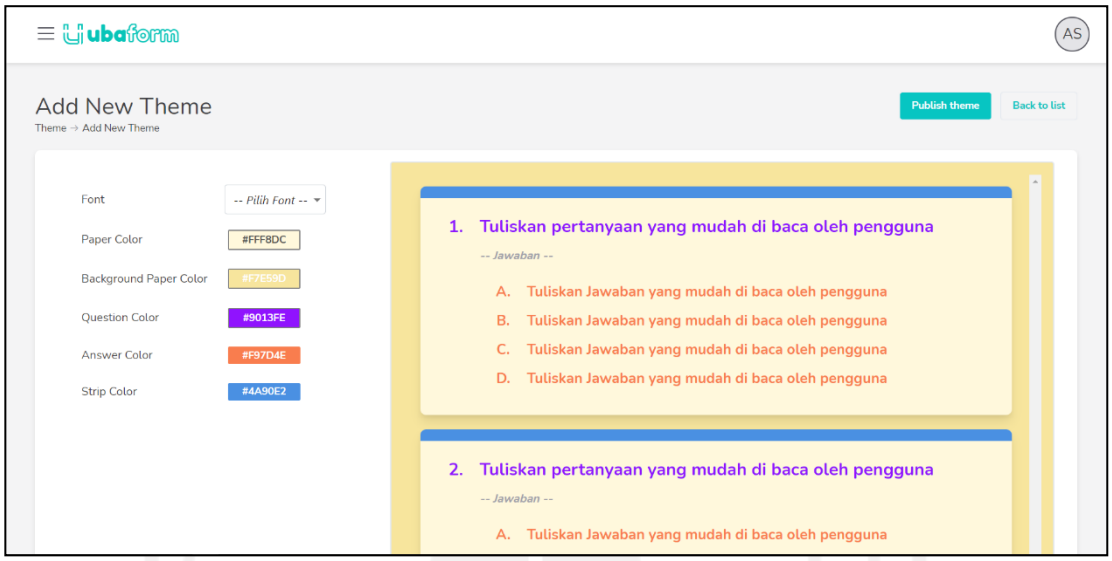
Gambar 4.25 Halaman *Theme management detail theme* admin Ubaform *(Tablet View)*



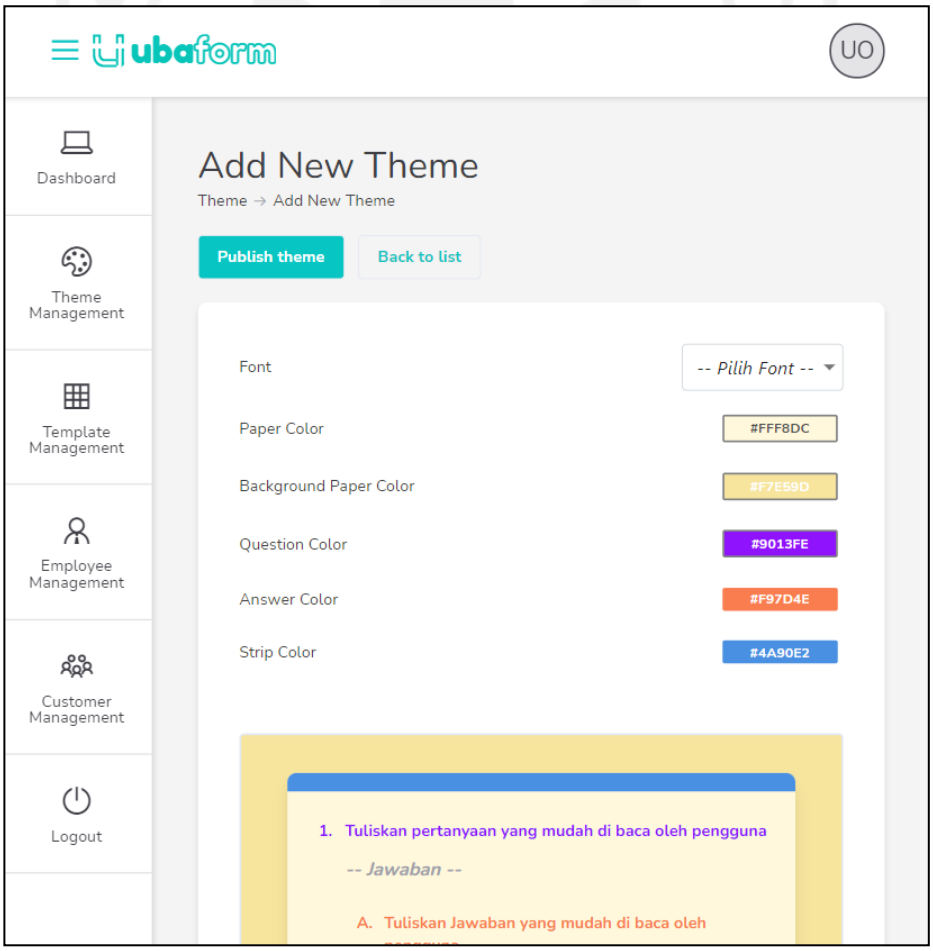
Gambar 4.26 Halaman *Theme management detail theme* admin Ubaform *(Mobile View)*

Halaman *theme management detail theme* admin Ubaform adalah halaman yang menampilkan detail tema secara keseluruhan dari atribut tema yang ada bedanya dengan list tema adalah pada halaman detail juga ditampilkan juga preview tema nya.

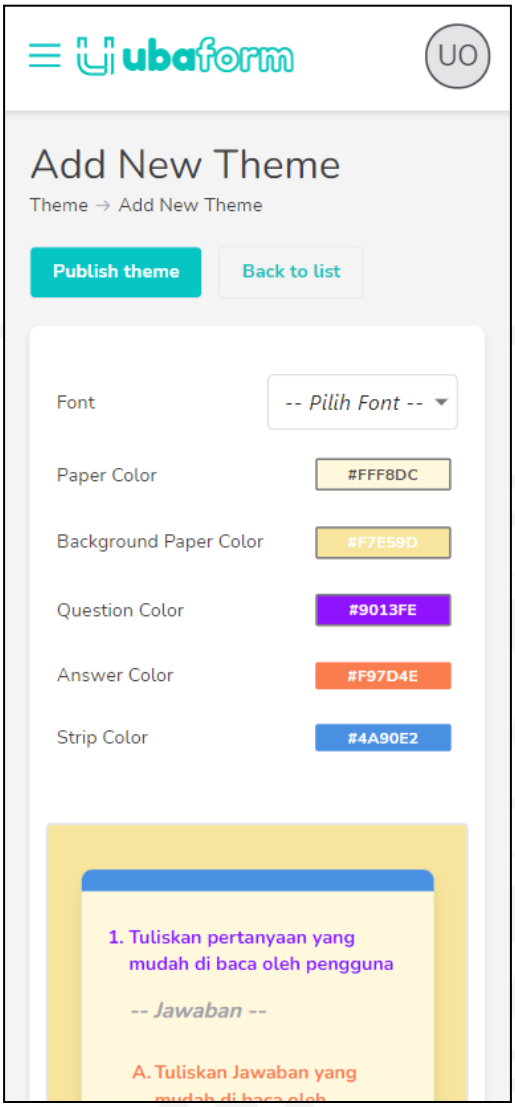
- Halaman *Theme Management (Create Theme)*



Gambar 4.27 Halaman *Theme management create theme* admin Ubaform *(Desktop View)*



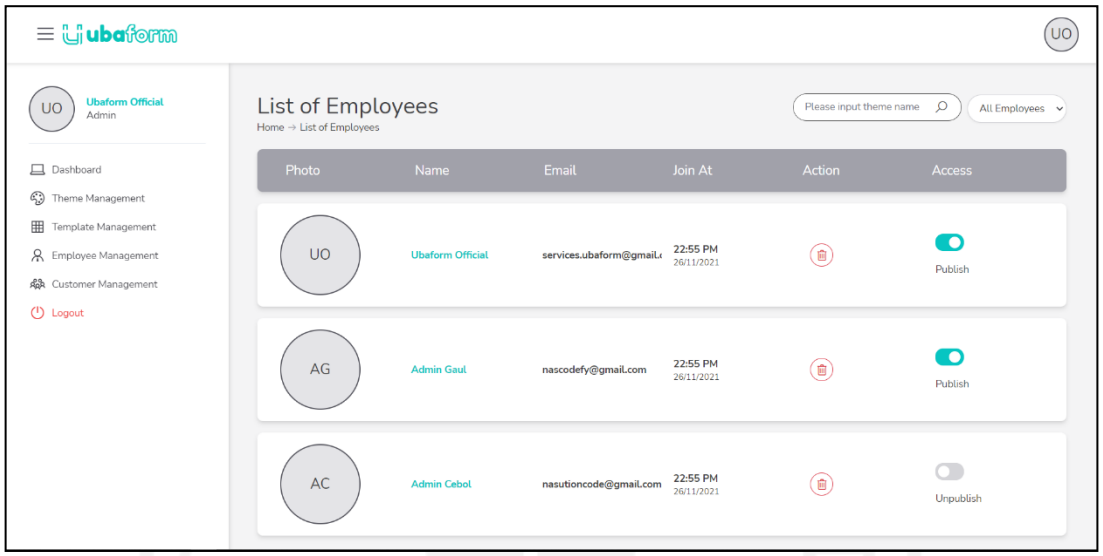
Gambar 4.28 Halaman *Theme management create theme* admin Ubaform *(Tablet View)*



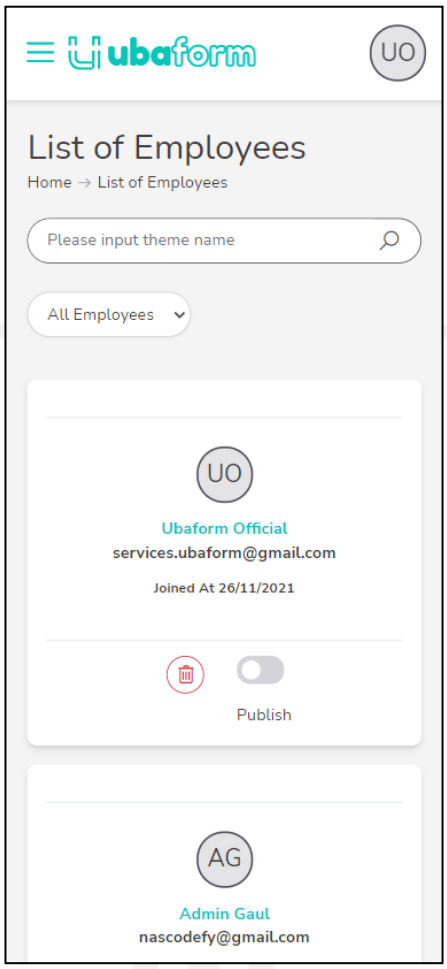
Gambar 4.29 Halaman *Theme management create theme* admin Ubaform *(Mobile View)*

Halaman *theme management create theme* admin Ubaform adalah halaman yang digunakan untuk membuat tema baru. Secara umum ada dua komponen yang ditampilkan pada tampilan ini yaitu komponen editor dan preview temanya dan ketika button publish diklik akan menampilkan *fields* yang harus diisi dengan. *Fields* yang diisi antara lain nama tema, deskripsi tema, tipe teme dan gambar tema.

- Halaman *Employee Management (List Employee)*



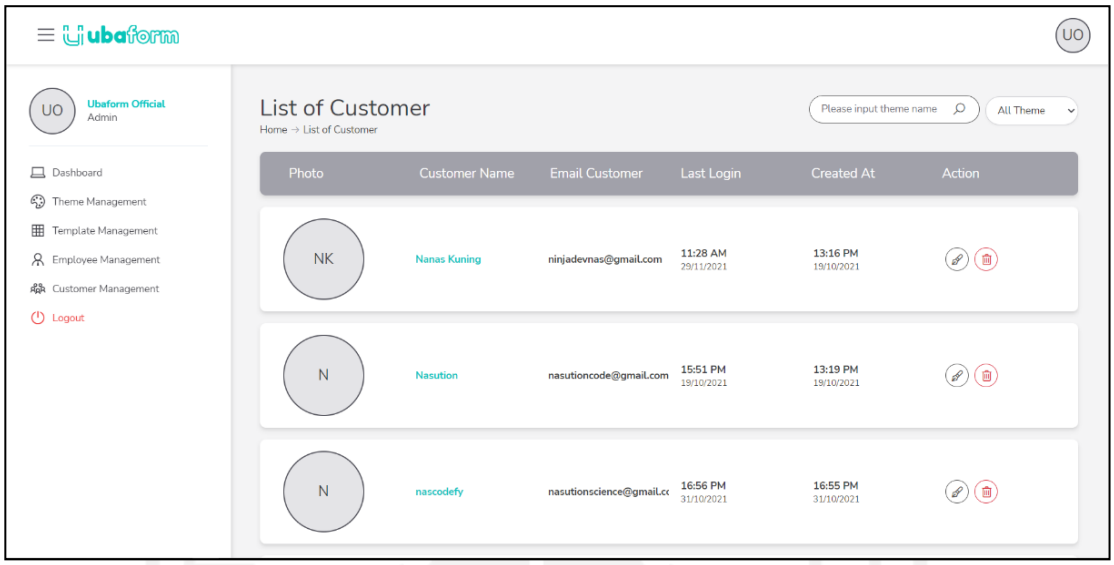
Gambar 4.30 Halaman *list employee* admin Ubaform *(Desktop View)*



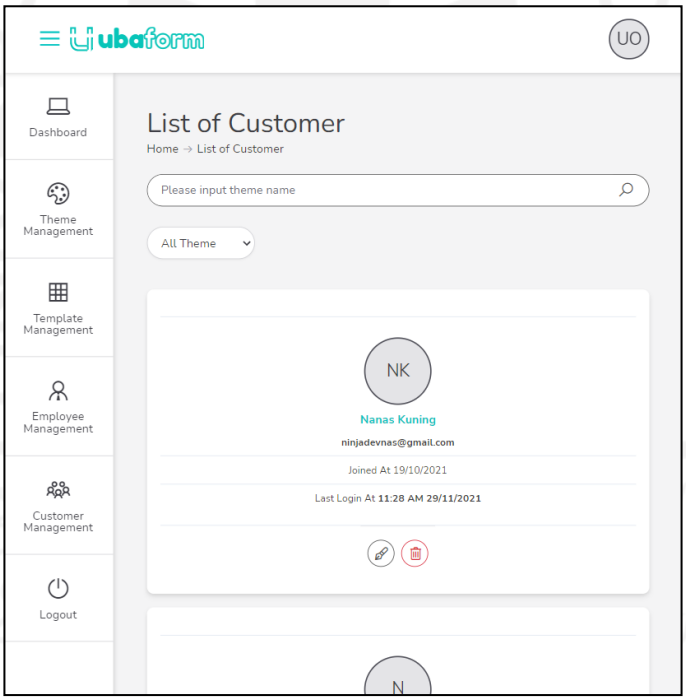
Gambar 4.31 Halaman *list employee* admin Ubaform *(Mobile View)*

Halaman *employee management (list employee)* admin Ubaform adalah halaman yang menampilkan daftar keseluruhan *employee* (Admin) yang telah dibuat. Atribut *employee* yang ditampilkan seperti nama, email, waktu bergabung dan *employee* aktif. Halaman ini memiliki fitur cari *employee*, filter *employee* dan aktivasi (publish) *employee.*

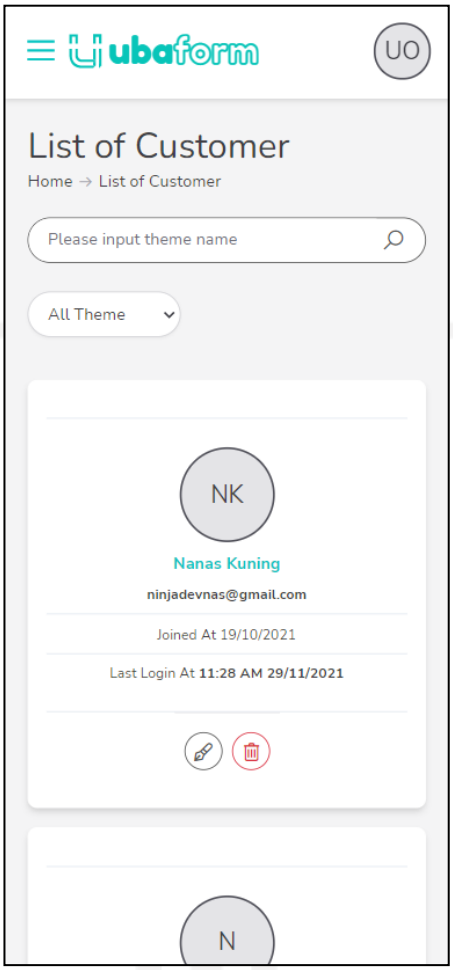
- Halaman *Employee Management (List Employee)*



Gambar 4.32 Halaman *list customer* admin Ubaform *(Desktop View)*



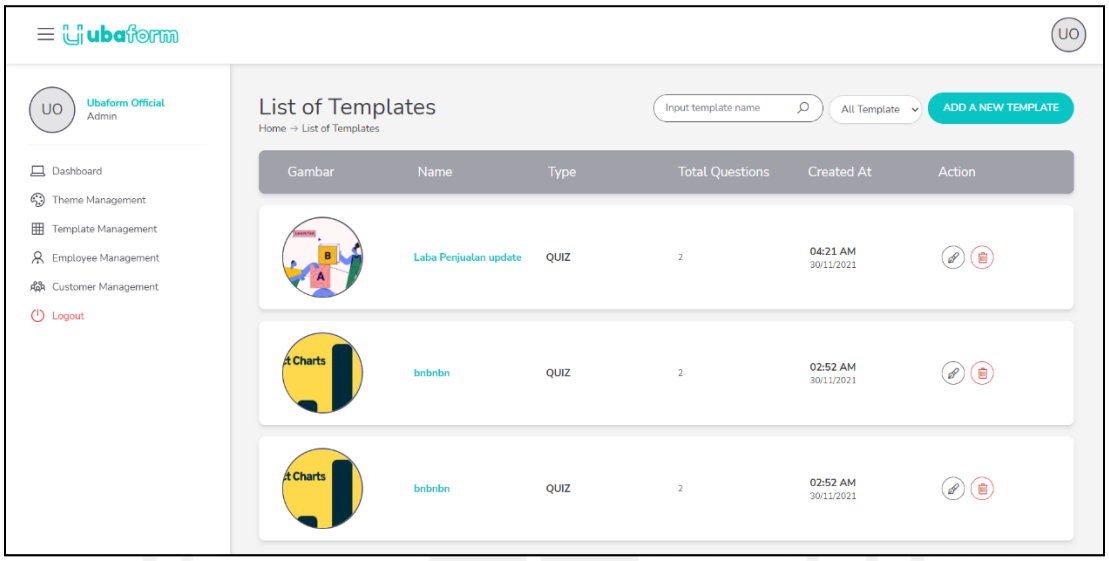
Gambar 4.33 Halaman *list customer* admin Ubaform *(Tablet View)*



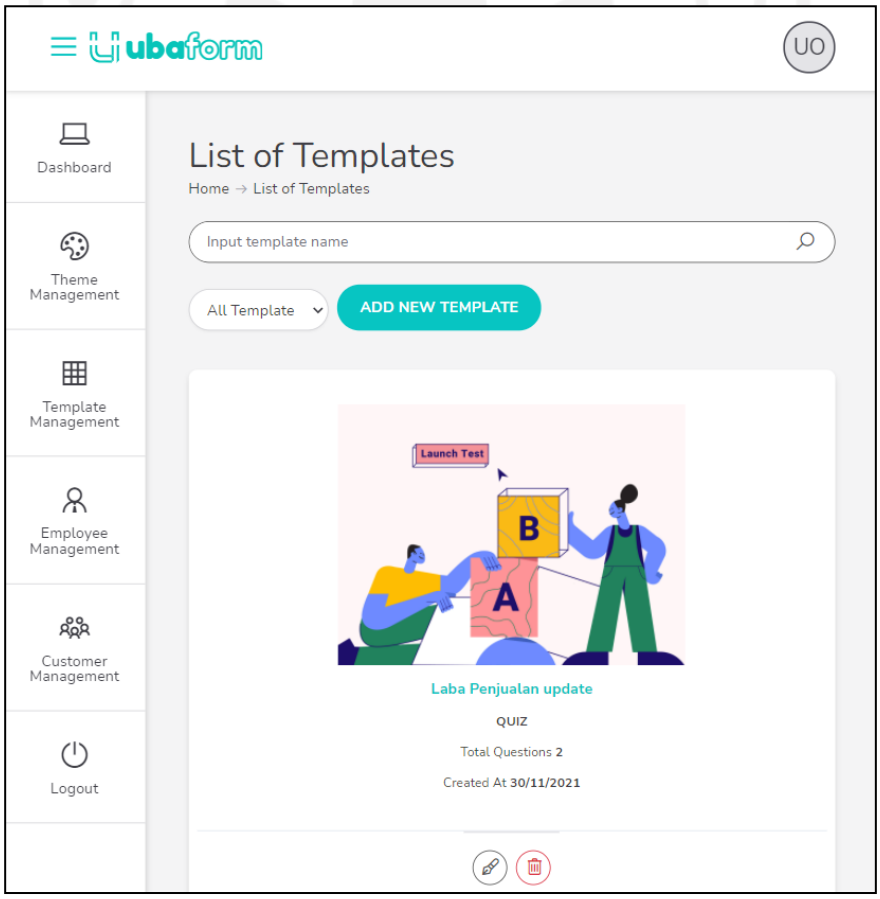
Gambar 4.34 Halaman *list customer* admin Ubaform *(Mobile View)*

Halaman *customer management (list customer)* admin Ubaform adalah halaman yang menampilkan daftar keseluruhan *customer* (Admin) yang telah dibuat. Atribut *customer* yang ditampilkan seperti nama, email, waktu bergabung dan terakhir login. Halaman ini memiliki fitur cari *employee*, filter *employee* dan hapus *customer*

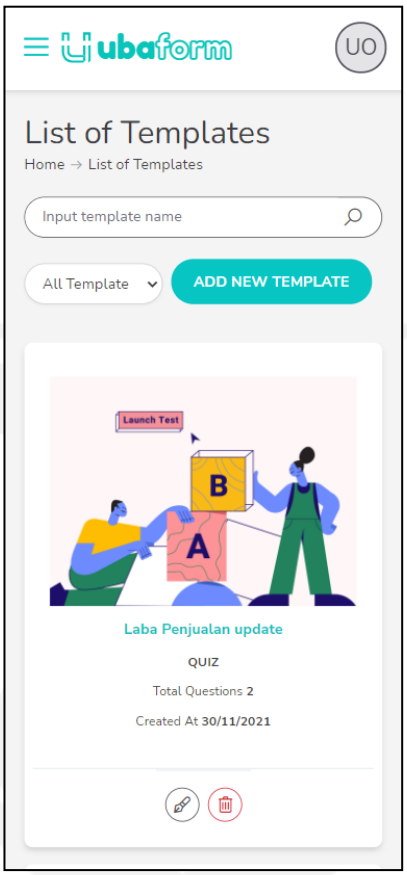
- Halaman *Template Management (List Template)*



Gambar 4.35 Halaman *list template* admin Ubaform *(Desktop View)*



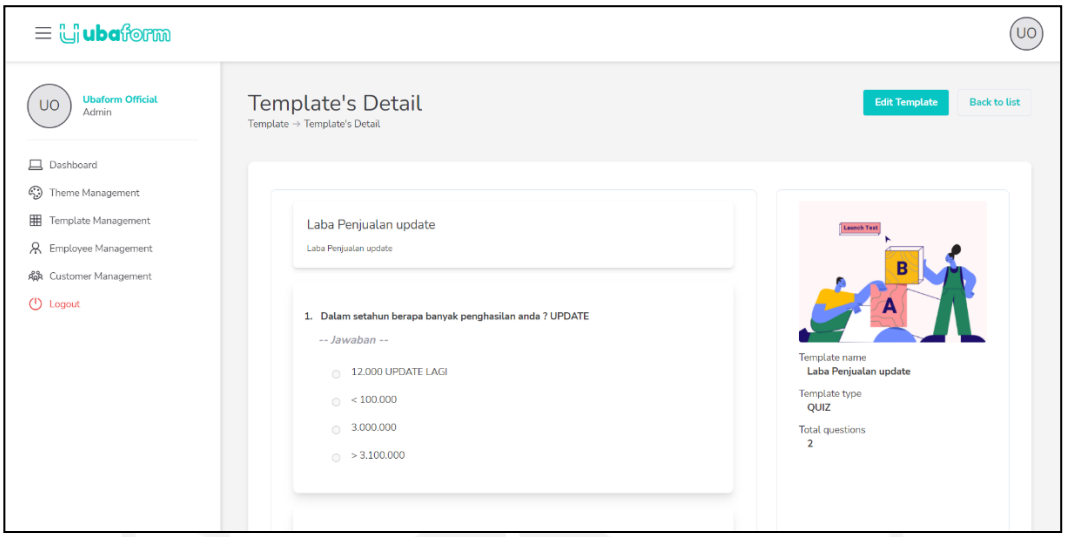
Gambar 4.36 Halaman *list template* admin Ubaform *(Tablet View)*



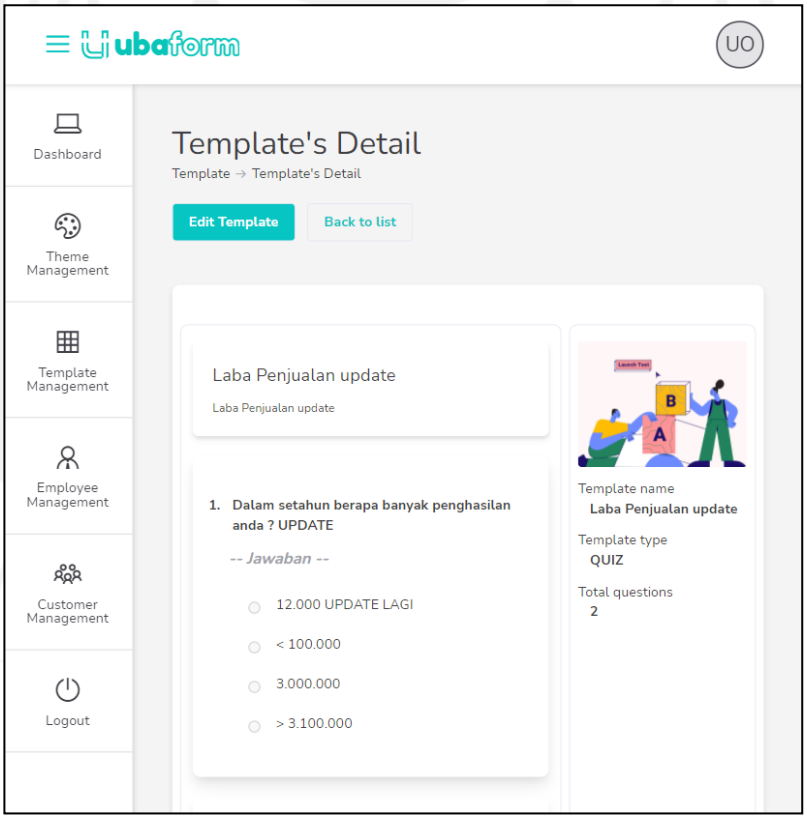
Gambar 4.37 Halaman *list template* admin Ubaform *(Mobile View)*

Halaman *template management (list template)* admin Ubaform adalah halaman yang menampilkan daftar keseluruhan *template* (Admin) yang telah dibuat. Atribut *template* yang ditampilkan seperti nama, tipe, waktu dibuat dan total *questions*. Halaman ini memiliki fitur cari *template*, filter *template*, *edit* dan hapus *template*

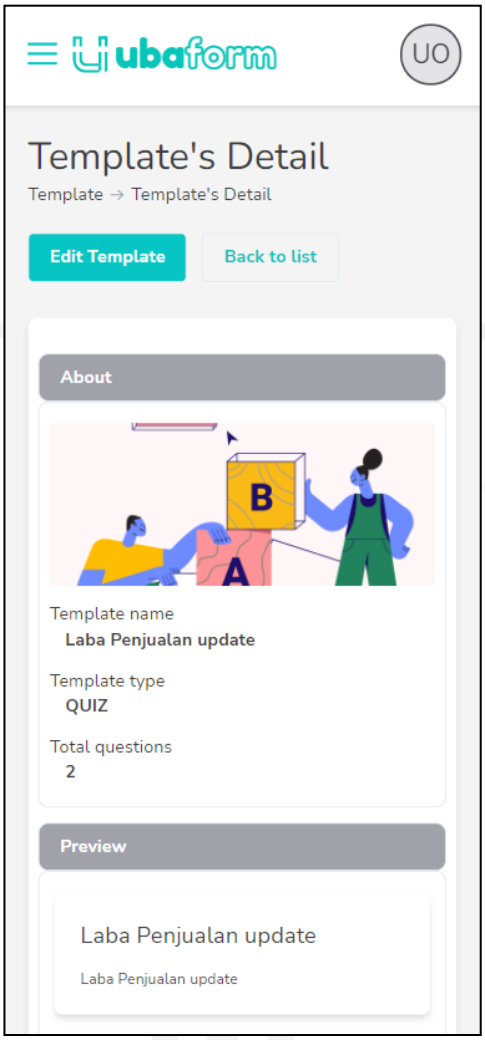
- Halaman *Template Management (Detail Template)*



Gambar 4.38 Halaman *detail template* admin Ubaform *(Desktop View)*



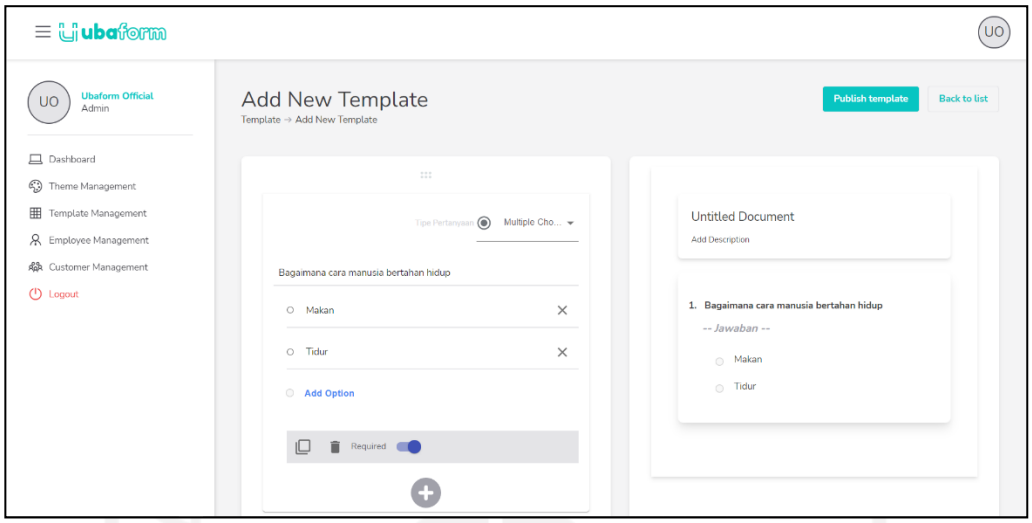
Gambar 4.39 Halaman *detail template* admin Ubaform *(Tablet View)*



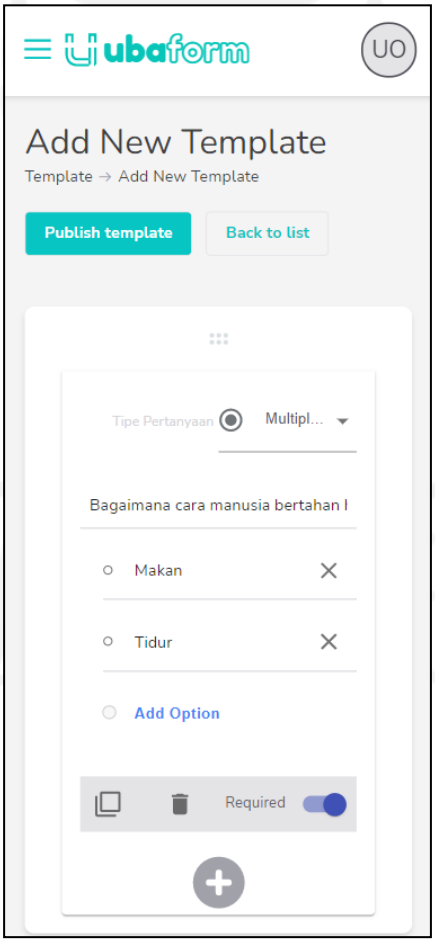
Gambar 4.40 Halaman *detail template* admin Ubaform *(Mobile View)*

Halaman *template management detail template* admin Ubaform adalah halaman yang menampilkan detail *template* secara keseluruhan dari atribut template yang ada bedanya dengan list tema adalah pada halaman detail juga ditampilkan juga preview *template* nya.

- Halaman *Template Management (Create Template)*



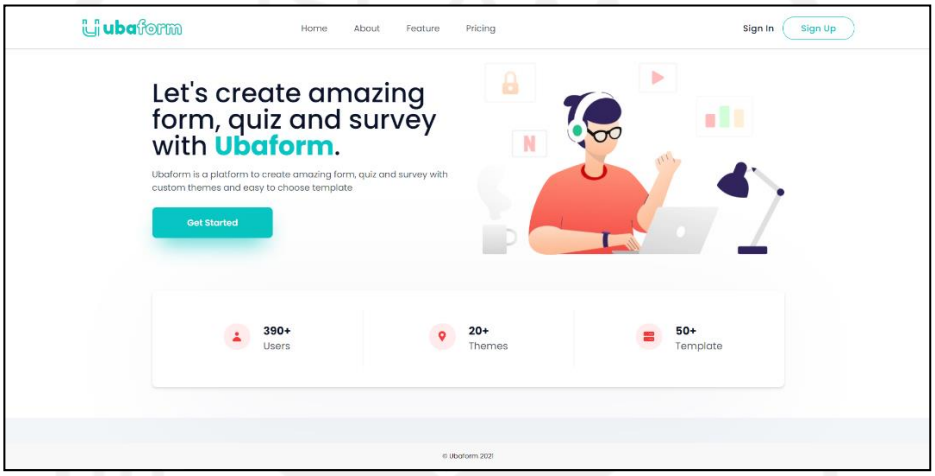
Gambar 4.41 Halaman *create template* admin Ubaform *(Desktop View)*



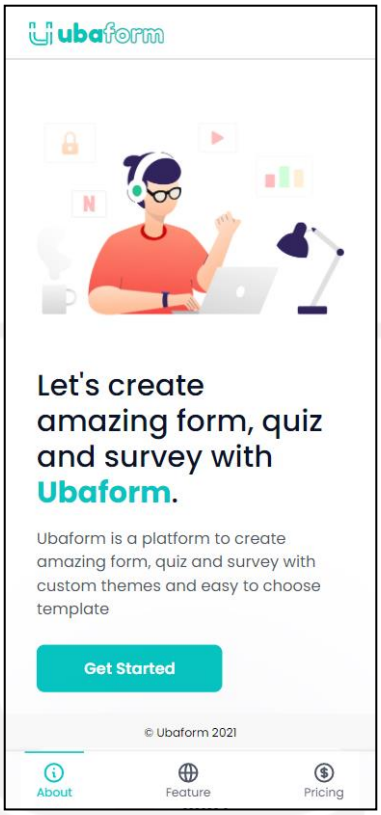
Gambar 4.42 Halaman *create template* admin Ubaform *(Mobile View)*

Halaman *template management create template* admin Ubaform adalah halaman yang digunakan untuk membuat *template* baru. Secara umum ada dua komponen yang ditampilkan pada tampilan ini yaitu komponen editor dan preview temanya dan ketika button publish diklik akan menampilkan *fields* yang harus diisi dengan. *Fields* yang diisi antara lain nama *template*, deskripsi *template*, tipe *template* dan gambar *template*

- Halaman *Landing Customer*



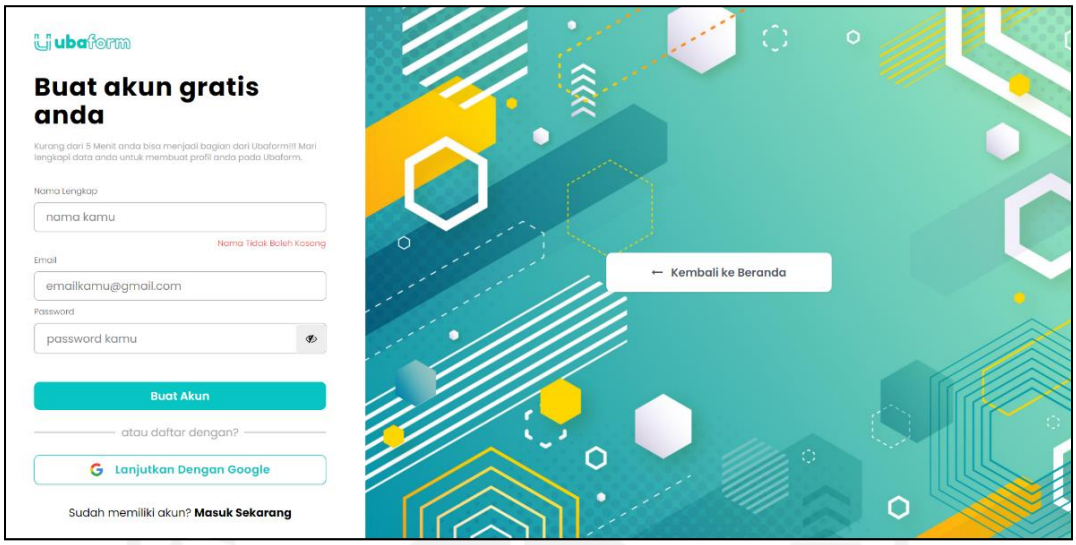
Gambar 4.43 Halaman *Landing customer* Ubaform *(Desktop View)*



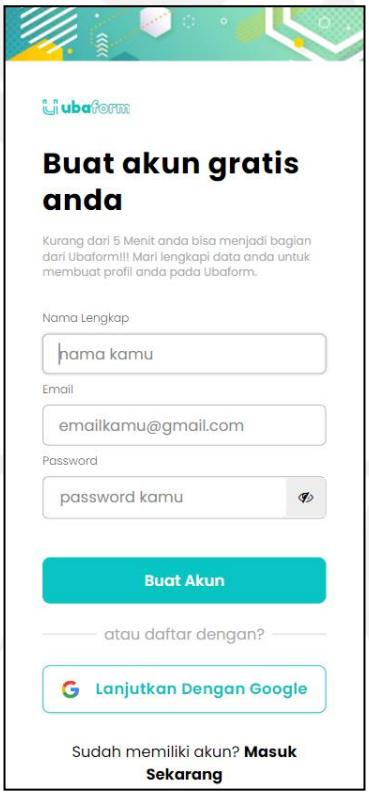
Gambar 4.44 Halaman *Landing customer* Ubaform *(Mobile View)*

Halaman *landing* Ubaform adalah halaman statik bersifat public yang dapat diakses oleh semua pengguna internet baik yang sudah melakukan *login* atau belum. Halaman ini berisi informasi dasar tentang layanan pada *startup* Ubaform, yaitu seperti ap aitu Ubaform, keunggulan dan harga paket langganan.

- Halaman *Register Customer*



Gambar 4.45 Halaman *Register customer* Ubaform *(Desktop View)*



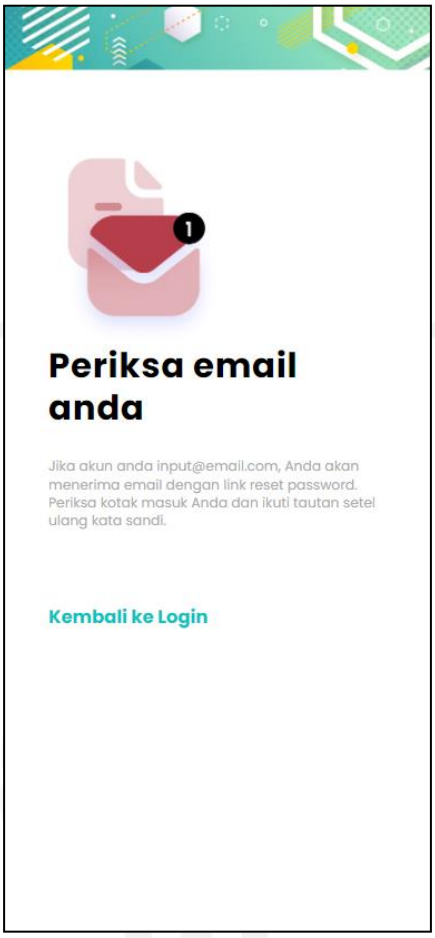
Gambar 4.46 Halaman *Register customer* Ubaform *(Mobile View)*

Halaman *register* pada gambar 4.45 dan 4.46 adalah halaman yang digunakan oleh pengguna untuk melakukan pendaftaran pada web *customer* Ubaform. Pengguna perlu melakukan pendaftaran untuk bisa sebelum melakukan *login* untuk dapat menggunakan

layanan. Pendaftaran dapat dilakukan dengan menggunakan email dan oAuth google *authentication*. Pengguna dengan pendaftaran menggunakan email akan diarahkan untuk melakukan pendaftaran lanjutan dengan mengakses URL yang dikirimkan ke email yang digunakan untuk mendaftar. Hal ini akan membuat filter pendaftaran hanya bisa dilakukan dengan email yang valid dan menghindari spam email. Adapaun pendaftaran menggunakan oAuth google *authentication* akan langsung dapat *login* dan diarahkan ke halaman dashboard.

- Halaman Cek *Email Register Customer*

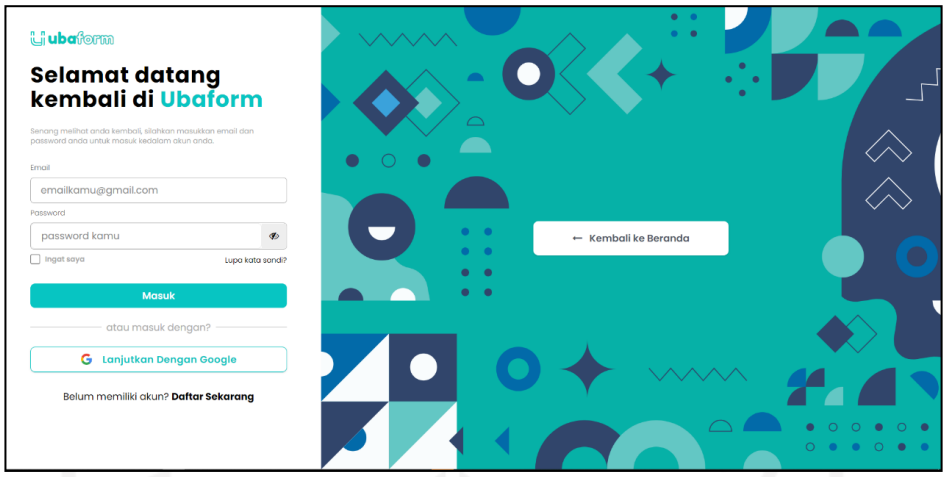
Gambar 4.47 Halaman *Cek Email Register customer* Ubaform *(Desktop View)*



Gambar 4.48 Halaman *Cek Email Register customer* Ubaform *(Mobile View)*

Halaman *cek email register* pada gambar 4.47 dan 4.48 adalah halaman yang digunakan untuk memberitahu pengguna setelah melakukan pendaftaran dengan email untuk melakukan pengecekan email dan melanjutkan proses pendaftaran.

- Halaman *Login Customer*

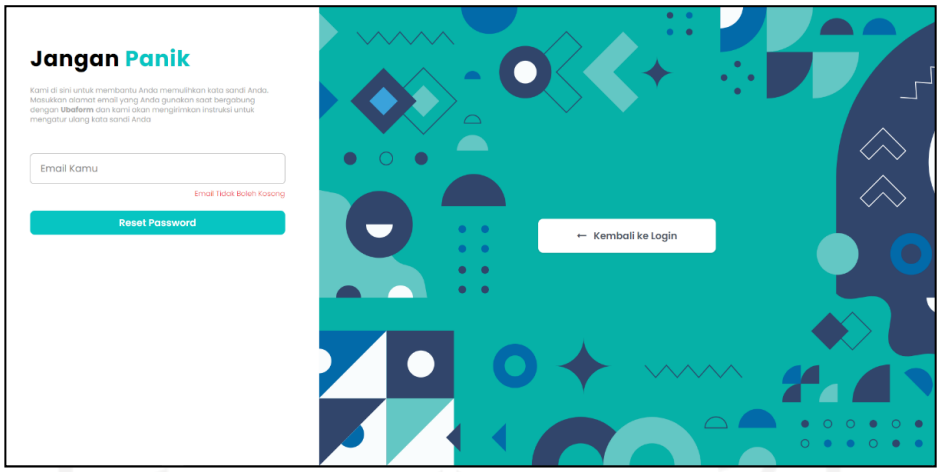


Gambar 4.49 Halaman *Login customer* Ubaform *(Desktop View)*

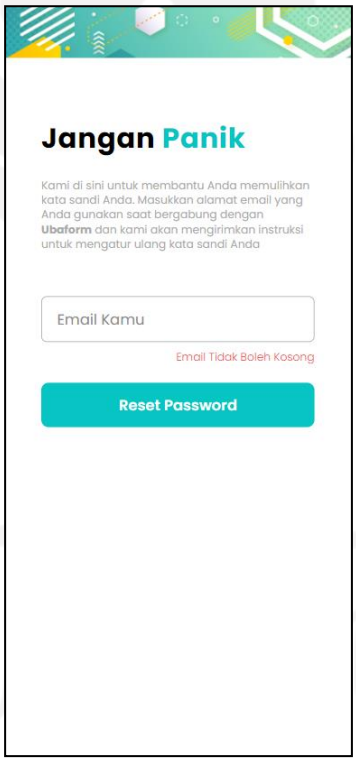
Gambar 4.50 Halaman *Login customer* Ubaform *(Desktop View)*

Halaman *login customer* pada gambar 4.49 dan 4.50 adalah halaman yang digunakan untuk melakukan *login* untuk bisa mengakses dan menggunakan layanan *startup* Ubaform. Pada halaman ini terdapat beberapa komponen utama seperti *field*, *button* google *login* dan *link forgot password*.

- Halaman *Forgot Password Customer*



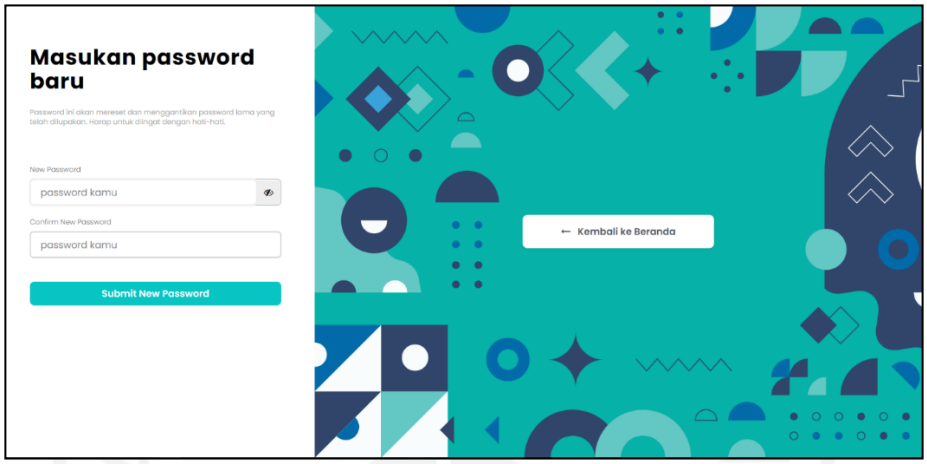
Gambar 4.51 Halaman *forgot password customer* Ubaform *(Desktop View)*



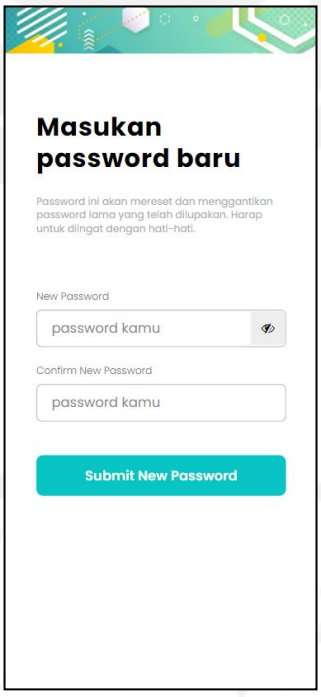
Gambar 4.52 Halaman *forgot password customer* Ubaform *(Mobile View)*

Halaman *forgot password* pada gambar 4.51 dan 4.52 adalah halaman yang digunakan ketika pengguna lupa *password.* Pada halaman ini pengguna akan diminta mengisi email yang digunakan sebagai akun terdaftar yang *password* nya telah dilupakan. Setelah itu pengguna akan diarahkan untuk mengecek email yang berisi URL yang mengarahkan ke halaman *reset password.*

- Halaman *Reset Password Customer*



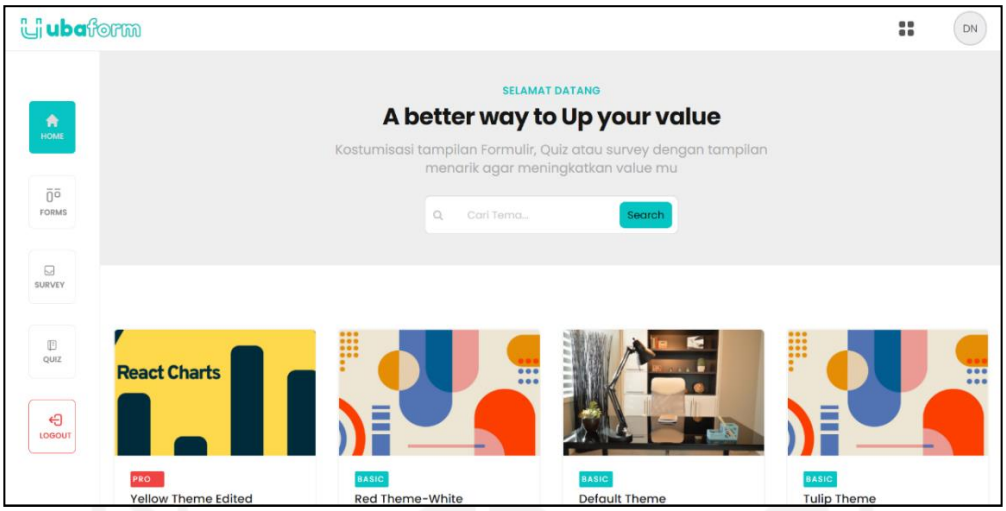
Gambar 4.53 Halaman *Reset Password customer* Ubaform *(Destop View)*



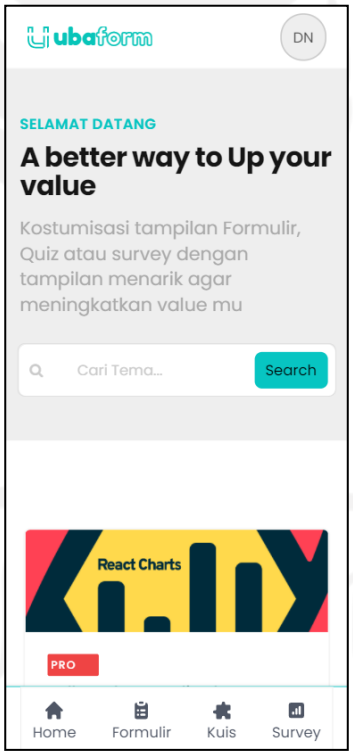
Gambar 4.54 Halaman *Reset Password customer* Ubaform *(Mobile View)*

Halaman *reset password* pada gambar 4.53 dan 4.53 adalah halaman yang diakses melalui URL yang terkirim ke email pengguna sebelumnya. Pada halaman ini pengguna akan diminta untuk memasukan password baru.

- Halaman *Home Dashboard Customer*



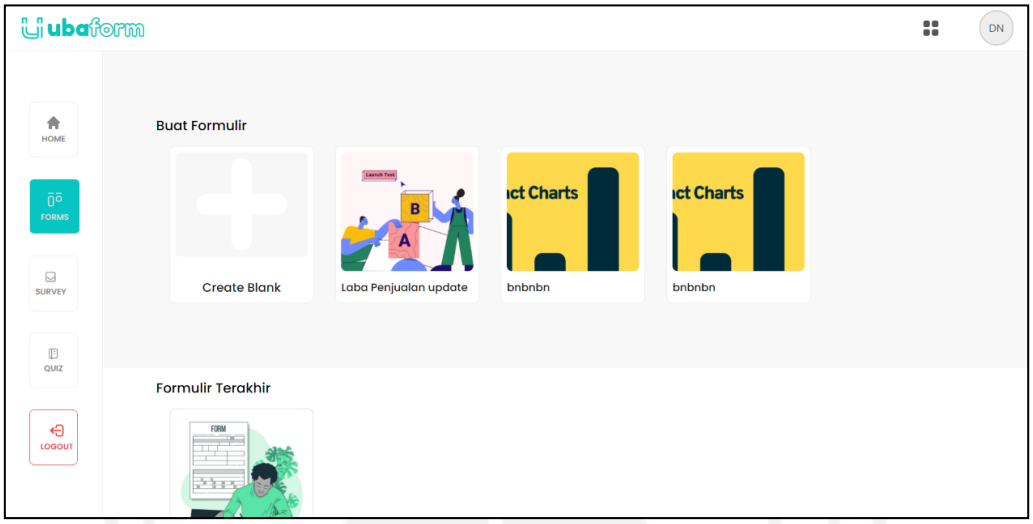
Gambar 4.55 Halaman *Home Dashboard customer* Ubaform *(Desktop View)*



Gambar 4.56 Halaman *Home Dashboard customer* Ubaform *(Mobile View)*

Halaman *home dashboard* pada gambar 4.55 dan 4.56 adalah halaman yang dapat diakses apabila pengguna telah melakukan *login* sebelumnya. Pada halaman ini pengguna akan ditampilkan daftar tema yang ada secara keseluruhan baik tipe tema yang dapat digunakan setelah berlangganan maupun tema yang bersifat *free.*

- Halaman *Form Dashboard Customer*



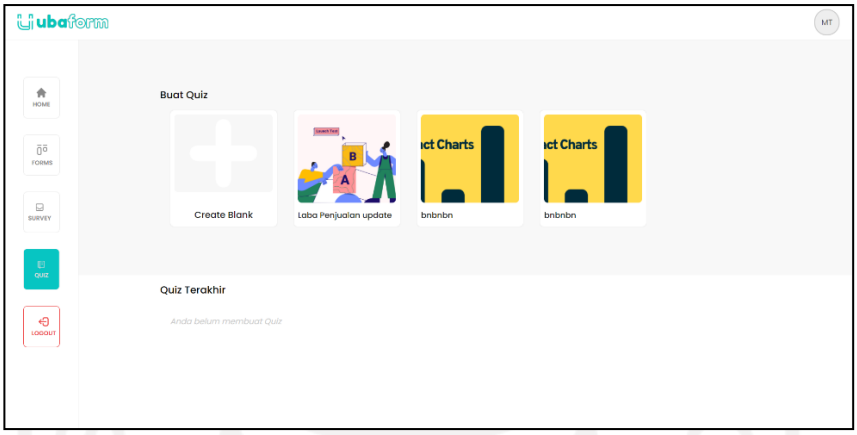
Gambar 4.57 Halaman *Form Dashboard customer* Ubaform *(Desktop View)*

Gambar 4.58 Halaman *Form Dashboard customer* Ubaform *(Mobile View)*

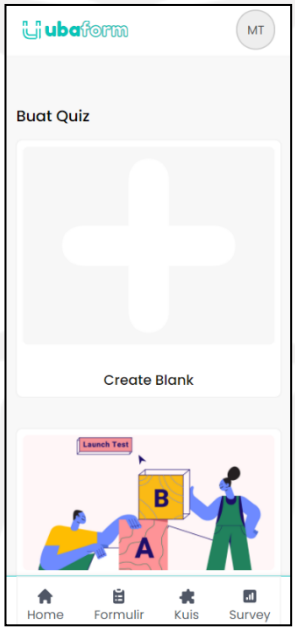
Halaman *form dashboard* pada gambar 4.57 dan 4.58 adalah halaman dashboard yang digunakan untuk membuat projek formulir dan juga melihat daftar formulir yang telah dibuat sebelumnya. Pembuatan formulir dapat dilakukan dengan memilih pembuatan dengan *blank* projek atau dengan menggunaan *template.* Pembuatan dengan *blank* projek artinya pengguna

akan membuat dari sketsa kosong dari awal sedangkan pembuatan dengan menggunakan *template* hanya tinggal mengedit dan menyesuaikan dengan kebutuhan pengguna. Jumlah projek yang dapat dibuat dibatasi berdasarkan tipe *subscription* dengan batasan *credit* tertentu.

- Halaman *Quiz Dashboard Customer*



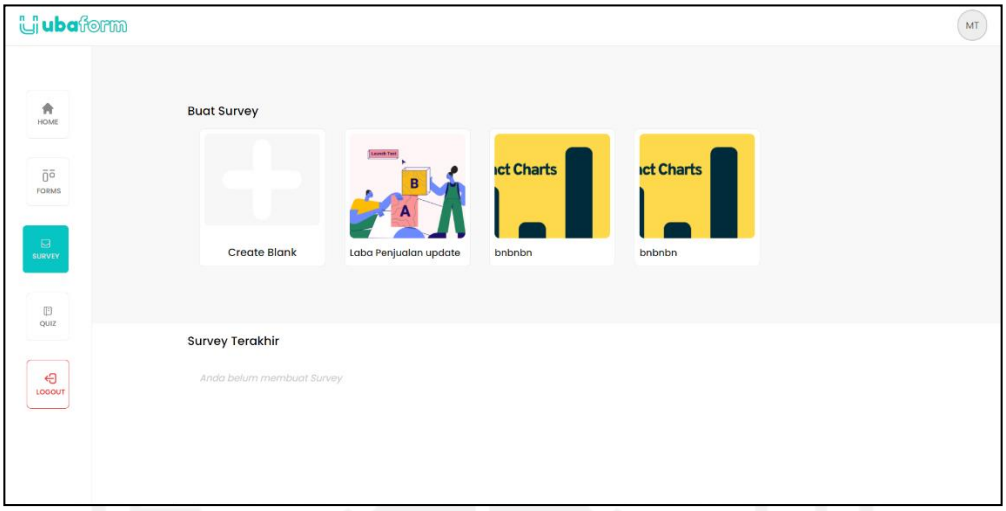
Gambar 4.59 Halaman *Quiz Dashboard customer* Ubaform *(Desktop View)*



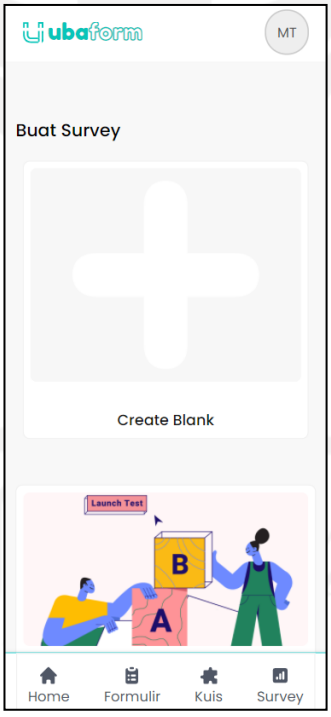
Gambar 4.60 Halaman *Quiz Dashboard customer* Ubaform *(Mobile View)*

Halaman *quiz dashboard* pada gambar 4.59 dan 4.60 adalah halaman dashboard yang digunakan untuk membuat projek kuis dan juga melihat daftar kuis yang telah dibuat sebelumnya. Pembuatan kuis dapat dilakukan dengan memilih pembuatan dengan *blank* projek atau dengan menggunaan *template* seperti pada pembuatan projek formulir.

- Halaman *Survey Dashboard Customer*



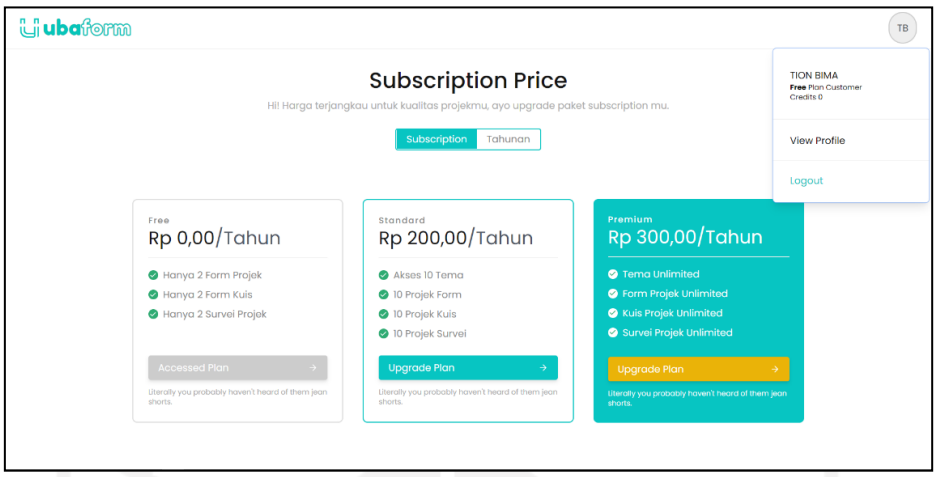
Gambar 4.61 Halaman *Survey Dashboard customer* Ubaform *(Desktop View)*



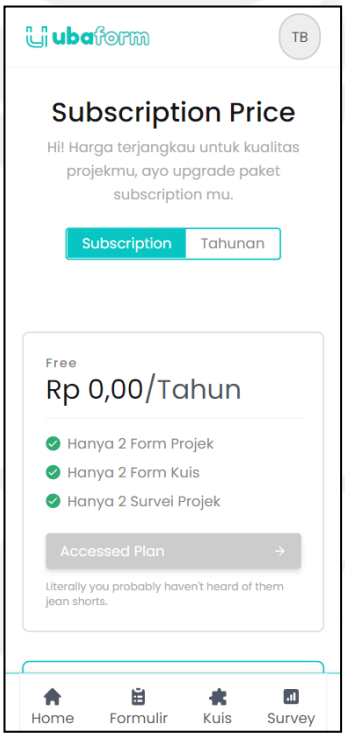
Gambar 4.62 Halaman *Survey Dashboard customer* Ubaform *(Mobile View)*

Halaman *survey dashboard* pada gambar 4.61 dan 4.62 adalah halaman dashboard yang digunakan untuk membuat projek survei dan juga melihat daftar survei yang telah dibuat sebelumnya. Pembuatan kuis dapat dilakukan dengan memilih pembuatan dengan *blank* projek atau dengan menggunaan *template* seperti pada pembuatan projek formulir.

- Halaman *Payment Customer*

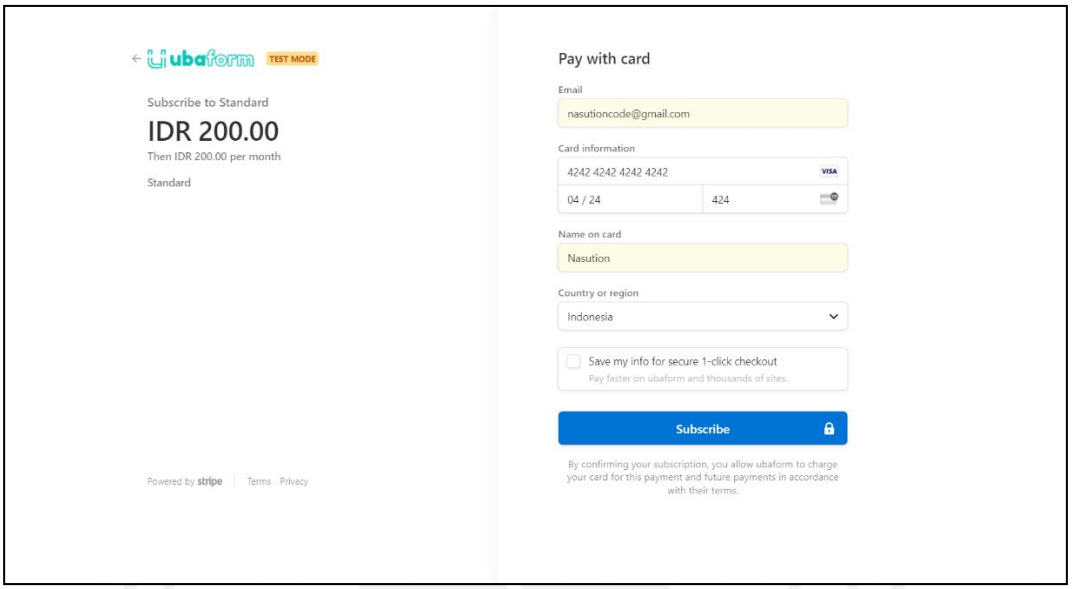


Gambar 4.63 Halaman *Payment customer* Ubaform *(Desktop View)*



Gambar 4.64 Halaman *Payment customer* Ubaform *(Mobile View)*

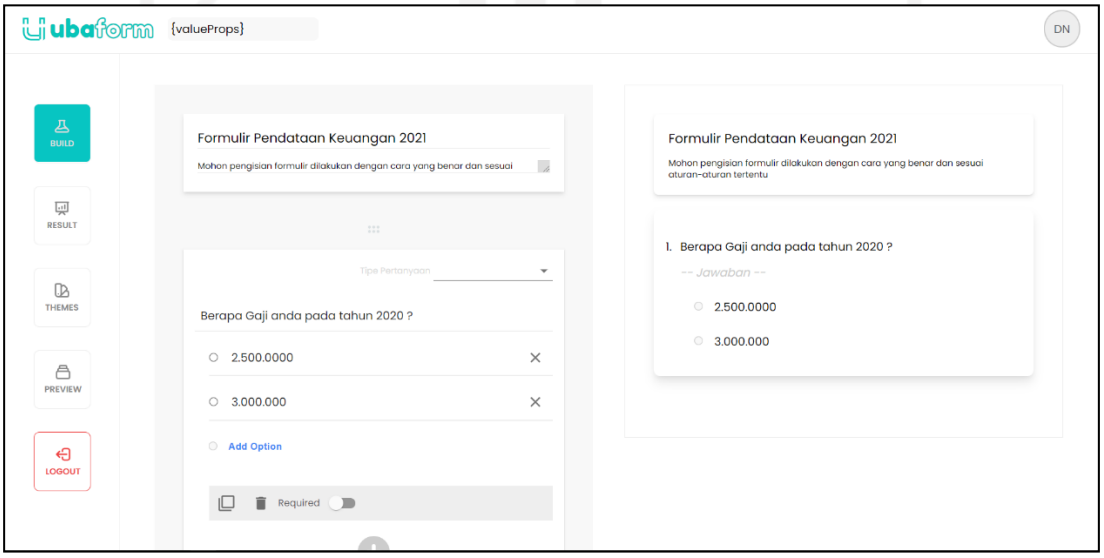
Halaman *payment* pada gambar 4.63 dan 4.64 adalah halaman yang digunakan untuk melihat daftar paket *subscription.* Halaman ini dikakses ketika pengguna akan melakukan pembelian paket *subscription.* Pada daftar *subscription* terdapat komponen harga dan apa saja yang akan didapat ketika membeli paket.



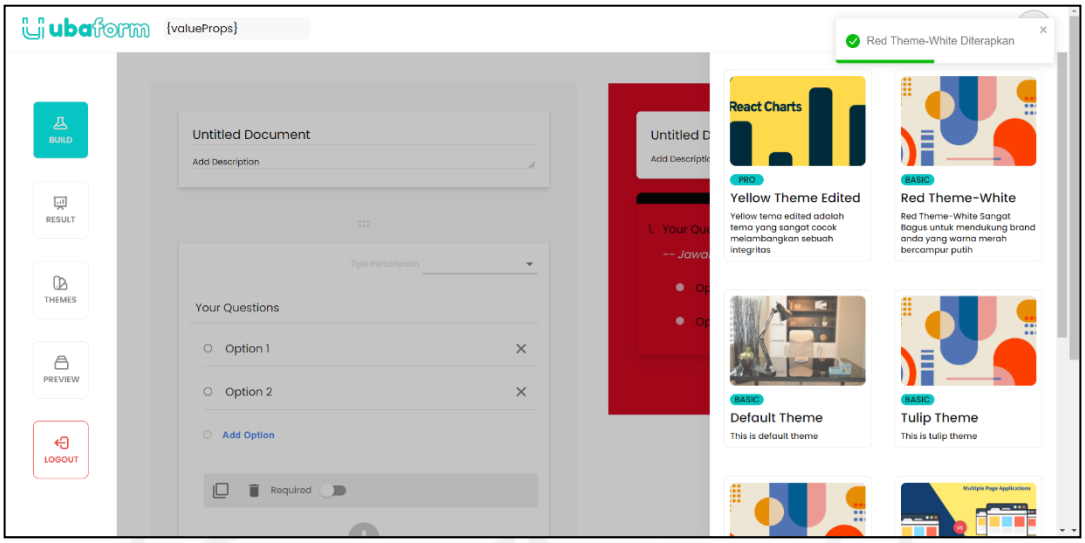
Gambar 4.65 Halaman *checkout customer* Ubaform

Halaman *checkout* pada gambar 4.65 adalah halaman yang akan muncul ketika pengguna melakukan pembelian dan memilih paket *subscription.* Pada halaman ini akan diminta mengisi data pembelian seperti nomor kartu dan lain sebagainya.

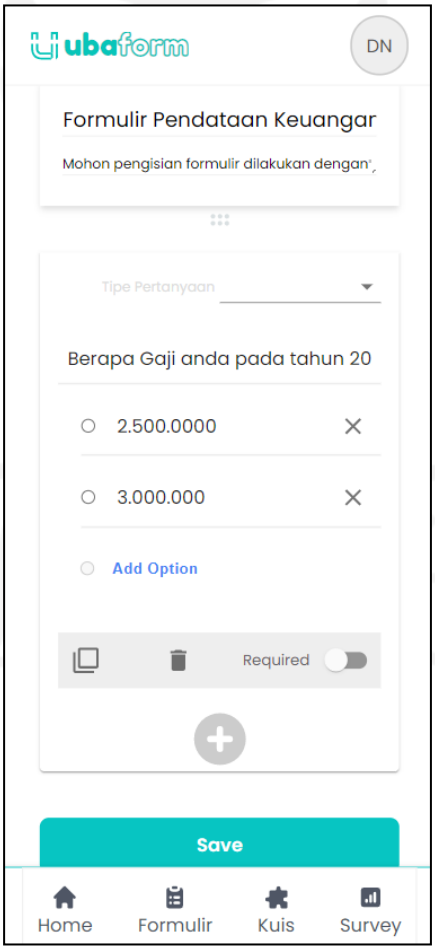
- Halaman *Builder Customer*



Gambar 4.66 Halaman *builder customer* Ubaform *(Desktop View)*



Gambar 4.67 Halaman penggunaan tema *builder customer* Ubaform *(Desktop View)*



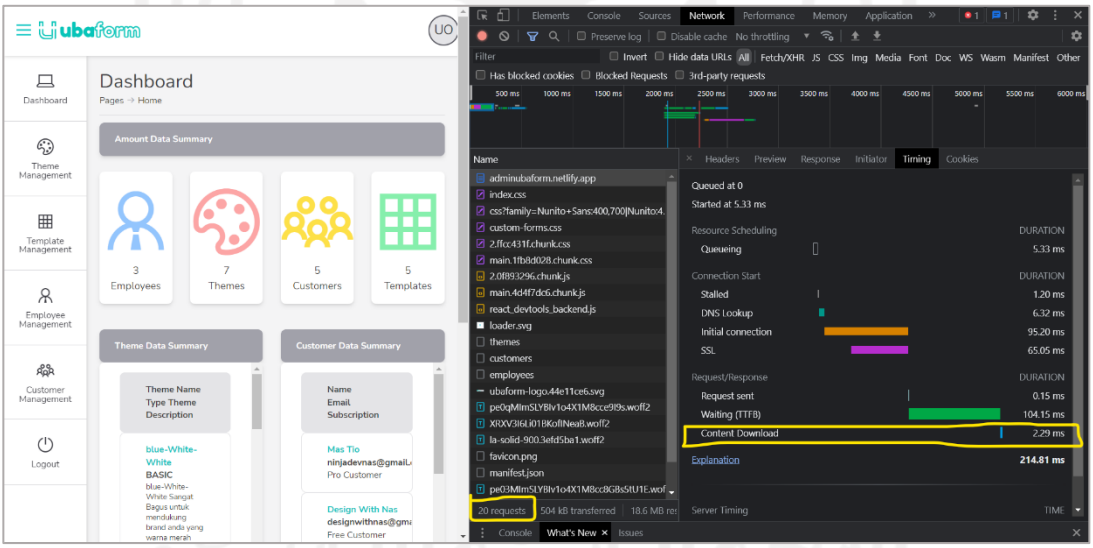
Gambar 4.68 Halaman *builder customer* Ubaform *(Mobile View)*

Halaman *builder* pada gambar 4.66 sampai 4.68 adalah halaman yang digunakan untuk membuat projek formulir, kuis atau survei. Pada halaman ini terdapat beberapa komponen seperti komponen pengeditan, *real time preview* dan lain sebagainya.

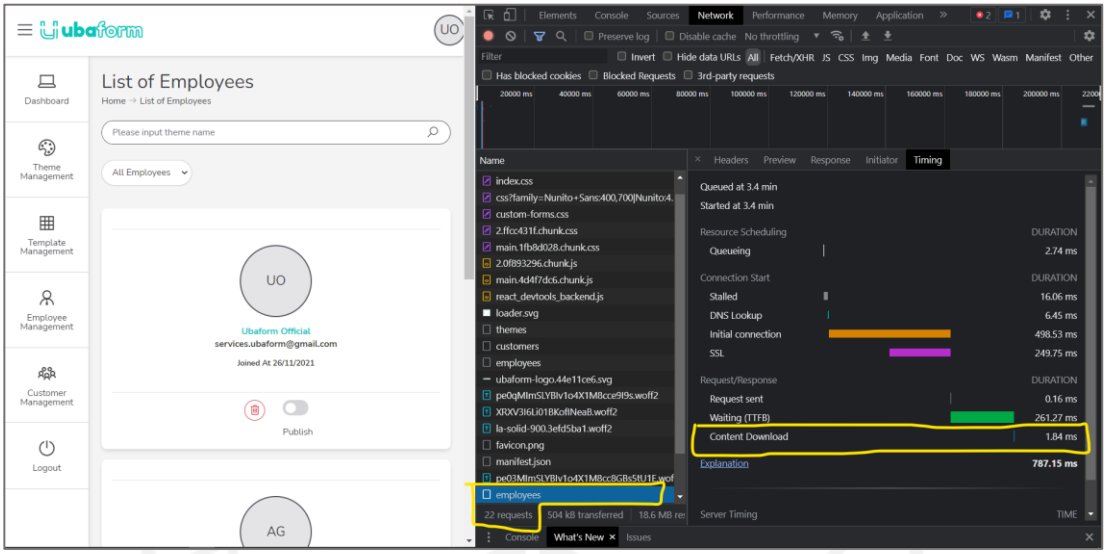
4.2.2 Pembahasan

Untuk pemahaman performa React JS penulis melakukan pengecekan pada *developer tools* dari browser pada *tab* performa dan *network*. Pengecekan tersebut dilakukan pada transaksi CRUD data pada web Ubaform dan SPA perpindahan antara halaman berdasarkan *request* dan *response* yang dilakukan ke server. Adapun hasil performa tersebut dapat dilihat sebagai berikut:

- Performa web Ubaform berbasis SPA dengan React JS



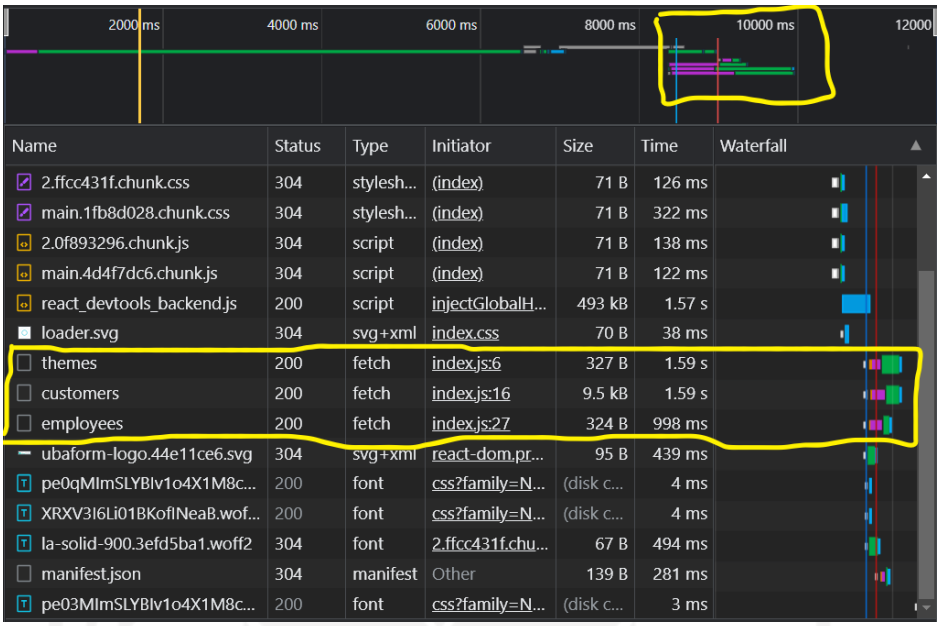
Gambar 4.69 Performa React JS Pertama kali dimuat



Gambar 4.70 Perform React JS Perpindahan Halaman Dengan Basis SPA

Dapat dilihat perbedaan pada gambar 4.69 dengan 4.70, untuk lebih terlihat perbedaan tersebut penulis tandai dengan garis warna kuning seperti pada kedua gambar tersebut. Pada gambar 4.10 adalah pertama kali web dimuat dengan keseluruhan *resource* dari penyusun web tersebut. Hal ini menghasilkan 20 *request* yang dilakukan ke server untuk mendapatkan *resource* serta data yang diinginkan tersebut dan waktu yang dibutuhkan untuk memuatnya adalah dapat dilihat pada *Content Download* yang ada pada *Tab timing* broswer *tool* pada gambar tersebut yaitu 2.29 ms. Hal ini masih sangat cepat mengingat pemuatan halaman yang terdiri dari 20 *request*. Kemudian jika dibandingkan dengan gambar 4.70 yang mana gambar tersebut adalah perpindahan antara halaman *Home* ke halaman *Employee*. Pada perpindahan halaman yang berbasis SPA dimana tidak ada pemuatan ulang secara keseluruhan yang dilakukan melainkan hanya pemuatan data yang dibutuhkan saja. Hal ini dapat terlihat pada gambar 4.70 yang juga ditandai dengan garis kuning. Perbedaan dapat terlihat ketika perpindahan halaman terjadi program akan mengambil data yang dibutuhkan pada server dalam konteks ini data yang diambil adalah data halaman yang menjadi objek perpindahan yaitu halaman *Employee.* Sehingga *request* data ke server hanya 2, hal ini dapat dilihat pada penambahan dari 20 *request* menjadi 22 *request* ke server. Pada SPA hanya 2 *request* tersebut yang dimuat ulang sedangkan 20 lainya sudah dimuat ketika pertama kali mengunjungi web Ubaform. Hal ini tentunya berdampak pada kecepatan yang mampu meningkatkan performa web Ubaform yang awalnya untuk memuat keseluruhan halaman dengan 20 *request* membutuhkan waktu 2.2 ms dengan SPA menjadi 1.84 ms *content download*.

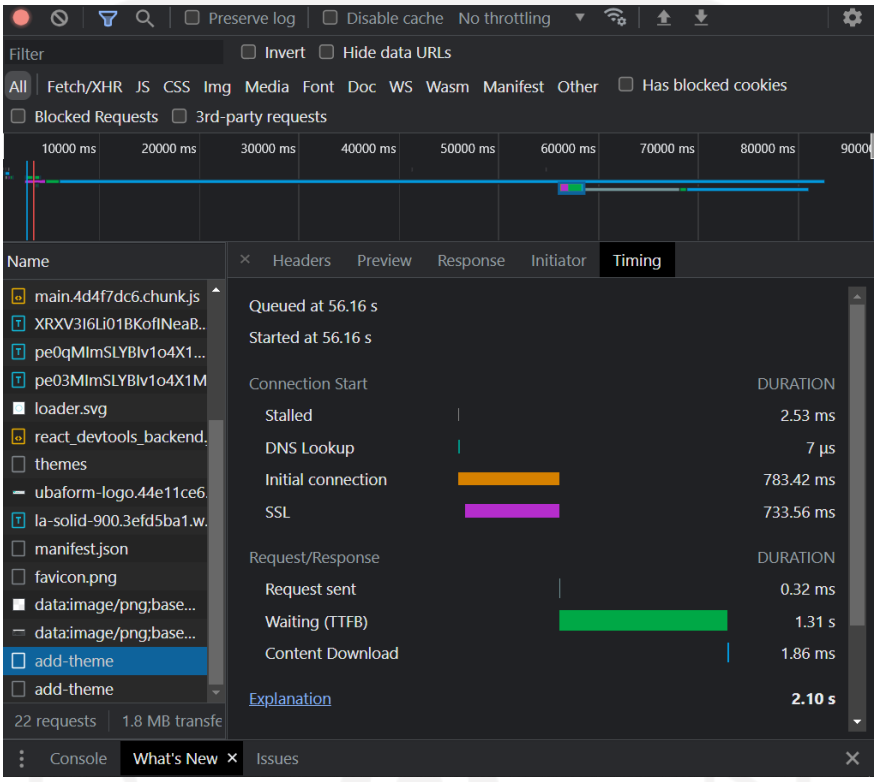
- Performa web Ubaform dengan *request* CRUD data ke server



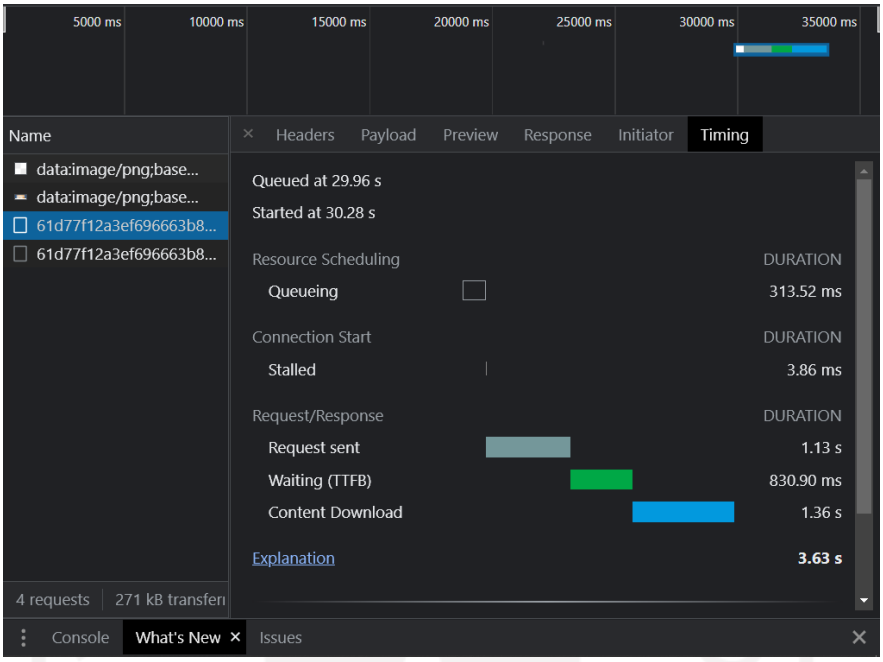
Gambar 4.71 Performa *Read/Fetch* data React JS

Pada gambar 4.71 dapat dilihat bahwa pada saat data *themes, customers* dan *employees* dimuat pada satu halaman dalam konteks ini halaman *dashboard Home* dilakukan atau dimulai pada waktu yang hampir bersamaan, hal ini merupakan fitur unggulan yang mana ketika menggunakan Javascript sebagai bahasa pengembangan dikedua sisi *backend dan frontend* dapat menggunakan fitur bawaan yang disebut *asynchronous.* Pada dasarnya sebuah kode program akan dieksekusi berdasarkan urutan misalnya mulai dari baris kode yang paling atas baru baris kode selanjutnya akan dieksekusi, hal ini akan membuat program menjadi lambat untuk dieksekusi karena dilakukan satu-persatu sesuai urutan. Namun beda halnya ketika menggunakan Javascript yang memiliki fitur *asynchronous* yang dapat melakukan ekskusi tanpa harus menunggu baris kode lainya selesai. Hal ini akan berdampak pada performa dari program sehingga menjadi lebih cepat. Dengan menerapkan fitur tersebut pada React JS sebagai *library* penyusun *user interface* dan Express JS sebagai penyusun *backend* dapat memaksimalkan fitur *asynchronous.* Ketika sisi *frontend* yang menggunakan React JS melakukan banyak *request* dengan fitur tersebut *request* yang dilakukan tidak dilakukan satu-persatu seperti sebuah antrian panjang namun dapat dilakukan secara bersamaan, hal ini juga berlaku pada sisi *backend* yang menggunakan Express JS ketika menerima dan memberikan

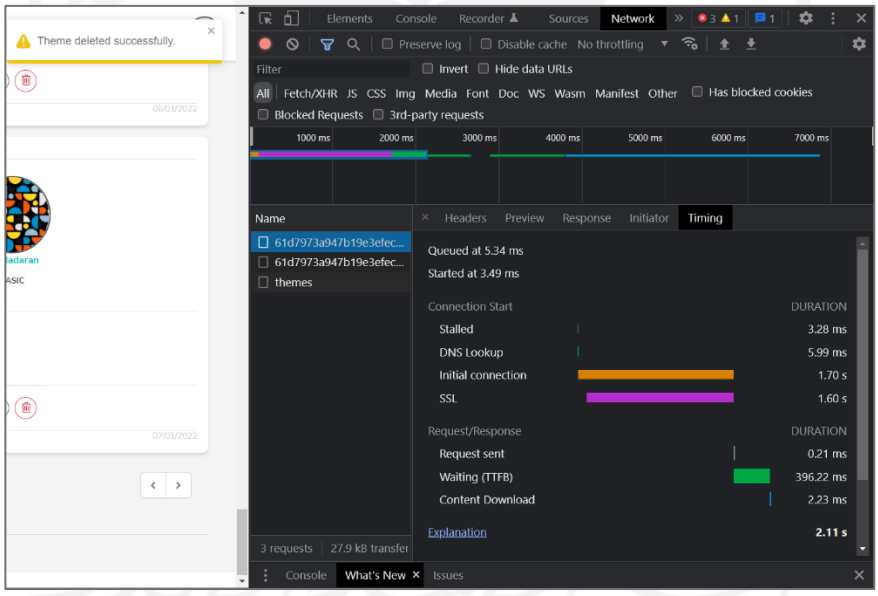
response data tidak perlu dilakukan satu-persatu, namun dapat dilakukan secara bersamaan sehingga eksekusi *request* dan *response* lebih cepat. Fitur *asynchronous* tidak akan terlalu efektif jika pengembang tidak full menggunakan Javascript dikedua sisi *frontend* dan *backend*, hal ini diakibatkan ketika hanya sisi *frontend* saja yang menggunakan fitur tersebut maka sisi *backend* tetap akan menerima dan memberikan *response* data secara berurutan satu-persatu.



Gambar 4.72 Performa *Request Add* data React JS



Gambar 4.73 Performa *Request Update* data React JS



Gambar 4.74 Performa *Request Delete* data React JS

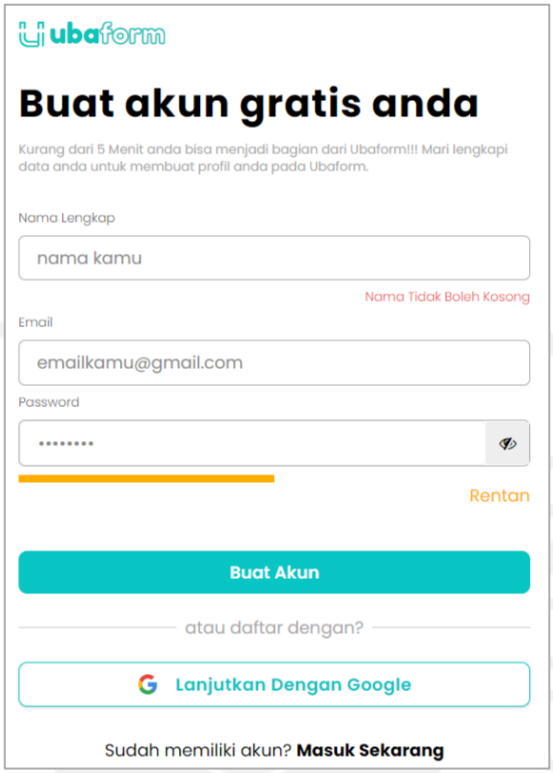
Pada gambar 4.72 sampai 4.74 merupakan performa *request* data yang dilakukan ke server. Gambar 4.72 merupakan *request add* data dengan waktu *request sent* 0.32 ms, gambar 4.73 merupakan *request update* data dengan waktu *request sent* 1.31 s dan gambar 4.74 merupakan *request delete* data dengan waktu *request sent* 0.2 ms. Ketiga performa request tersebut bahkan

tidak sampai pada waktu 2 detik. Hal ini masih dibawah tolak ambang batas pengguna meninggalkan website yaitu diatas 4 detik menurut Kinsta sebelumnya.

Performa pengembangan *user interface* web Ubaform berbasis SPA menggunakan React JS sudah terjawab pada perbandingan tersebut yaitu mampu meningkatkan performa. Meskipun pengembangan SPA tidak hanya dapat dilakukan menggunakan React JS namun faktor lain adalah fleksibilitas penggunaan React JS yang merupakan sebuah *library* yang tidak ada aturan bawaan seperti angular JS yang merupakan sebuah *framework*.

- Kemudahan dan kecepatan dalam implementasi React JS

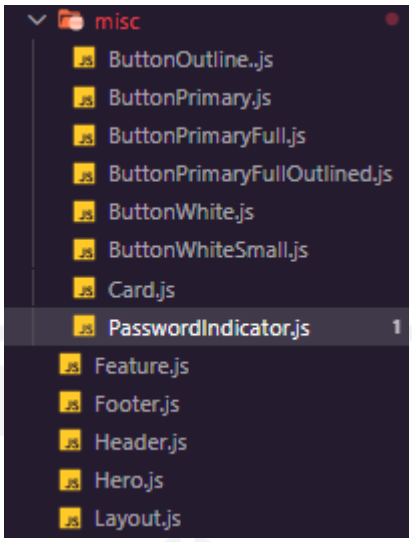
Dalam proses implementasi React JS tentunya hal yang memudahkan adalah tidak perlunya dilakukan penulisan kode dari awal untuk membuat sebuah algoritma atau fungsi tertentu. Hal ini dikarenakan pengembang dapat melakukan instalasi *library* lain kedalam projek React JS menggunakan NPM. Tidak hanya satu atau dua *library* yang dapat diinstal pada projek React JS melainkan tanpa batas, namun tentunya semakin banyak *library* yang diinstal maka semakin besar pula ukuran *file* projek. Oleh karena itu instalasi *library* juga perlu diperhatikan, jika suatu fungsi dapat dibuat secara manual dan tidak perlu waktu lama maka sebaiknya tidak menginstal *library* lain. Meskipun demikian dengan fleksibilitas dari React JS sebagai sebuah *library* yang dapat menerima *library* lain sangat membantu proses pengembangan. Selain itu juga pada React JS ukuran *file* projek memang besar namun ketika dilakukan *build production* projek tersebut akan dikompilasi dan dikompres menjadi sebuah folder dengan nama `build` dengan ukuran yang lebih kecil, hanya *file* inilah yang nantinya akan diluncurkan ke server sehingga mampu menghemat penyimpanan. Untuk salah satu contoh kemudahan dari proses pengembangan ini dapat dilihat sebagai berikut



Gambar 4.75 Implementasi *library* indikator pasword

Dapat dilihat pada gambar 4.75 bahwasanya pada kolom pasword terdapat indikator yang menentukan apakah password yang dimasukan oleh pengguna memiliki tingkat keamanan yang bagus atau tidak. Indikator pasword ini dibuat dengan menambahkan atau menginstal *library* lain yang bernama `zxcvbn` yang mana *library* ini memudahkan dan mempercepat proses pengembangan web Ubaform. Hal ini dikarenakan dengan *library* tersebut semua komponen penyusun indikator pasword sudah lengkap mulai dari fungsi, algoritma dan CSS *style*. Jadi pengembang hanya melakukan modifikasi dan penyesuaian sesuai kebutuhan saja.

Selain itu pengguna React JS dengan berbasis komponen-komponen kecil mampu mempercepat proses pengembangan yaitu dengan dapat digunakannya secara berulang-ulang komponen kecil tersebut kedalam komponen mana saja yang membutuhkan. Hal ini dapat dilihat sebagai berikut



Gambar 4.76 *Reusable* komponen pada web Ubaform

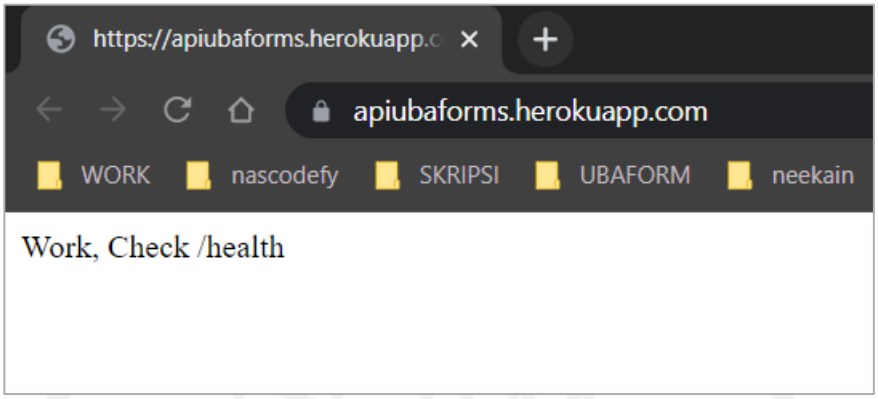
Dapat dilihat pada gambar 4.76 terdapat beberapa komponen yang dibuat dengan atribut tertentu namun bersifat dapat digunakan berulang-ulang *(reusable)* tentunya secara dinamis untuk setiap halaman yang diinginkan. Oleh karena keunggulan bawaan React JS tersebut implementasi ini membantu mempercepat pengembangan web Ubaform.

4.3 Hasil dan Pembahasan Implementasi Express JS

Implementasi Express JS pada web Ubaform sebagai *framework* yang digunakan untuk membangun sisi *backend* adalah pilihan yang tepat. Selain dikarenakan penggunaan Mongo DB sebagai database yang berbasis JSON sehingga sangat cocok dipadukan dengan bahasa pemrograman javascript, hal ini juga dikarenakan performa dan kemudahan yang diberikan oleh Express JS.

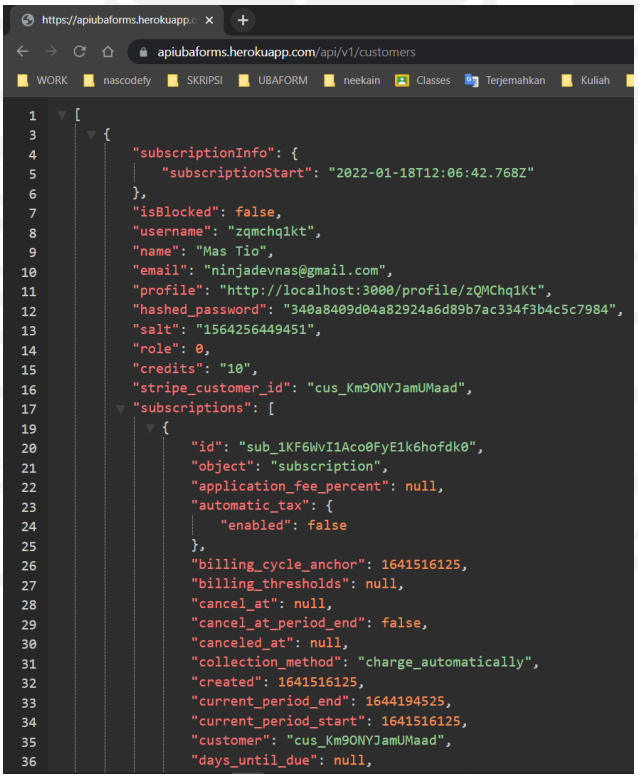
4.3.1 Hasil

Hasil yang didapat dari implementasi Express JS tentunya berupa API yang telah terintegrasi dengan Mongo DB. Dengan API ini sisi *frontend* akan dapat terhubung ke bagian server atau *backend* dari web Ubaform. API yang telah dibuat dalam pengembangan web Ubaform dapat diakses pada https://apiubaforms.herokuapp.com/. Namun untuk lebih lengkapnya akses API dilakukan dengan menggunakan *browser* chrome maupun Postman dan hasilnya dapat dilihat sebagai berikut



Gambar 4.77 Akses *base* URL API web Ubaform

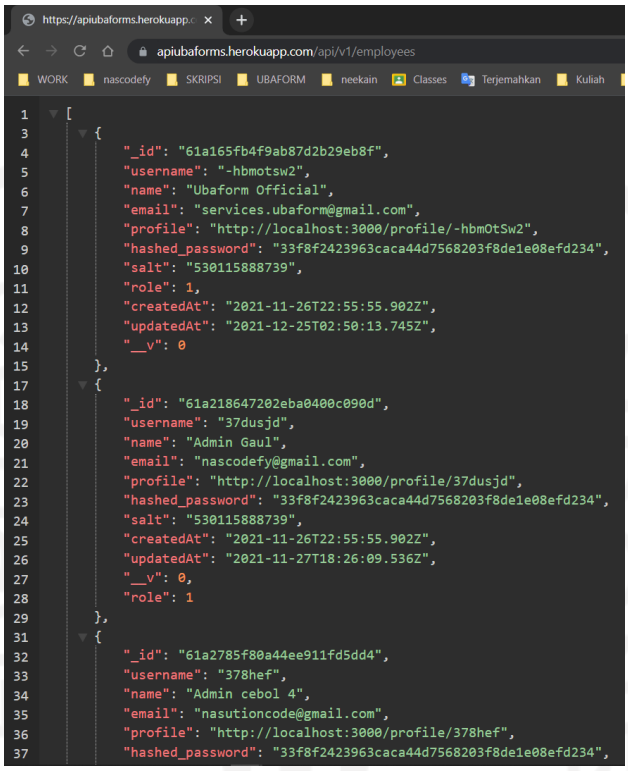
Dapat dilihat pada gambar 4.77 akses *base* URL API web Ubaform dapat diakses melalui browser chrome dan ini juga berlaku untuk pengaksesan dari mana saja bahkan menggunakan *mobile* atau sejenisnya. Hal ini dikarenakan API web Ubaform sudah dilakukan peluncuran *(deployment)* ke sisi server.



Gambar 4.78 Akses API data *customers* Ubaform

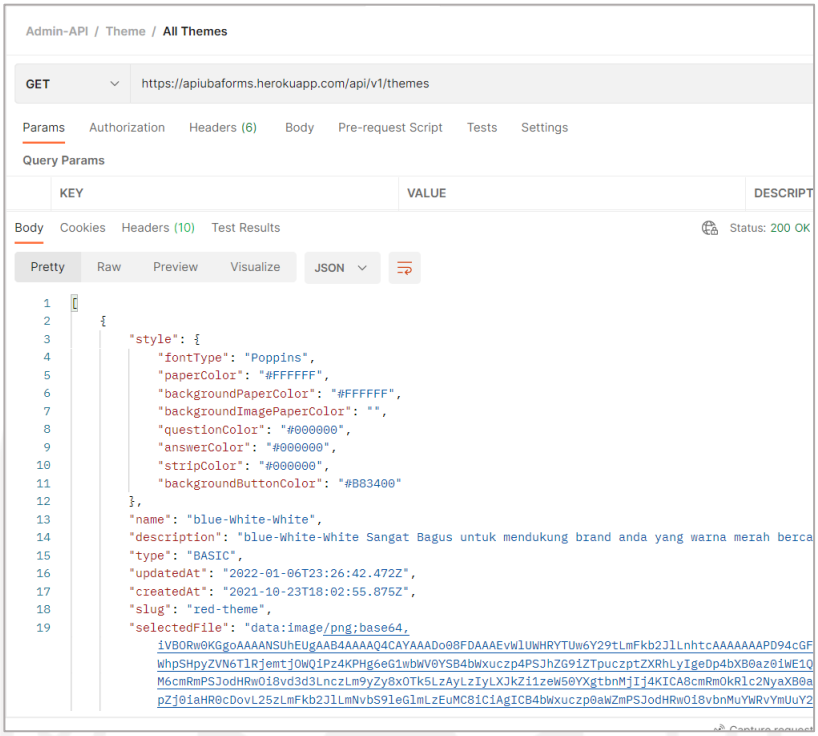
Data yang terlihat pada gambar 4.78 adalah data-data *customer* yang terdaftar dalam web Ubaform. Dapat dilihat bahwa format data pada gambar 4.78 adalah berupa JSON. Data ini

yang akan diolah dan ditampilkan pada sisi *frontend* web Ubaform dengan mengakses API https://apiubaforms.herokuapp.com/api/v1/customers.



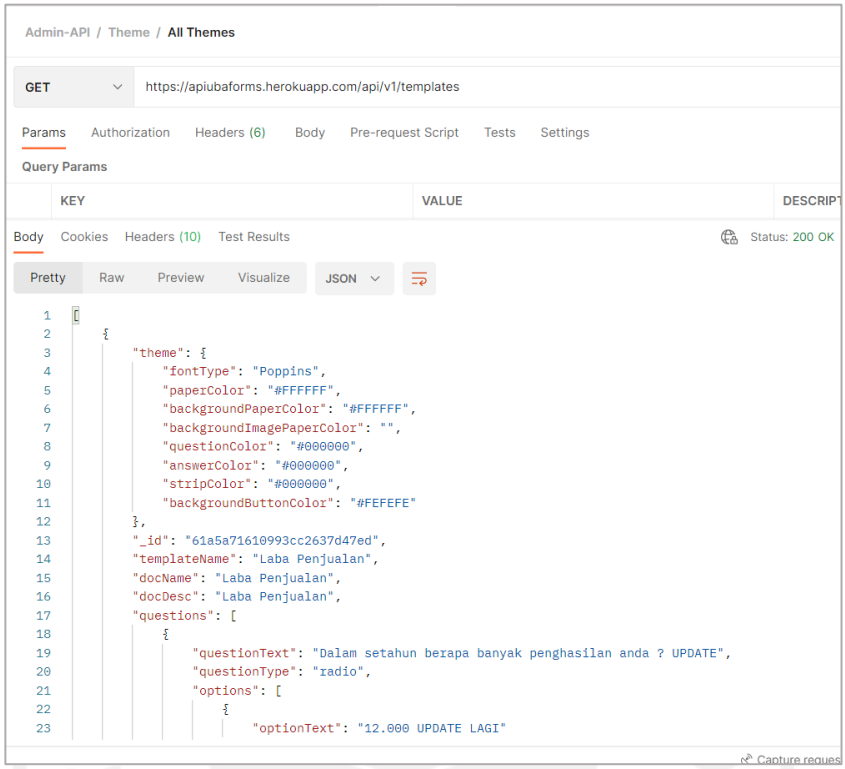
Gambar 4.79 Akses API data *employees* Ubaform

Data yang terlihat pada gambar 4.79 adalah data-data *employee* yang terdaftar dalam web Ubaform. Data ini yang akan diolah dan ditampilkan pada sisi *frontend* web Ubaform dengan mengakses API https://apiubaforms.herokuapp.com/api/v1/employees.



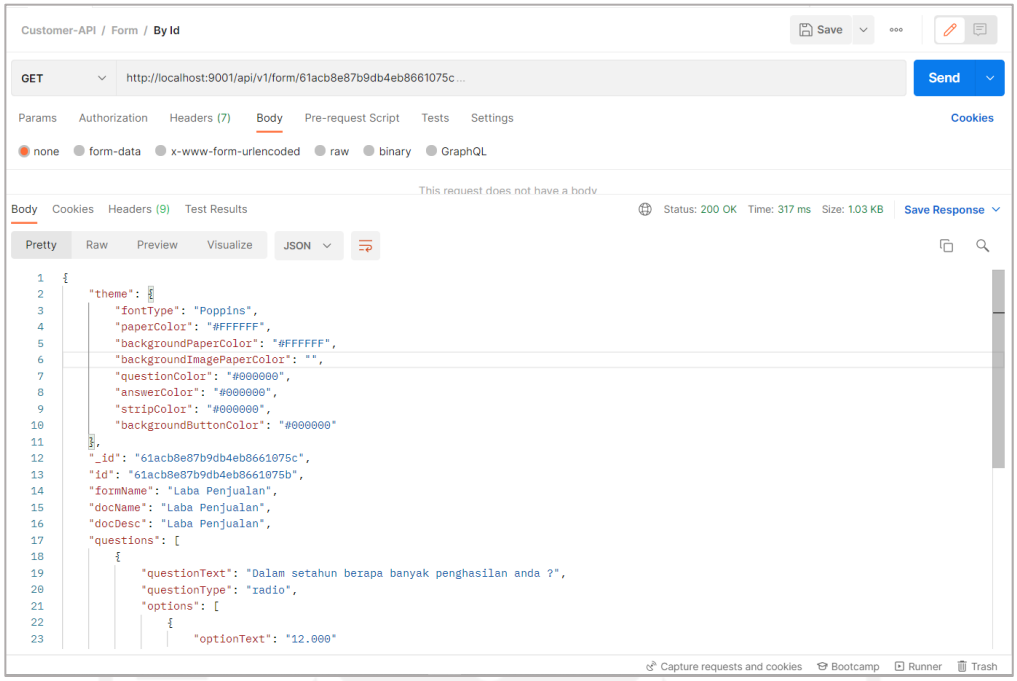
Gambar 4.80 Akses API data *themes* Ubaform

Data yang terlihat pada gambar 4.80 adalah data-data *themes* yang telah dibuat oleh admin dan dapat digunakan untuk melakukan kostumisasi tema pada saat pembuatan formulir, kuis atau survei. Data ini yang akan diolah dan ditampilkan pada sisi *frontend* web Ubaform dengan mengakses API https://apiubaforms.herokuapp.com/api/v1/themes.



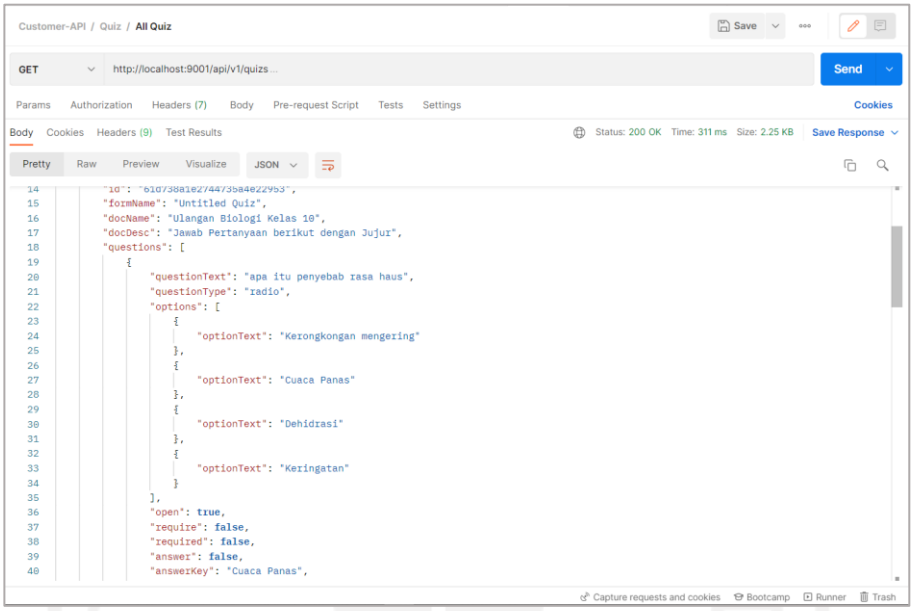
Gambar 4.81 Akses API data *templates* Ubaform

Data yang terlihat pada gambar 4.81 adalah data-data *templates* yang telah dibuat oleh admin dan dapat digunakan sebagai template dalam pembuatan formulir, kuis atau survei. Dengan data ini *customer* tidak perlu membuat ketiga hal tersebut dari awal namun hanya melakukan modifikasi pertanyaan sesuai kebutuhan. Data ini yang akan diolah dan ditampilkan pada sisi *frontend* web Ubaform dengan mengakses API https://apiubaforms.herokuapp.com/api/v1/templates.



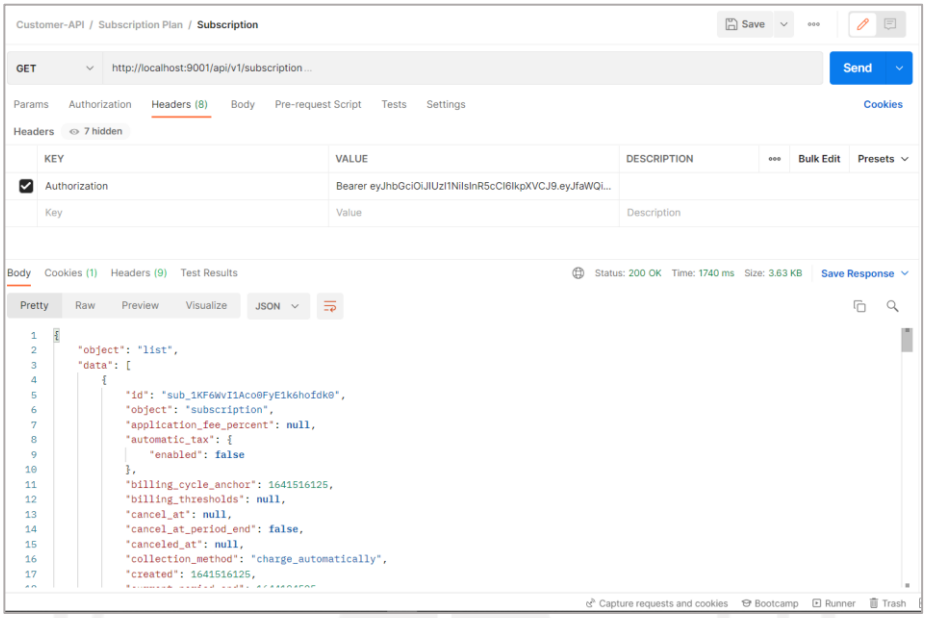
Gambar 4.82 Akses API data *form* Ubaform

Data yang terlihat pada gambar 4.82 adalah data-data projek formulir yang telah dibuat oleh *customer*. Akses data ini hanya dapat dilakukan oleh *customer* yang membuatnya sedangkan admin hanya dapat melihat saja. Oleh karena itu akses data ini memerlukan token dari *customer* pembuatnya.



Gambar 4.83 Akses API data *quiz* Ubaform

Data yang terlihat pada gambar 4.83 adalah data-data projek kuis yang telah dibuat oleh *customer*. Akses data ini hanya dapat dilakukan oleh *customer* yang membuatnya sedangkan admin hanya dapat melihat saja. Oleh karena itu akses data ini memerlukan token dari *customer* pembuatnya.

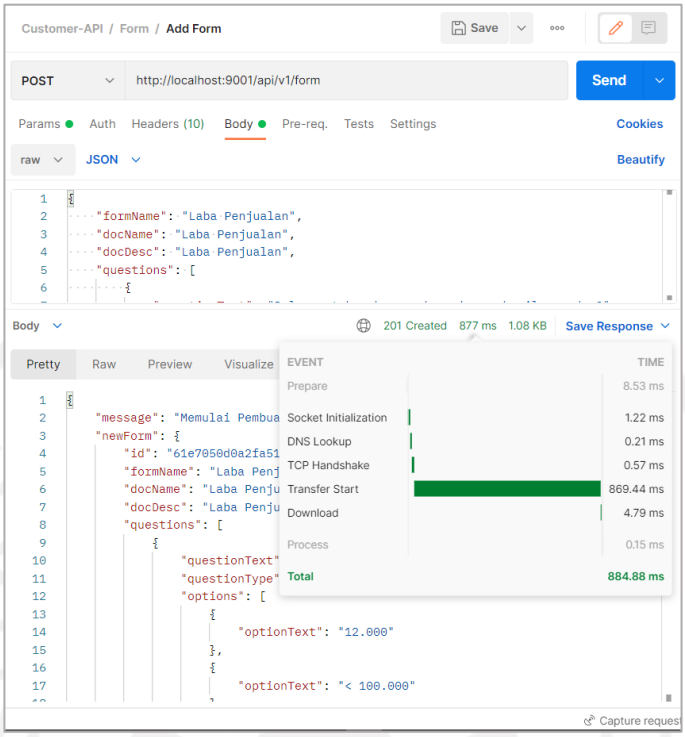


Gambar 4.84 Akses API data *subscriptions* Ubaform

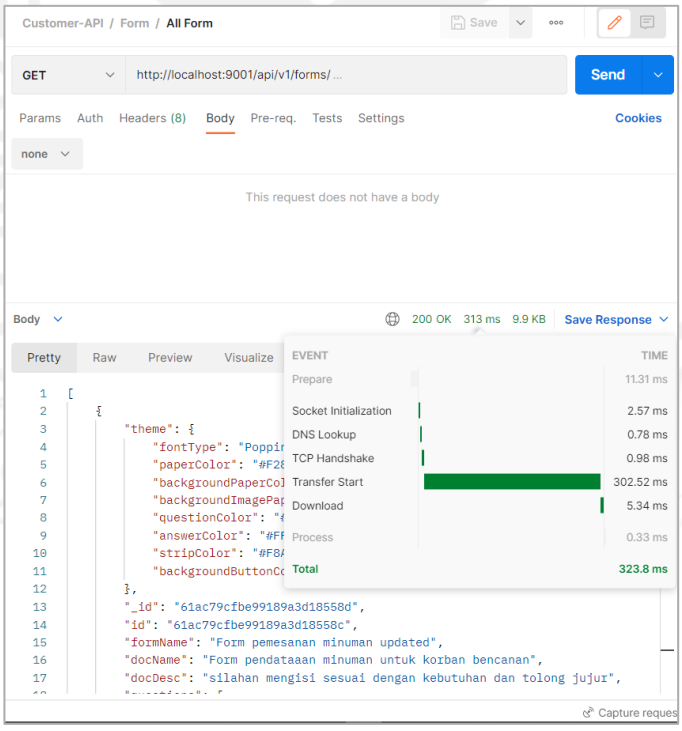
Data yang terlihat pada gambar 4.84 adalah data-data projek formulir yang telah dibuat oleh *customer*. Akses data ini hanya dapat dilakukan oleh *customer* yang membuatnya sedangkan admin hanya dapat melihat saja. Oleh karena itu akses data ini memerlukan token dari *customer* yang melakukan transaksi pembelian *subscription*.

4.3.2 Pembahasan

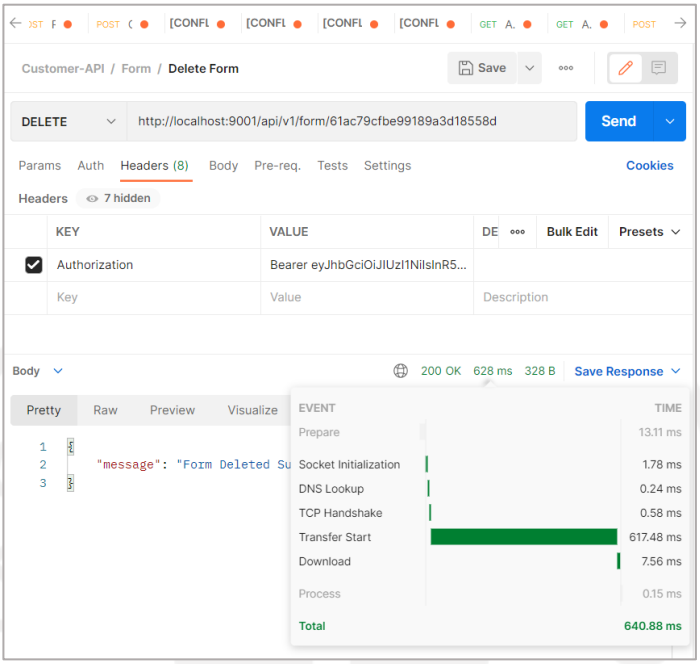
- Performa CRUD dengan Express JS



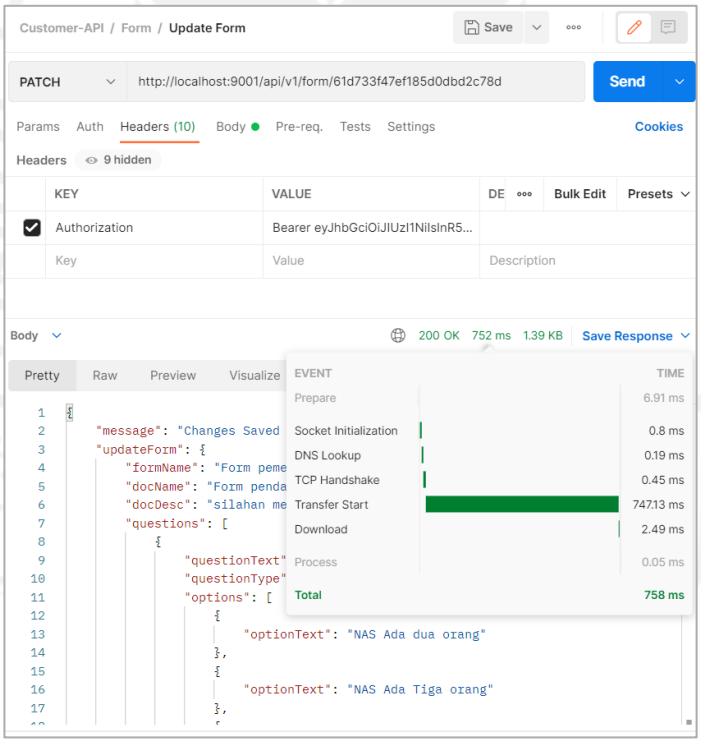
Gambar 4.85 Performa *Create* data framework Express JS



Gambar 4.86 Performa *Read* data *framework* Express JS



Gambar 4.87 Performa *Delete* data *framework* Express JS



Gambar 4.88 Performa *Update* data *framework* Express JS

Dapat dilihat pada gambar 4.85 sampai 4.88 merupakan catatan waktu performa Express JS dalam melakukan CRUD data web Ubaform dengan API. Pengetesan performa ini

dilakukan dengan menggunakan aplikasi Postman dengan fitur yang dilakukan pengetesan adalah fitur manajemen formulir. Dengan performa *create* data adalah 877 ms, *read* data 313 ms, *update* data 628 ms dan *delete* data 752 ms. Berdasarkan kecepatan performa tersebut dapat disimpulkan bahwa kecepatan yang didapat dari implementasi Express JS dengan data web Ubaform bahkan tidak sampai menghabiskan waktu 1 detik. Waktu tersebut sangat cepat jika dibandingkan waktu pemuatan web yang tidak normal menurut Kinsta adalah diatas 4 detik. Meskipun hal ini bukan merupakan keseluruhan proses pemuatan data pada website melainkan hanya pemuatan data saja, namun hal ini tetap mempunyai dampak besar terhadap jumlah waktu yang dipengaruhi terhadap kecepatan pemuatan website. Hal ini dapat dilihat pada pemuatan keseluruhan web Ubaform mulai dari data sampai komponen penyusun pada pembahasan implementasi React JS sebelumnya.

- Kemudahan dan kecepatan pengembangan web Ubaform dengan Express JS

Untuk kemudahan pengembangan web Ubaform dengan implementasi Express JS hampir sama halnya dengan React JS. Meskipun Express JS merupakan sebuah *framework* namun bersifat tidak memiliki aturan penggunaan *(unopinionated)* dimana kebebasan penggunaan *framework* diserahkan sepenuhnya kepada pengembang. Dengan fleksibilitas yang dibawa terrsebut Express JS memudahkan pengembang dalam membangun sisi *backend* atau server. Hal ini juga menyebabkan pengembang dapat dengan mudah melakukan instalasi *library* lain kedalam projek sehingga selain mempermudah juga mempercepat proses pengembangan.

4.4 Implementasi Node JS

Implementasi Node JS dapat diartikan menjadi inti dari implementasi React JS dan Node JS sebelumnya. Hal ini dikarenakan implementasi dari kedua hal tersebut dapat terjadi karena adanya Node JS sebagai *environment* untuk menjalankan serta menginstal kedua hal tersebut dan juga *library* pendukung lainnya. Tidak hanya itu bahkan implementasi dari Mongo DB perlu adanya instalasi *library* dan hal tersebut juga membutuhkan Node JS. Jadi dapat disimpulkan bahwasanya kemudahan dan kecepatan pengembangan web Ubaform adalah dampak dari implementasi React JS, Express JS dan Mongo DB dengan Node JS sebagai *environment* pengembangan yang menjalankan, menginstal, melakukan *debug* dan lain sebagainya.

BAB V
KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil serta proses *development* implementasi MERN pada pengembangan aplikasi *startup* Ubaform dapat disimpulkan bahwasanya, implementasi tersebut dapat meningkatkan performa dari web Ubaform baik pada sisi *backend, frontend* atau keseluruhan pemuatan data dan *resource* penyusun halaman web lainnya. Hal tersebut dapat dilihat dari catatan waktu yang telah dijelaskan pada Bab IV sebelumnya.

Penggunaan MERN sebagai teknologi pengembangan web juga sangat memberi kemudahan dan efisiensi waktu pengerjaan lebih cepat dari awal sampai akhir. Hal ini dikarenakan dengan adanya *reusable* komponen pada React JS serta dapat melakukan instalasi *library* atau *package* tambahan yang sangat membantu mempercepat dan mempermudah menggunakan NPM. Dengan adanya NPM waktu yang digunakan untuk mengembangkan sebuah fitur yang awalnya membutuhkan waktu sehari atau lebih dapat diefisiensikan menjadi beberapa jam pengerjaan dikarenakan adanya instalasi *library* pendukung fitur yang dikembangkan tersebut. Hal ini dapat dilihat secara detail pada pembahasan implementasi React JS di Bab IV penelitian ini.

Selain itu MERN juga berbasis JavaScript memudahkan pengembang dalam memahami kode baik di sisi *frontend* dan *backend* sekaligus. Meskipun ada beberapa teknologi lain yang dapat diimplementasikan sebagai alternatif lain seperti angular JS untuk membangun sisi *frontend* berbasis SPA dan MySQL sebagai database, namun React JS lebih fleksibel karena pada dasarnya React JS merupakan sebuah *library* yang mana lebih fleksibel penggunaannya seperti struktur folder dan penambahan *library* pendukung lainnya sedangkan untuk Mongo DB tentu dalam penelitian ini memiliki performa yang lebih baik dibandingkan dengan MySQL serta kemudahan dalam struktur penyimpanan data yang berupa JSON yang mana lebih fleksibel dan mudah untuk diolah seperti yang dijelaskan secara rinci pada Bab IV sebelumnya.

5.2 Saran

Untuk saran pada proses pengembangan berikutnya adalah untuk dapat menyelesaikan dan meningkatkan keseluruhan fitur yang ada seperti fitur pengerjaan dan hasil formulir, kuis atau survei dan memperbanyak pilihan tema dan *template* yang ada serta komponen kostumisasi lainnya seperti kostumisasi font atau penggunaan gambar. Selain itu untuk dapat memperkaya kategori pertanyaan dan dengan pilihan pembuatan menggunakan suara, video atau gambar. Hal lain yang dapat ditingkatkan adalah kualitas penulisan kode untuk kedepannya dapat terformat sesuai standar yang tepat agar pengembang dapat dengan mudah memahami serta membuat dokumentasi sehingga hal ini juga akan membantu proses *maintenance* kedepannya.

DAFTAR PUSTAKA

StrongLoop, IBM, and other expressjs.com contributors. (2021, September 11). *Express Homepage*. Retrieved from Express Website: https://expressjs.com/

Developer Mozila. (2021, September 13). *Express/Node introduction*. Retrieved from Developer Mozila Website: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

Dicoding. (2021, May 19). *Contoh Use Case Diagram Lengkap dengan Penjelasannya*. Retrieved from Dicoding: https://www.dicoding.com/blog/contoh-use-case-diagram/

DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte. (2020). *Thesis*, 58-66.

Einav, Y. (2019, Januari 20). *Amazon Found Every 100ms of Latency Cost them 1% in Sales*. Retrieved from GIGASPACES: https://www.gigaspaces.com/blog/amazon-found-every-100ms-of-latency-cost-them-1-in-sales

Facebook. (2021, October 14). *React A JavaScript library for building user interfaces*. Retrieved from React: https://reactjs.org/

Facebook. (2021, Oktober 15). *Thinking in React*. Retrieved from React: https://id.reactjs.org/docs/thinking-in-react.html

Hadjigeorgiou, C. (2013). RDBMS vs NoSQL: Performance and Scaling. *Christoforos Hadjigeorgiou*, 25-30. Retrieved from https://static.epcc.ed.ac.uk/dissertations/hpc-msc/2012-2013/RDBMS%20vs%20NoSQL%20-%20Performance%20and%20Scaling%20Comparison.pdf

HANRY HAM., S. M. (2019, 12 30). *Apa itu React.Js?* Retrieved from Binus University Chool of Science: https://socs.binus.ac.id/2019/12/30/apa-itu-react-js/

IBM Cloud Education. (2020, Desember 21). *What is MongoDB?* Retrieved from IBM website: https://www.ibm.com/cloud/learn/mongodb

Jeff Gothelf, J. S. (2021, October 17). *Chapter 5. Minimum Viable Products and Prototypes*. Retrieved from Oreilly: https://www.oreilly.com/library/view/lean-ux-2nd/9781491953594/ch05.html

Karlsson, J. (2020, December 04). *MongoDB Schema Design Best Practices*. Retrieved from MongoDB: https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

KBBI. (2021, September 8). *KBBI Daring*. Retrieved from KBBI Daring Website: https://kbbi.kemdikbud.go.id/entri/situs%20web

KBBI. (2021, September 8). *KBBI Daring*. Retrieved from KBBI Daring Website: https://kbbi.kemdikbud.go.id/entri/domain

Kemp, S. (2021, September 8). *DIGITAL 2021: GLOBAL OVERVIEW REPORT*. Retrieved from Data Reportal Website: https://datareportal.com/reports/digital-2021-global-overview-report

KINSTA. (2021, May 18). *A Beginner's Guide to Website Speed Optimization*. Retrieved from Kinsta: https://kinsta.com/learn/page-speed/#intro

Kunwar, K. (2021, October 16). *Why PayPal migrated from java to Node.js- going back to the time*. Retrieved from Prokura: https://www.prokurainnovations.com/why-paypal-migrated-from-java-to-node-js-going-back-to-the-time/

Microsoft. (2017, Juni 23). *Sharding pattern*. Retrieved from Microsoft Documentation Website: https://docs.microsoft.com/en-us/azure/architecture/patterns/sharding

Mistry, J. (2021, October 11). *A Comprehensive Guide To The 7 Phases of Web Development Life Cycle*. Retrieved from Monocubed: https://www.monocubed.com/web-development-life-cycle/

Mullins, C. S. (2018, February 15). *What Do We Mean by Database Scalability?* Retrieved from D Zone Website: https://dzone.com/articles/what-do-we-mean-by-database-scalability

Nodejs.org. (2021, October 15). *Node.js v8.17.0 Documentation*. Retrieved from Node.js: https://nodejs.org/dist/latest-v8.x/docs/api/documentation.html

npm, inc. (2021, October 15). *About npm.* Retrieved from npmjs: https://www.npmjs.com/about

OpenJS Foundation. (2021, October 15). *About Node.js*. Retrieved from NODE JS: https://nodejs.org/en/about/

OpenJS Foundation. (2021, October 15). *An introduction to the npm package manager*. Retrieved from nodejs: https://nodejs.dev/learn/an-introduction-to-the-npm-package-manager

Oxford University Press. (2021, September 8). *Definition of server noun*. Retrieved from Oxford Learner's Dictionaries: https://www.oxfordlearnersdictionaries.com/definition/english/server?q=server

Oxford University Press. (2021, 9 8). *Definition of website noun*. Retrieved from Oxford Learner's Dictionaries: https://www.oxfordlearnersdictionaries.com/definition/english/website?q=website

Patel, R. (2019, July 1). *Node Js vs PHP: Comparing Stats, Features and Performance in 2021*. Retrieved from AGLOWID: https://aglowiditsolutions.com/blog/node-js-vs-php/

Paul DuBois. (2021, September 8). *MySQL 8.0 Reference Manual Data Types*. Retrieved from My SQL Documentation Website: https://dev.mysql.com/doc/refman/8.0/en/data-types.html

Sacha Greif, R. B. (2019, December 22). *State of JavaScript survey, 2019*. Retrieved from stateofjs: https://2019.stateofjs.com/

Schaefer Lauren. (2021, September 10). *NoSQL vs SQL Databases*. Retrieved from Mongo DB Inc Website: https://www.mongodb.com/nosql-explained/nosql-vs-sql

Schaefer, L. (2021, September 9). *What is NoSQL?* Retrieved from MongoDB, Inc: https://www.mongodb.com/nosql-explained

Schwarzmüller, M. (2018, Juli 25). *SQL vs NoSQL*. Retrieved from Academind Website: https://academind.com/tutorials/sql-vs-nosql

Shah, M. (2017, November 23). *MongoDB vs MySQL: A Comparative Study on Databases*. Retrieved from SIMFORM: https://www.simform.com/blog/mongodb-vs-mysql-databases/

Sharma, K. (2020, December 25). *TOP 12 SOFTWARE DEVELOPMENT METHODOLOGIES & ITS ADVANTAGES & DISADVANTAGES*. Retrieved from TatvaSoft: https://www.tatvasoft.com/blog/top-12-software-development-methodologies-and-its-advantages-disadvantages/

StrongLoop, IBM, and other expressjs.com contributors. (2021, September 13). *Companies using Express in production*. Retrieved from Express JS Website: https://expressjs.com/en/resources/companies-using-express.html

Vorbach, P. (2021, October 16). *npm-stat*. Retrieved from npm-stat: https://npm-stat.com/charts.html?package=clone

WAIROOY, I. K. (2020, 03 31). *Software Engineering : Agile Software Development Process Model – Scrum Model*. Retrieved from BINUS University school of computer science: https://socs.binus.ac.id/2020/03/31/software-engineering-agile-software-development-process-model-scrum-model/

LAMPIRAN