# PENGEMBANGAN *RESTFUL API* DAN WEBSITE ADMIN UNTUK APLIKASI *SCREENING SPEECH DELAY*

**(Skripsi)**

**Oleh**

**SILVIA NAIM**
**NPM 1915061018**

**FAKULTAS TEKNIK**
**UNIVERSITAS LAMPUNG**
**BANDAR LAMPUNG**
**2023**

# PENGEMBANGAN *RESTFUL API* DAN WEBSITE ADMIN UNTUK APLIKASI *SCREENING SPEECH DELAY*

**Oleh**

**SILVIA NAIM**

**Skripsi**

**Sebagai Salah Satu Syarat untuk Mencapai Gelar**
**SARJANA TEKNIK**

**Pada**

**Program Studi Teknik Informatika**
**Jurusan Teknik Elektro**
**Fakultas Teknik Universitas Lampung**


**FAKULTAS TEKNIK**
**UNIVERSITAS LAMPUNG**
**BANDAR LAMPUNG**
**2023**

# ABSTRAK

## PENGEMBANGAN *RESTFUL API* DAN WEBSITE ADMIN UNTUK APLIKASI *SCREENING SPEECH DELAY*

**Oleh**

**SILVIA NAIM**

*Speech delay* merupakan suatu keadaan terlambat berbicara yang ditandai dengan kemampuan berbicara anak di bawah rata-rata dibandingkan dengan anak lainnya. Peran orang tua sangat penting dalam menunjang kemampuan berbahasa dan bicara anak. Sayangnya, pengetahuan tentang perkembangan ini tidak dimiliki oleh semua orang tua. Di tempat penelitian Rumah Terapi Al-Birru, *screening speech delay* dilakukan secara manual dan harus di lakukan secara *offline*. Pada penelitian ini dikembangkan *RESTful API* dan Website Admin untuk Aplikasi *Screening Speech Delay*. Platform ini memberikan kemudahan penggunaan untuk membantu orang tua mendeteksi dini *speech delay*. Dengan mengidentifikasi masalah bahasa dan bicara anak secara cepat, intervensi dapat dilakukan lebih awal untuk hasil yang lebih efektif. Metode yang diterapkan dalam pengembangan sistem ini adalah metode *Agile Scrum*. Selama 5 *Sprint* dengan total 42 *Product Backlog*, didapatkan hasil dari pengembangan berupa *RESTful API* dan platform pengelolaan di sisi admin. *RESTful API* telah diuji menggunakan metode pengujian *Black Box* dalam 18 skenario, sementara Website Admin telah diuji menggunakan metode *End to End Testing* melalui tools *Cypress* dalam 12 skenario. Kedua pengujian tersebut berhasil mencapai keluaran sesuai dengan yang diharapkan. Kemudian dilakukan *Stress Testing* untuk mengetahui tingkat kinerja *RESTful API*. Hasil *Stress Testing* didapatkan bahwa Aplikasi *Screening Speech delay* akan optimal apabila maksimal diakses oleh 120 *request* dalam periode *ramp up* 1 detik dan *loop count* sebanyak 1 kali.

Kata kunci : *Screening Speech delay*, *RESTful API*, Website Admin, *Agile Scrum*

## *ABSTRACT*

### *DEVELOPMENT OF RESTFUL API AND ADMIN WEBSITE FOR SPEECH DELAY SCREENING APPLICATION*

***By***

**SILVIA NAIM**

*Speech delay is a condition characterized by a delay in speech development, where a child's speaking ability falls below the average compared to other children. The role of parents is crucial in supporting a child's language and speech skills. Unfortunately, not all parents possess knowledge about this developmental aspect. At Rumah Terapi Al-birru in Bandar Lampung research facility, speech delay screening is currently conducted manually and offline. In this study, a RESTful API and an Admin Website were developed for the Speech Delay Screening Application. This platform offers user-friendly features to help parents detect speech delay early on. By identifying language and speech issues in children quickly, interventions can be initiated earlier for more effective outcomes. The Agile Scrum method was employed in the development of this system, which spanned five sprints and included a total of 42 Product Backlog items. The development resulted in a RESTful API and an administrative management platform. The RESTful API was tested using Black Box testing in 18 scenarios, while the Admin Website underwent End-to-End Testing using Cypress tools in 12 scenarios. Both tests successfully yielded the expected outcomes. Furthermore, Stress Testing was performed to assess the performance level of the RESTful API. The Stress Testing results indicated that the Speech Delay Screening Application would perform optimally when accessed by a maximum of 120 requests within a 1-second ramp-up period and a loop count of 1.*



*Keyword* : *Screenin g Speech delay, RESTful API*, Website Admin, *Agile Scrum*

| | | |
|---|---|---|
| Judul Skripsi | : | **PENGEMBANGAN *RESTFUL API* DAN WEBSITE ADMIN UNTUK APLIKASI *SCREENING SPEECH DELAY*** |
| Nama Mahasiswa | : | Silvia Naim |
| Nomor Pokok Mahasiswa | : | 1915061018 |
| Program Studi | : | Teknik Informatika |
| Jurusan | : | Teknik Elektro |
| Fakultas | : | Teknik |

**MENYETUJUI**

1. Komisi Pembimbing

| Pembimbing Utama | Pembimbing Pendamping |
|---|---|
| **Ing. Hery Dian Septama, S.T.** | **Puput Budi Wintoro, S.Kom., M.T.I.** |
| NIP 19850915 200812 1 001 | NIP 19841031 201903 1 004 |

2. Mengetahui

| Ketua Jurusan<br>Teknik Elektro | Ketua Program Studi<br>Teknik Informatika |
|---|---|
| **Herlinawati, S.T., M.T.** | **Mona Arif Muda, S.T., M.T.** |
| NIP 19710314 199903 2 001 | NIP 19711112 200003 1 002 |

**MENGESAHKAN**

1. Tim Penguji

Ketua : **Ing. Hery Dian Septama, S.T.** .....

Sekretaris : **Puput Budi Wintoro, S.Kom., M.T.I.** .....

Penguji : **Yessi Mulyani, S.T., M.T.** .....

2. Dekan Fakultas Teknik

**Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc.**
NIP. 19750928 200112 1 002

Tanggal Lulus Ujian Skripsi : **7 Agustus 2023**

## SURAT PERNYATAAN

Saya yang bertandatangan dibawah ini, menyatakan bahwa skripsi saya yang berjudul "Pengembangan *RESTful API* dan Website Admin untuk Aplikasi *Screening Speech Delay*" dengan ini menyatakan bahwa skrpsi saya dibuat oleh saya sendiri. Semua hasil yang tertuang dalam skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung. Apabila di kemudian hari terbukti bahwa skripsi merupakan salinan atau dibuat oleh orang lain, maka saya bersedia menerima sanksi sesuai dengan ketentuan hukum atau akademik yang berlaku.

Bandar Lampung, 7 Agustus 2023

Pembuat pernyataan,


Silvia Naim

NPM. 1915061018

# RIWAYAT HIDUP

Penulis dilahirkan di Pesawaran, pada tanggal 2 Juli 2001. Penulis merupakan anak kedua dari 2 bersaudara dari pasangan Bapak Dawud dan Ibu Pariyah.

Penulis memulai jenjang pendidikan dari TK Al-Imran, SD Negeri 1 Pejambon dan lulus pada tahun 2013, SMP PGRI Pejambon dan lulus pada tahun 2016, SMA Negeri 1 Natar dan lulus pada tahun 2019 dan di tahun yang sama diterima pada Program Studi Teknik Informatika Universitas Lampung melalui jalur SBMPTN.

Selama menjalani proses perkuliahan secara aktif di Jurusan Teknik Elektro Program Studi Teknik Informatika Universitas Lampung, penulis mengikuti Himpunan Mahasiswa Teknik Elektro (Himatro) Unila sebagai anggota Departemen Sosial dan Wirausaha pada Periode 2020 dan sebagai anggota Departemen Pendidikan dan Pengembangan Diri pada Periode 2021. Pada tahun 2022, penulis mendapatkan kesempatan untuk mengikuti Magang dan Studi Independen Bersertifikat (MSIB) di Binar Academy pada Learning Path *Front-end JavaScript*. Pada tahun yang sama, penulis berkesempatan mendapatkan beasiswa kelas Dicoding bidang *React.JS* Developer hinga level mahir yang diselenggarakan oleh IDCamp Indosat Ooredoo. Kemudian pada Mei 2023, penulis berhasil mendapatkan juara 3 pada kompetisi *Mini Hackathon* bertemakan "*Unlocking HER Potentials*" yang diselenggarakan oleh Web Programming Unpas (WPU).

## MOTTO

"Belajarlah agar dapat mengamalkannya, bukan semata-mata hanya untuk mencari nilai atau pengakuan sebagai orang yang pintar, berilmu dan sejenisnya"

"… dan aku belum pernah kecewa dalam berdoa kepada Engkau, wahai Rabbku."
**(QS. Maryam Ayat 4)**

"Sebaik-baik manusia adalah yang paling bermanfaat bagi manusia"
**(HR. Ahmad, ath-Thabrani, ad-Daruqutni)**

"Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya…"
**(QS. Al-Baqarah Ayat 286)**

"Menyia-nyiakan waktu lebih mengerikan daripada kematian. Sebab, menyia-nyiakan waktu akan memisahkanmu dari Allah dan negeri akhirat. Sementara itu, kematian hanya akan memisahkanmu dari dunia dan penghuninya."
**(Imam Ibnul Qayyim)**

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Dengan menyebut nama Allah Yang Maha Pengasih lagi Maha Penyayang. *Alhamdulillahirobbilalamin* dengan segala anugerah dan karunia-Mu yang tiada habisnya kepada penulis, skripsi ini dapat diselesaikan dengan baik dan lancar. Semoga dengan keberhasilan yang telah dicapai ini penulis dapat menuju masa depan yang lebih baik dan dapat menggapai cita-cita serta selalu berada di jalan-Mu.

Kudedikasikan karya ini kepada :

"Ibunda Pariyah dan Ayahanda Dawud atas dukungan dan kasih sayang tulus sepanjang hidup. Terima kasih kepada Ibu dan Bapak atas doa yang tak henti-hentinya dipanjatkan serta pengorbanan yang tak terhitung nilainya. Semoga Allah selalu memberikan rahmat serta ridho-Nya untuk Ibu dan Bapak"

"Kakakku Rizky Sugesti yang selalu memberikan semangat dan motivasi serta pembelajaran dalam kehidupan hingga aku bisa menjadi seperti sekarang. Untuk keponakanku juga Salwa Wardatun Shalihah yang sering bermain denganku saat di rumah. Semoga kelak kamu menjadi pribadi yang berakhlak baik dan shalihah yaa"

Seluruh Keluarga Besar Teknik Informatika 2019

Program Studi Teknik Informatika, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Lampung

# UCAPAN TERIMA KASIH

Segala puji bagi Allah Subhanahu wa Ta'ala yang telah memberikan rahmat, hidayah dan karunia-Nya sehingga penulis dapat melaksanakan dan menyelesaikan penelitian ini yang berjudul "Pengembangan *RESTful API* dan Website Admin untuk Aplikasi *Screening Speech Delay*". Selama masa penelitian penulis mendapatkan banyak bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih sebanyak-banyaknya kepada:

1. Allah Subhanahu wa Ta'ala yang senantiasa memberikan kemudahan dan kelancaran kepada penulis serta Rasulullah Muhammad Shallallahu 'Alaihi wa Sallam yang telah menjadi suri tauladan yang baik bagi penulis;
2. Ibu dan Ayah serta keluarga penulis yang selalu memberikan motivasi, do'a dan dukungan kepada penulis;
3. Bapak Dr. Eng. Helmy Fitriawan, S.T., M.Sc. selaku Dekan Fakultas Teknik Universitas Lampung;
4. Ibu Herlinawati, S.T., M.T. selaku Ketua Jurusan Teknik Elektro Universitas Lampung;
5. Bapak Mona Arif Muda, S.T., M.T. selaku Ketua Program Studi Teknik Informatika Universitas Lampung dan telah membantu proses kelancaran pengerjaan penelitian;
6. Bapak Ing. Hery Dian Septama, S.T. selaku Pembimbing Akademik yang telah memberikan bimbingan selama perkuliahan di setiap semester dan selalu memberikan motivasi serta telah bersedia menjadi Pembimbing Utama pada penelitian penulis;
7. Bapak Puput Budi Wintoro, S.Kom., M.T.I. selaku Pembimbing Pendamping penelitian yang selalu memberikan motivasi dan memberikan bimbingan kepada penulis untuk menjadi lebih baik;

8. Ibu Yessi Mulyani, S.T., M.T. selaku Penguji penelitian yang telah banyak memberikan saran dan masukan;
9. Mbak Rika selaku Admin Program Studi Teknik Informatika yang telah banyak membantu penulis dalam segala urusan administrasi selama perkuliahan;
10. Seluruh dosen dan staf Jurusan Teknik Informatika Unila yang memberi masukan dan mempermudah proses pembuatan skripsi ini;
11. Teman-teman seperjuangan skripsi khususnya Selvia Eldina, Reistha Ramadhanty, Yovanta Anjelina, Meilika Dwi Putri, Dwi Liliyawati, Alfiyah Widyaningsih dan Husniatun Aini;
12. Teman-teman grup Jamaica, LinkedIn, KKN's, TI B 2019, RQM3, Enjoyneer Girl, dan Sifa Wisa yang telah banyak memberikan saran, do'a, dukungan, dan bantuannya;
13. Teman-teman Teknik Informatika 2019 yang selalu mendukung penulis;
14. Semua pihak yang turut serta dalam membantu menyelesaikan penelitian dan tidak mungkin penulis sebutkan satu persatu.

Penulis berharap bahwa laporan ini dapat menjadi sumber acuan yang berguna untuk kemajuan pengetahuan dalam bidang teknik informatika. Dengan demikian, diharapkan bahwa penelitian ini memberikan manfaat kepada para pembacanya.

Bandar Lampung, 7 Agustus 2023

Penulis,

Silvia Naim

# DAFTAR ISI

Halaman

# DAFTAR GAMBAR

Halaman

# DAFTAR TABEL

Halaman

# I. PENDAHULUAN

## 1.1 Latar Belakang

*Speech delay* merupakan suatu keadaan terlambat berbicara yang ditandai dengan kemampuan berbicara anak di bawah rata-rata dibandingkan dengan anak lainnya [1]. Gangguan keterlambatan bicara ini menimbulkan dampak lain bagi anak dalam mengembangkan keterampilan sosialnya dan ketika membangun hubungan sosial dengan orang lain [2]. Gangguan tersebut dialami oleh sebagian anak kecil yang usianya masih relatif balita [3].

Menurut Komdat Kesmas, Bidang Gizi dan Kesehatan Keluarga, Dinas Kesehatan Provinsi Lampung, pada tahun 2020 ditemukan sejumlah anak berusia 28 hari hingga 1 tahun yang menjalani terapi wicara di Provinsi Lampung, terdiri dari 15 anak laki-laki dan 2 anak perempuan, serta berumur 1 tahun ke atas. Pada usia 4 tahun, ada 37 laki-laki dan 18 perempuan. Menurut data penelitian dari Rumah Terapi Al-birru di Bandar Lampung pada tahun 2022, sebanyak 96 anak didiagnosis dengan keterlambatan bicara dan mendapatkan terapi wicara. Tidak ada data yang tepat dan pasti. *American Academy of Pediatrics* (AAP) telah menerbitkan pedoman klinis yang merekomendasikan skrining perkembangan bayi di bawah usia 36 bulan dengan tes skrining standar. Pedoman tersebut merekomendasikan tes skrining untuk perkembangan anak pada usia 9, 18, 24 atau 30 bulan, termasuk perkembangan bahasa. [4] guna mengidentifikasi kemungkinan adanya masalah dalam perkembangan bicara dan bahasa pada anak secara dini. Hal ini dapat membantu memulai intervensi lebih cepat, sehingga dapat meningkatkan kemungkinan keberhasilan perawatan dan membantu anak untuk mencapai potensi bicara dan bahasa dengan optimal.

Peran orang tua sangat penting dalam menunjang kemampuan bahasa dan bicara anak. Sayangnya, pengetahuan tentang perkembangan ini tidak dimiliki oleh semua orang tua. Apabila keterlambatan bicara pada anak tidak diidentifikasi dengan cepat, akan sulit mengubah kebiasaan bicara yang sudah ada. Upaya skrining atau pendeteksian dini keterlambatan bicara menjadi kunci utama dalam mengatasi situasi ini. Sayangnya, skrining ini umumnya dilakukan oleh terapis wicara atau dokter spesialis anak, yang memerlukan investasi besar dalam hal waktu, jarak, dan biaya.

Pada era teknologi digital, penggunaan aplikasi berbasis teknologi dapat membantu memfasilitasi deteksi dini *speech delay* dengan biaya yang lebih terjangkau dan mudah digunakan oleh orang tua. Oleh karena itu, dicetuskannya "*Birrussment*" (Al-Birru Assessment) yaitu aplikasi berbasis web untuk deteksi dini / *screening speech delay*. Untuk mengimplementasikan aplikasi *screening speech delay* tersebut, diperlukan pengembangan *Aplication Programming Interface* (API) dengan arsitektur *RESTful API* sebagai alat untuk komunikasi antara berbagai sumber daya atau komponen perangkat lunak yang berbeda. Selain *RESTful API*, perlu juga Website Admin untuk mengelola Aplikasi *Screening Speech delay*.

Dalam rangka memudahkan pengembangan teknologi untuk deteksi dini *speech delay*, maka pada skripsi ini penulis akan berfokus terhadap implementasi *RESTful API* (*backend development*) dan Website Admin (*front-end website development*). *Application Programming Interface* (API) adalah antamuka yang digunakan untuk mengakses aplikasi atau layanan dari sebuah program. *API* memungkinkan pengembang untuk memakai fungsi yang sudah ada dari aplikasi lain sehingga tidak perlu membuat ulang dari awal [5]. REST merupakan seperangkat prinsip arsitektur yang melakukan transmisi data melalui antarmuka yang terstandarisasi seperti *HTTP*. *REST API* bekerja layaknya seperti aplikasi web biasa. Client dapat mengirimkan permintaan kepada server melalui protokol *HTTP* dan kemudian server memberikan *response* balik kepada client [6]. Sebuah layanan web yang memenuhi semua persyaratan dan ketentuan arsitektur REST disebut layanan web RESTful [7]. *RESTful API* pada penelitian ini dibangun menggunakan node js.

*Node.JS* adalah lingkungan runtime *JavaScript* sisi server *asynchronous open-source* berdasarkan *JavaScript Engine Chrome* V8 yang sama yang digunakan di browser Chrome populer[8]. Dengan adanya Pengembangan *RESTful API*, dari sisi *front-end* akan mudah untuk melakukan pendaftaran akun (*register*), masuk akun (*login*), dan proses Create, Read, Update dan Delete (CRUD) untuk mengakses sumber daya yang dibutuhkan aplikasi *screening speech delay* berupa pertanyaan *screening* (question) maupun hasil *screening* pengguna (result).

Selain *RESTful API*, penulis mengembangkan Website Admin yang dibangun dengan menggunakan teknologi *React.JS*, sebuah *library* yang cukup populer dengan ekosistem yang besar untuk membuat *interface* interaktif yang menarik. Dengan salah satu kekuatannya berbasis komponen, *React.JS* ada di sini untuk membantu pengembang memecah bagian aplikasi mereka menjadi komponen yang lebih kecil dan dapat digunakan kembali. Website Admin difokuskan untuk efisiensi administrator pada aplikasi deteksi dini *speech delay* dalam pengelolaan pengguna, pertanyaan Assessment, dan hasil Assessment dari aplikasi deteksi dini *speech delay*.

## 1.2 Perumusan Masalah

Berdasarkan latar belakang, kajian masalah yang mendasari penelitian ini adalah:

1. Bagaimana cara memudahkan *Front-end* Developer untuk mengakses sumber daya dari *backend* server serta *database* tanpa perlu memiliki akses langsung ke sumber daya?
2. Bagaimana cara mengembangkan *RESTful API* dan Website Admin untuk aplikasi deteksi dini *speech delay* dalam rangka mengatasi keterbatasan pengetahuan orang tua dalam memantau perkembangan bahasa dan bicara anak?
3. Bagaimana cara mengembangkan *RESTful API* dan Website Admin untuk aplikasi deteksi dini *speech delay* dalam mengatasi keterbatasan waktu, jarak, dan biaya untuk *screening speech delay* ke ahli terapi bicara atau dokter tumbuh kembang anak?

4. Bagaimana cara mengatasi kesulitan administrator mengelola informasi pengguna, hasil test dan kegiatan administratif lainnya pada aplikasi *screening speech delay*?

## 1.3 Tujuan Penelitian

Tujuan penelitian ini adalah:

1. Mengembangkan *RESTful API* untuk Aplikasi *Screening Speech delay* agar dapat mengakses sumber daya dari *backend* server serta *database* tanpa perlu memiliki akses langsung ke sumber daya.
2. Mengembangkan *RESTful API* dan Website Admin untuk aplikasi deteksi dini *speech delay* agar menambah pengetahuan orang tua dalam memantau perkembangan bahasa dan bicara anak.
3. Mengembangkan *RESTful API* dan Website Admin untuk aplikasi deteksi dini *speech delay* yang dapat *screening speech delay* anak tanpa terhambat waktu, jarak, dan biaya.
4. Mengembangkan Website Admin untuk memudahkan administrator melacak data pengguna, hasil test dan perubahan skor Assessment anak.

## 1.4 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan beberapa manfaat sebagai berikut:

**Bagi Penulis**

Meningkatkan pemahaman tentang pengembangan *web Service RESTful API*, melakukan pemodelan hingga pengaplikasian pembuatan *API* yang mampu menghasilkan hasil diagnosa *speech delay* dan bermanfaat bagi orang tua dan profesional medis.

**Bagi *Front End* Developer**

Mempermudah *front-end* developer dalam mengembangkan aplikasi dengan menyediakan *RESTful API* yang terdokumentasi dengan baik untuk diintegrasikan dengan aplikasi *Screening Speech delay*.

**Bagi Pengguna Aplikasi**

Meningkatkan aksebilitas deteksi dini *speech delay* bagi orang tua dan profesional medis dimanapun.

**Bagi Administrator**

Mempermudah administrator untuk pengelolaan data pengguna, melacak kemajuan, dan membuat laporan untuk pengambilan keputusan berdasarkan informasi terkait hasil *screening*.

**Bagi Partner (Rumah Terapi Al – Birru)**

Dapat meningkatkan efisiensi dengan aplikasi *Screening Speech delay* untuk client dan menyebarluaskan nama klinik Rumah Terapi Al Birru. Rumah Terapi Al Birru merupakan partner yang akan membantu penyusunan teknis Assessment pada aplikasi *screening speech delay*.

## 1.5 Batasan Masalah

Dalam penelitian ini, pembatasan masalah meliputi hal-hal sebagai berikut:

1. Sistem ini hanya bersifat membantu proses Assessment atau *screening* bukan keputusan akhir, semua keputusan tetap berada pada ahli terapi bicara atau dokter tumbuh kembang anak karena tetap dibutuhkan pengamatan secara berkelanjutan.
2. Penelitian berfokus pada pengembangan *RESTful API* dan Website Admin untuk aplikasi *screening* atau deteksi dini *speech delay*.
3. Pembuatan *RESTful API* menggunakan teknologi dan tools seperti *Node.JS*, *Express*, dan *MongoDB* sedangkan Website Admin menggunakan teknologi *React.JS*.

## 1.6 Sistematika Penulisan Tugas Akhir

Sistematika penulisan skripsi akhir ini terdiri dari 5 (lima) bab sebagai berikut:

**BAB I : PENDAHULUAN**

Bab ini memberikan gambaran umum mengenai konteks penelitian, rumusan masalah, tujuan penelitian, manfaat penelitian, keterbatasan penelitian dan sistematika penulisan.

**BAB II : TINJAUAN PUSTAKA**

Bab ini membahas teori dasar yang digunakan sebagai sumber untuk memahami masalah dalam melakukan penelitian pengembangan *RESTful API* dan administrasi web untuk aplikasi *speech delay*.

**BAB III : METODE PENELITIAN**

Bab ini membahas metodologi penelitian yang digunakan dalam pengembangan *RESTful API* dan administrasi web untuk aplikasi *Speech delay*

**BAB IV : HASIL DAN PEMBAHASAN**

Bab ini menyajikan hasil dan pembahasan yang diperoleh dalam kerangka penelitian. *RESTful API* dan Website Admin Untuk Aplikasi *Speech delay*

**BAB V : SIMPULAN DAN SARAN**

Bab ini memuat kesimpulan atas hasil penelitian yang dilakukan serta saran untuk penelitian selanjutnya.

**DAFTAR PUSTAKA**

**LAMPIRAN**

# II. TINJAUAN PUSTAKA

## 2.1 Speech delay

Keterlambatan bicara atau *speech delay* adalah keterlambatan bicara yang ditandai dengan kemampuan bicara seorang anak di bawah rata-rata dibandingkan dengan anak lainnya [1]. Tanda-tanda anak mengalami keterlambatan perkembangan bahasa, yaitu perbendaharaan kata yang lebih sedikit daripada anak seusianya, pengucapan yang buruk, dan masalah penyesuaian psikososial [9]. Keterbelakangan bicara dapat menjadi gejala dari banyak gangguan yang berbeda termasuk keterbelakangan mental, gangguan pendengaran, gangguan bahasa ekspresif, kurangnya stimulasi psikososial, autisme, bisu selektif, afasia reseptif dan kelumpuhan otak [10].

## 2.2 Screening

*Screening* atau Skrining perkembangan adalah proses untuk menentukan apakah mereka memerlukan evaluasi lebih lanjut. Skrining perkembangan harus menggunakan alat atau instrumen yang dapat diandalkan dan menilai semua bidang perkembangan, yaitu keterampilan motorik halus dan kasar, bahasa, aspek kognitif dan aspek sosial asosiasi individu.

*American Academy of Pediatrics* (AAP) telah menerbitkan pedoman klinis yang merekomendasikan skrining perkembangan anak di bawah 36 bulan dengan tes skrining standar. Pedoman tersebut merekomendasikan tes skrining untuk perkembangan anak pada usia 9, 18, 24 atau 30 bulan, termasuk perkembangan bahasa. [4] guna mengidentifikasi kemungkinan adanya masalah dalam perkembangan bicara dan bahasa pada anak secara dini. Hal ini dapat membantu memulai intervensi lebih awal, meningkatkan kemungkinan pengobatan yang

berhasil, dan membantu anak-anak mencapai potensi bicara dan bahasa yang optimal.

*Screening* pada *speech delay* merujuk proses evaluasi awal untuk mengidentifikasi kemungkinan adanya keterlambatan perkembangan bicara pada seorang anak. Tujuannya adalah untuk mengidentifikasi anak-anak yang membutuhkan penilaian lebih lanjut atau intervensi lebih lanjut. Aplikasi Birrussment menggunakan *resource* pertanyaan *screeening* berdasarkan Rumah Terapi Al-birru yang menerapkan metode *screening* menggunakan form *Assessment of Language Development* dari buku *ASSESSMENT in Speech-Language Pathology* [11].

## 2.3 RESTful API

REST atau *REpresentational State Transfer* adalah seperangkat prinsip arsitektur yang mengirimkan data melalui antarmuka standar seperti *HTTP* sementara *Application Programming Interfaces* (API) adalah antarmuka yang digunakan untuk mengakses aplikasi atau layanan dari suatu program. *API* memungkinkan pengembang untuk menggunakan fungsionalitas yang ada dari aplikasi lain, menghilangkan kebutuhan untuk membangun kembali dari awal [5].

*REST API* berfungsi seperti aplikasi web biasa. client dapat mengirimkan permintaan ke server melalui protokol *HTTP*, kemudian server akan merespon kembali ke klien [6]. Dengan menggunakan *REST API*, developer dapat membuat endpoint atau titik akhir yang dapat diakses melalui *HTTP* oleh aplikasi atau layanan lain. *API* ini dapat digunakan untuk mengambil data dari sumber eksternal, mengirim data, melakukan operasi CRUD (Create, Read, Update dan Delete). Pada dasarnya, tujuan dari *REST API* adalah untuk memfasilitasi komunikasi antar sistem untuk berbagi dan mengakses data dengan cara yang terstandarisasi. Sistem yang memenuhi *REST* disebut sistem *RESTful*. Layanan web yang memenuhi semua syarat dan ketentuan arsitektur *REST* disebut layanan web *RESTful* [7].
Rincian berikut tentang penggunaan metode *HTTP*.

Tabel 2.1. Rincian penggunaan *HTTP methods*

| *HTTP Method* | Operasi | *Collection Resource* (e.g. */users*) | *Single Resource* (e.g. */users*/123) |
|---|---|---|---|
| *POST* | Create | 201 (Dibuat), tajuk 'Lokasi' dengan tautan ke /users/{id} yang berisi ID baru | Hindari menggunakan POST pada satu sumber daya |
| *GET* | Read | 200 (OK), daftar pengguna. Gunakan penomoran halaman, penyortiran, dan pemfilteran untuk menavigasi daftar besar | 200 (OK), pengguna tunggal. 404 (Tidak Ditemukan), jika ID tidak ditemukan atau tidak Valid |
| *PUT* | Update /Replace | 405 (Metode tidak diizinkan), kecuali jika Anda ingin memperbarui setiap sumber daya di seluruh kumpulan sumber daya | 200 (OK) or 204 (No Content). Use 404 (*Not Found*), jika ID tidak ada atau Invalid |
| *PATCH* | Partial Update / Modify | 405 (Metode tidak diizinkan), kecuali jika Anda ingin mengubah koleksi itu sendiri | 200 (OK) atau 204 (Tanpa Konten). Gunakan 404 (*Not Found*), jika ID tidak ditemukan atau tidak Valid |
| *DELETE* | Delete | 405 (Metode tidak diizinkan), kecuali Anda ingin menghapus seluruh koleksi — gunakan dengan hati-hati | 200 (baik). 404 (Tidak Ditemukan), jika ID tidak ditemukan atau tidak Valid |

## 2.4 Node Js

*Node.JS* adalah *open-source*, lingkungan *runtime JavaScript* lintas platform berdasarkan *engine JavaScript* V8 Chrome. Perangkat lunak ini memungkinkan pengembang untuk menulis perangkat lunak seperti alat baris perintah dan skrip sisi server dalam bahasa pemrograman *JavaScript* di luar lingkungan browser [12]. Dengan kata lain, *Node.JS* memberikan kemampuan kepada pengembang untuk menjalankan kode *JavaScript* pada sisi server (*server-side*) tanpa harus terbatas pada lingkungan browser saja.

*Node.JS* sendiri diperkenalkan pada tahun 2009 oleh Ryan Dahl dan berkembang pesat sejak dimasukkan ke dalam JSConf pada awal tahun 2009. Sebelum dikenal luas, *JavaScript* merupakan bahasa pemrograman yang banyak digunakan untuk membuat aplikasi web *client-side*. dan masih banyak digunakan sampai sekarang. Umumnya, aplikasi server menggunakan bahasa *PHP, Java, Ruby* atau *Python* yang sebelumnya dikenal, sedangkan bahasa *JavaScript* lebih umum digunakan untuk mengembangkan aplikasi sisi browser untuk membuat aplikasi lebih interaktif. Pada dasarnya, *Node.JS* adalah *JavaScript* yang berjalan di sisi server. Salah satu kekuatan *Node.JS* adalah teknik *non-blocking*. *Node.JS* sendiri memiliki keunggulan sebagai teknik *non-blocking* yang memungkinkan sistem untuk mengeksekusi operasi secara paralel tanpa menunggu operasi sebelumnya selesai, sehingga memungkinkan beberapa permintaan diproses secara paralel [13].

### 2.4.1 NPM (Node *Package* Manager)

NPM adalah singkatan dari *Node Package Manager*, yaitu pengelola *package* yang memungkinkan developer *JavaScript* menemukan dan menginstal *package* kode ke aplikasi jaringan atau *server-side*. NPM adalah registri perangkat lunak terbesar di dunia. Pengembang *open-source* di setiap benua menggunakan npm untuk berbagi dan meminjam paket, dan banyak organisasi juga menggunakan npm untuk mengelola pengembangan pribadi. NMP terdiri dari tiga komponen yang berbeda [14], meliputi:

a. Situs web: menggunakan situs web untuk menemukan *package*, menyiapkan profil, dan mengelola aspek lain dari pengalaman npm oleh user. Misalnya, User dapat menyiapkan organisasi untuk mengelola akses ke *package* publik atau privat.
b. *Command Line Interface* (CLI): CLI berjalan dari terminal, dan merupakan cara sebagian besar pengembang berinteraksi dengan npm.
c. Registri: Registri adalah *database* publik yang besar dari perangkat lunak *JavaScript* dan informasi meta yang mengelilinginya.

### 2.4.2 *Express* JS

*Express*, dengan mengacu pada *tagline*-nya, "*Fast, unopinionated, minimalist web framework for Node.JS*" [15] , adalah *framework* unggulan dan populer di ekosistem *Node.JS* karena kecepatannya, terlepas dari perspektif tertentu dan sifat minimalisnya.

*Express.JS* dianggap cepat karena *routing* yang efisien, kemampuan I/O *asynchronous*, modularitas, dukungan *middleware*, dan dukungan komunitas yang kuat. memiliki mekanisme perutean yang sederhana dan efisien yang memungkinkan pengembang dengan mudah menentukan rute dan menangani permintaan *HTTP*. Mekanisme perutean ini didasarkan pada penggunaan fungsi *middleware* berantai untuk menangani permintaan dan tanggapan.

*Express* tidak memaksakan pendekatan atau kebijakan tertentu pada pengembangan aplikasi. Dalam hal ini, *Express* memberikan kebebasan kepada developer untuk memilih dan mengimplementasikan metode yang paling sesuai dengan kebutuhan mereka. Dengan kata lain, *Express* tidak memiliki preferensi atau pandangan kaku tentang bagaimana pengembangan aplikasi *Node.JS* harus diikuti.

*Express.JS* adalah *framework* web minimalis untuk *Node.JS*. Alasan *Express.JS* dianggap minimalis adalah karena hanya menyediakan fungsionalitas dasar yang diperlukan untuk membangun aplikasi web, seperti *routing, middleware,* dan pengaturan tampilan. Oleh karena itu, pengguna dapat memilih modul tambahan

yang ingin digunakan berdasarkan kebutuhan aplikasinya. Ini membuat *Express.JS* fleksibel dan mudah digunakan. Karena sifatnya yang minimalis, *Express.JS* juga menawarkan performa yang cepat dan bobot yang ringan. Ini membuatnya cocok untuk membangun aplikasi web skala kecil hingga menengah.

Dalam keseluruhan, alasan utama mengapa *Express.JS* dianggap minimalis adalah karena hanya menyediakan fungsionalitas dasar yang diperlukan untuk membangun aplikasi web, sehingga pengguna dapat memilih modul tambahan mana yang ingin digunakan berdasarkan kebutuhan aplikasinya. Selain itu, dokumentasi yang komprehensif dan kinerja yang cepat juga menjadi sorotan dari *framework* ini.

## 2.5 MongoDB

Pemilihan basis data merupakan salah satu hal penting dalam perancangan dan pengembangan aplikasi. Pemilihan basis data tidak terbatas pada fungsi penyimpanan data, tetapi juga harus mendukung keunggulan dan kinerja produk yang dikembangkan sehingga layanan yang diberikan dapat memuaskan pengguna. Selain itu, pemilihan *database* dapat mempengaruhi beberapa aspek lain seperti pengembangan situs web di masa mendatang, sumber daya manusia, keuangan, dan waktu pengembangan. [16].

### 2.5.1 *MongoDB* dikategorikan *NoSQL*

*Database NoSQL* (n*on SQL* atau *Not Only SQL*) dikembangkan pada akhir tahun 2000-an dengan fokus pada *scaling*, *fast queries*, memungkinkan perubahan aplikasi yang sering, dan membuat pemrograman menjadi lebih sederhana bagi pengembang. Sementara itu, basis data relasional yang dapat diakses dalam *SQL* (*Structured Query Language*) dikembangkan pada tahun 1970-an dengan fokus pada pengurangan duplikasi karena biaya penyimpanan data lebih tinggi daripada biaya pengembangan oleh pengembang perangkat lunak. *Database SQL* cenderung memiliki skema tabular yang kaku dan kompleks dan seringkali memerlukan

penskalaan vertikal yang mahal [17]. Tabel di bawah merangkum perbedaan utama antara *database SQL* dan *NoSQL* [17] :

Tabel 2.2. Perbedaan *database SQL* dan *NoSQL*

| Variabel Perbandingan | *Database SQL* | *Database NoSQL* |
|---|---|---|
| Model Penyimpanan Data | Tabel dengan baris dan kolom tetap | • *Document: JSON documents*,<br>• *Key-value*: *key-value pairs*,<br>• *Wide-column*: *tables with rows and dynamic columns*<br>• *Graph: nodes and edges* |
| Sejarah Pembangunan | Dikembangkan pada tahun 1970-an dengan fokus pada pengurangan duplikasi data | Dikembangkan pada akhir tahun 2000-an dengan fokus pada *scalling* dan memungkinkan perubahan aplikasi yang cepat didorong oleh praktik *Agile* dan *DevOps*. |
| Contoh | Oracle, My*SQL*, Microsoft *SQL* Server, dan Postgre*SQL* | • *Document*: *MongoDB* and CouchDB,<br>• *Key-value*: Redis and DynamoDB,<br>• *Wide-column*: Cassandra and HBase,<br>• *Graph*: Neo4j and Amazon Neptune |
| Skema | Kaku | Fleksibel |
| *Scaling* | *Vertical* | *Horizontal* (peningkatan di seluruh server komoditas) |
| *Multi-record* ACID *Transactions* | Didikung | Kebanyakan tidak mendukung transaksi ACID *multi-record*. Namun, beberapa seperti *MongoDB* mendukung ACID. |
| *Joins* (Penggabungan) | Biasanya diperlukan | Umumnya tidak diperlukan |
| Pemetaan data ke *object* | Membutuhkan ORM (pemetaan objek-relasional) | Banyak orang tidak membutuhkan ORM. Dokumen *MongoDB*<br>langsung dipetakan ke struktur data dalam bahasa pemrograman paling populer. |

## 2.6 React.JS

*React.JS* adalah pustaka *JavaScript* yang dirancang khusus untuk membuat antarmuka pengguna (UI) untuk aplikasi web dan seluler (asli). *React.JS* berusaha untuk kecepatan, kesederhanaan dan skalabilitas [18]. Berikut ini konsep dan istilah penting pada *React.JS* :

a. *JSX*

*JSX* adalah ekstensi sintaksis untuk *JavaScript* yang memungkinkan pengembang menulis *markup* mirip *HTML*s dalam file *JavaScript*. Meskipun ada cara lain untuk menulis komponen, sebagian besar pengembang *React.JS* lebih menyukai singkatnya *JSX* dan sebagian besar basis kode menggunakannya. *JSX* dan *React.JS* adalah dua hal yang terpisah. Mereka sering digunakan bersama, tetapi pengembang dapat menggunakannya secara terpisah satu sama lain. *JSX* adalah ekstensi sintaksis, sedangkan *React.JS* adalah pustaka *JavaScript*. Setiap komponen *React.JS* adalah fungsi *JavaScript* yang dapat berisi *markup* yang dirender *React.JS* di browser. Komponen *React.JS* menggunakan ekstensi sintaks yang disebut *JSX* untuk merepresentasikan *markup* ini. *JSX* sangat mirip dengan *HTML*, tetapi sedikit lebih ketat dan dapat menampilkan informasi yang dinamis. Cara terbaik untuk memahami ini adalah dengan mengubah *markup HTML* menjadi *markup JSX* [18].

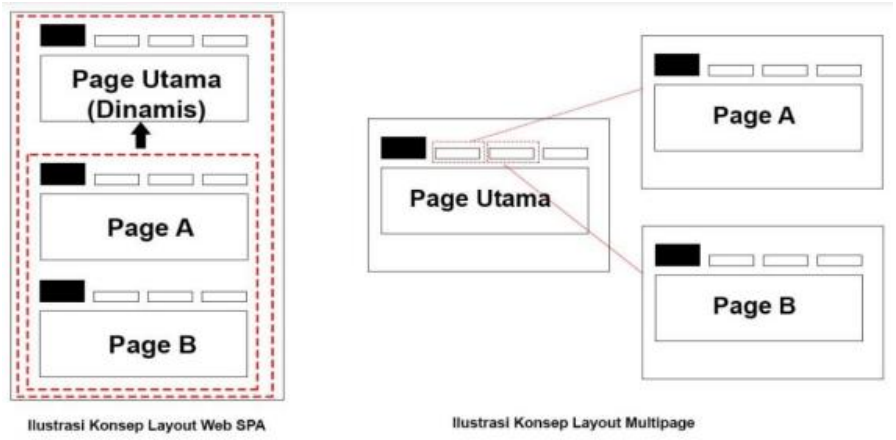
*b.* *React.JS Component*

Komponen *React.JS* menerima data dan mengembalikan apa yang akan muncul di layar. Developer dapat meneruskan data baru kepada mereka sebagai *response* terhadap interaksi, seperti saat pengguna memasukkan masukan. *React.JS* kemudian akan memperbarui tampilan agar sesuai dengan data baru. Pengembang tidak perlu membangun seluruh halaman mereka di *React.JS*. Tambahkan *React.JS* ke halaman *HTML* yang ada dan jadikan komponen *React.JS* interaktif di mana pun di dalamnya [18].

c. SPA

*Single Page Application* (SPA) atau Aplikasi Halaman Tunggal adalah aplikasi web yang sesuai pada satu halaman web dengan tindakan dinamis tanpa perlu

menyegarkan halaman (*refresh page*). Interaksi aplikasi pada halaman dapat ditangani tanpa server dan dapat meningkatkan performa dalam beberapa cara, seperti waktu muat, penggunaan AJAX, navigasi halaman yang mudah, dan lainnya. Pengguna akhir akan merasa lebih nyaman karena sangat mudah menelusuri atau menavigasi berbagai situs web dan filter konten [19]. Jika dilihat pada ilustrasi di bawah ini menjelaskan bahwa teknik tersebut sedikit berbeda dengan teknik edit web yang ada pada umumnya, dimana pembuatan halaman web dengan menggunakan teknik tradisional dilakukan dengan cara tertentu yang benar-benar terpisah, yang akan menyebabkan browser menjadi merender halaman web. halaman memakan waktu lama. cukup lama, karena halaman akan ditampilkan setiap kali halaman dimuat [20].

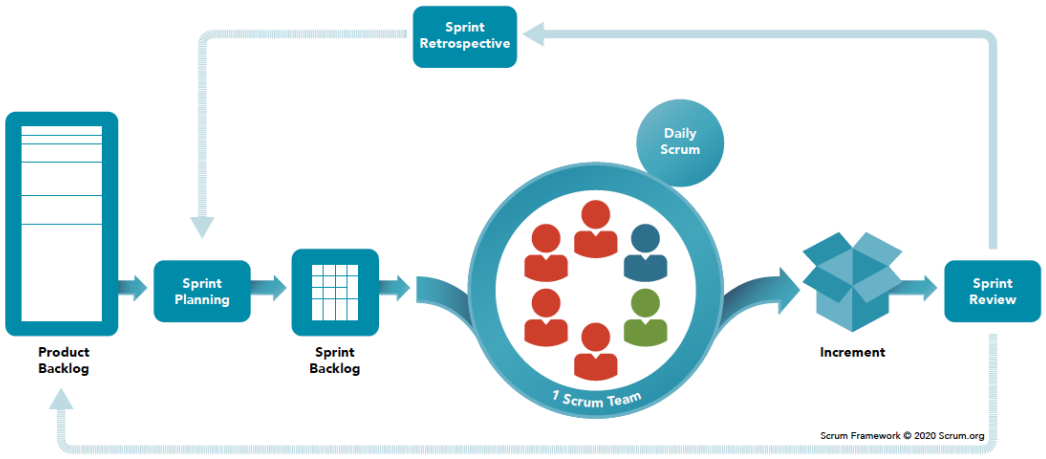



Gambar 2.1. Illustrasi *Single Page Application* (SPA).

## 2.7 Scrum

Metode *Scrum* merupakan salah satu yang termasuk dalam proses pengembangan perangkat lunak *agile*. *Scrum* dinilai memiliki kemampuan untuk menciptakan perangkat lunak berkualitas baik yang sesuai dengan harapan pengguna, dapat digunakan dalam proyek besar maupun kecil, serta mudah diterapkan. Tahapan kegiatan *Scrum* meliputi *Product Backlog*, *Sprint Backlog*, *Daily Scrum*, *Sprint Review*, dan *Sprint Retrospective*. Peran dalam *Scrum* meliputi *Product Owner*, *Scrum Master*, dan Developer. *Scrum* memiliki fase-fase yang terstruktur dan

bersifat iteratif, sehingga jika produk dari *Sprint* pertama tidak memenuhi kebutuhan, sistem dapat dikembangkan di *Sprint* berikutnya, sesuai penilaian pengguna [21]. Adapun Siklus Metode *Agile* dengan *Framework Scrum* adalah sebagai berikut [22] :




Gambar 2.2. Siklus Metode *Agile* dengan *Framework Scrum*

### 2.7.1 *Scrum* Team

Tim *Scrum* terdiri dari beberapa anggota yang bekerja sama untuk mencapai tujuan proyek secara kolaboratif dan adaptif. Tim *Scrum* terdiri dari *Scrum Master*, *Product Owner*, dan pengembang. Saat membentuk tim *Scrum*, penting untuk memahami peran kunci dalam *Scrum*. Peran utama dalam *Scrum* adalah :

1. Developers

   Developers yang dimaksudkan dalam penelitian ini yaitu *front-end* developer dan *backend* developer. Developers harus bertanggung jawab untuk:

   a. Perencanaan *Sprint* dan *Sprint Backlog*,
   b. Menanamkan mutu dengan mengikuti Definisi Selesai. Tujuannya adalah untuk memastikan bahwa setiap elemen produk yang dikembangkan oleh tim telah memenuhi standar kualitas yang telah ditetapkan.
   c. Sesuaikan rencana mereka setiap hari untuk mencapai tujuan *Sprint*.
   d. Tanggung jawab bersama sebagai seorang profesional.

2. *Product Owner*

   *Product Owner* atau *Product Owner* bertanggung jawab untuk memahami dan menjelaskan kebutuhan pengguna atau pelanggan dari produk yang sedang dikembangkan dan memprioritaskan *backlog*.

3. *Scrum Master*

   Bertindak sebagai pemimpin tim membantu memfasilitasi dan mempertahankan kepatuhan terhadap prinsip-prinsip *Scrum*. *Scrum Master* bertanggung jawab untuk :

   a. Memastikan bahwa semua anggota tim memiliki pemahaman yang jelas tentang peran, proses, dan praktik *Scrum*.
   b. *Scrum Master* membantu tim dalam merencanakan, mengatur, dan mengelola *Sprint*. Mereka membantu menentukan tujuan dan ruang lingkup *Sprint*, dan memastikan bahwa *backlog Sprint* diatur dengan benar dan siap digunakan.
   c. Koordinasi komunikasi kelompok
   d. Memfasilitasi berdiri setiap hari.
   e. Memfasilitasi *Sprint Retrospective*.

### 2.7.2 Fase-fase *Scrum*

Sesuai dengan metode pengembangan perangkat lunak *Scrum*, maka terdapat fase-fase *Scrum* yang terdiri dari :

a. *Sprint*

   Inilah inti dari *Scrum*, yaitu kerangka waktu sebulan atau kurang untuk menghasilkan produk yang diklaim lengkap dan mencakup perencanaan *Sprint*, *Scrum* harian, *review Sprint* dan peningkatan *Sprint forward*. Bahkan, *Product Owner* bisa membatalkan *Sprint* karena alasan tertentu, seperti perubahan peraturan atau kondisi tertentu.

b. *Sprint Planning*

   Kegiatan perencanaan untuk mengeksekusi *Sprint* dan menentukan daftar produk yang akan diproduksi. Rencana ini dibuat oleh kerja kolaboratif dari

seluruh developer. *Scrum Master* harus memastikan bahwa acara berlangsung dan para peserta memahami tujuannya.

c. *Daily Scrum*

Kegiatan diskusi yang diadakan oleh tim pengembang setiap hari selama *Sprint* dan berlangsung selama 15 menit. Kegiatan ini bertujuan untuk mengoptimalkan kolaborasi dan kinerja dengan menelaah pekerjaan kemarin dan menyusun rencana kerja untuk hari berikutnya.

d. *Sprint Review*

Ini adalah aktivitas yang diadakan di akhir *Sprint* untuk memeriksa peningkatan atau hasil dan menyesuaikan *Product Backlog* sesuai kebutuhan. Selama peninjauan *Sprint*, tim *Scrum* dan pemangku kepentingan berkolaborasi tentang apa yang telah dilakukan dalam *Sprint*. Berdasarkan hal ini dan setiap perubahan yang dibuat pada *Product Backlog* selama *Sprint*, para peserta berkolaborasi tentang apa yang selanjutnya dapat mereka lakukan untuk memaksimalkan nilai. Tinjauan *Sprint* dilakukan hingga empat jam untuk *Sprint* sebulan.

e. *Sprint Retrospective*

Ini adalah kesempatan bagi tim *Scrum* untuk mengintrospeksi diri dan membuat rencana perbaikan untuk diterapkan di *Sprint* berikutnya. Peningkatan *Sprint* terjadi setelah penilaian *Sprint* dan sebelum merencanakan *Sprint* berikutnya. Kegiatan ini dilakukan hingga tiga jam untuk *Sprint* sebulan.

### 2.7.3 Artifacts *Scrum*

*Artifacts* atau luaran yang dihasilkan pada setiap proses *Scrum* adalah sebagai berikut:

a. *Product Backlog*

Yaitu daftar semua fitur, fungsi, dan persyaratan produk yang ingin Anda hasilkan. *Product Backlog* dapat direvisi sebagai bentuk perbaikan di *Sprint* berikutnya. *Product Owner* bertanggung jawab untuk memprioritaskan *Product Backlog* ini.

b. *Sprint Backlog*

Yakni daftar *Product Backlog* yang dipilih untuk *Sprint* ditambah rencana untuk memproduksi produk tersebut dan mencapai tujuan *Sprint*

c. *Increment*

Yakni jumlah *item Product Backlog* yang diselesaikan dalam *Sprint* dan nilai total Increment di semua *Sprint* sebelumnya.

## 2.8 End-to-end (E2E) *Testing*

*End-to-end Testing* adalah metode yang digunakan untuk memeriksa apakah alur aplikasi berfungsi seperti yang diharapkan dari awal hingga akhir. Pengujian *end-to-end* merupakan salah satu teknik pengujian yang wajib dilakukan oleh perusahaan pengembang web karena jika telah lolos pengujian *end-to-end*, secara umum dianggap telah memastikan interaksi pengguna dengan situs web dan berada di sesuai dengan kebutuha pengguna [23].

### 2.8.1 *Black Box*

Pengujian *Black Box* adalah pengujian kualitas perangkat lunak yang berfokus pada fungsionalitas perangkat lunak. Pengujian *Black Box* bertujuan untuk menemukan fungsi yang salah, kesalahan antarmuka, kesalahan struktur data, kesalahan kinerja, kesalahan inisialisasi dan terminasi [24]. Metode *Black Box Testing* adalah metode yang digunakan untuk menguji perangkat lunak tanpa harus mengkhawatirkan detail perangkat lunak. Tes ini hanya memeriksa nilai output terhadap nilai *input* yang sesuai. Tidak perlu mencoba mencari tahu kode program apa yang digunakan output [25].

### 2.8.2 *Cypress*

*Cypress* adalah sebuah perangkat lunak pengujian otomatis yang dirancang khusus untuk pengujian aplikasi web. *Cypress* digunakan untuk menguji fungsionalitas, performa, dan keandalan aplikasi web dengan cara mengeksekusi serangkaian tindakan dan assersi pada halaman web yang sedang diuji [26].

*Cypress* menawarkan berbagai fitur yang kuat dan dapat membantu para pengembang dan QA *engineer* dalam mengotomatisasi pengujian aplikasi web. Beberapa fitur utama dari *Cypress* meliputi [26]:

a. Arsitektur yang terintegrasi: *Cypress* menggunakan arsitektur tertanam, yang artinya dapat mengontrol browser secara langsung. Ini memungkinkan *Cypress* untuk mengakses semua aspek aplikasi web dan berinteraksi langsung dengan komponen di dalam aplikasi tersebut.
b. Pemahaman dan penulisan kode yang sederhana: *Cypress* dirancang agar mudah dipahami dan digunakan oleh pengembang. Sintaks yang intuitif dan dokumentasi yang baik memudahkan pengguna untuk menulis dan memahami kode pengujian.
c. Mode waktu nyata (real-time): *Cypress* memberikan tampilan real-time dari setiap tindakan yang dilakukan di situs web. Dengan ini, pengguna dapat melihat perubahan waktu nyata pada tampilan dan perilaku aplikasi mereka selama pengujian. Ini memfasilitasi pemecahan masalah (debug) dan pengembangan aplikasi .
d. Penguji kaya fitur (*rich testing*): *Cypress* memiliki perpustakaan asli untuk pengujian dengan banyak fitur, termasuk fungsionalitas asli untuk menemukan elemen, mengatur status, melakukan tindakan, dan memverifikasi perilaku aplikasi. *Library* ini memfasilitasi pembuatan pengujian yang kuat dan terstruktur.

### 2.8.3 *Stress Test*

*Stress Testing* adalah cara untuk melihat bagaimana situasi rawan kegagalan mempengaruhi tingkat stres sistem. Pengujian akan dilakukan dengan variabel jumlah pengguna dalam satu waktu dan akan menguji respon sistem ketika diakses oleh pengguna dalam jumlah besar dan dalam jangka waktu tertentu [27]. Tujuan dari *Stress Testing* adalah untuk memperkirakan beban maksimum yang dapat ditangani oleh server web .

### 2.8.4 Apache JMeter

Apache JMeter adalah proyek sumber terbuka yang digunakan untuk alat pengujian kinerja dan beban. Apache JMeter akan membuat beberapa simulasi pengguna yang akan mengakses server dengan jumlah pengguna yang berbeda dan interval waktu permintaan tergantung pada konfigurasi yang dilakukan pada Apache JMeter [27].

## 2.9 Penelitian Terkait

Penelitian oleh Irfan Kurniawan, Humaira, dan Fazrol Rozi berjudul "*REST API* Menggunakan NodeJS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android" [6] berfokus pada pengembangan *REST API*. Penelitian ini mengatasi permasalahan penjualan dan penawaran jasa hanya dilakukan secara langsung, yang tentunya ini membuat konsumen merasa kurang berminat, karena harus menguras tenaga pergi ketempat orang yang buka jasa. Maka, dikembangkan sebuah *system* mengenai transaksi jasa dengan basis *Application Programming Interface* (API) sebagai *backend* dan diimplementasikan ke *mobile* android sebagai *front-end*. Penulis mengimplementasi dan melakukan pengujian pemesanan jasa Online, mulai dari pembuatan *backend* serta menerapkannnya ke *front-end* untuk aplikasi *mobile*. Teknologi yang digunakan penulis dalam membuat *REST API* berupa *Node.JS*, *sequalize*, dan *Authentication* dengan *JWT*. Penulis melakukan implementasi terhadap rancangan yang telah dibuat pada pembahasan sebelumnya. Mulai dari implementasi terhadap pengkodean sistem dalam segi *backend* dan aplikasi dari implementasi terhadap *design interface* yang telah di buat. Lalu tahap selanjutnya yaitu tahap pengujian terhadap sistem yang dibangun guna untuk memastikan sistem serta aplikasi yang dibangun telah sesuai dengan apa yang diharapkan.

Penelitian oleh Ahsan Mubariz, Dahlia Nur, Eddy Tungadi, dan Muhammad Nur Yasir Utomo berjudul "Perancangan *Back-End* Server Menggunakan Arsitektur REST dan *Plat*form *Node.JS* (Studi Kasus: Sistem Pendaftaran Ujian Masuk Politeknik Negeri Ujung Pandang)" [13] berfokus pada pembuatan *Back-End* Server Menggunakan Arsitektur REST. Penelitian ini mengatasi kendala pada waktu *response* saat banyaknya akses secara bersamaan pada sistem *existing* Ujian

Masuk Politeknik Negeri Ujung Pandang (UMPN). Untuk itu, pada penelitian tersebut *back-end* dari sistem UMPN dibangun ulang menggunakan arsitektur REST dan *plat*form *Node.JS*. *Node.JS* memiliki keunggulan pada teknik *non-blocking* yang memungkinkan operasi-operasi dijalankan secara paralel, sehingga memungkinkan banyak *request* dapat diselesaikan secara paralel. Fitur-fitur dari sistem yang dibangun telah diuji dan berjalan dengan kinerja yang baik. Adapun hasil uji kinerja dari sistem yang telah dibangun menunjukkan peningkatan waktu *response* pada kondisi pengujian dengan akses dari *virtual* user*s* yang meningkat dengan rata-rata peningkatan sebesar 52% pada skenario yang diujikan. Adapun sistem yang dibuat memiliki kinerja yang lebih baik dibandingkan sistem existing berdasarkan hasil uji dengan rata-rata selisih waktu *response* berdasarkan skenario pengujian sebesar 3222,5 ms.

Penelitian oleh Shon Hadji, M. Tufik, dan Sri Mulyono berjudul "Implementasi metode *Scrum* pada pengembangan aplikasi *delivery order* berbasis website (studi kasus pada rumah makan lombok idjo semarang)" [21] yang berfokus pada Metode *Scrum*. Penelitian ini banyak membahas bagaimana menerapkan *Scrum* untuk membangun aplikasi *delivery order* berbasis website. Hasil dari penelitian tersebut, metode *Scrum* dapat mengatasi perubahan *requirements* pada saat fase pengembangan sistem dan *Scrum* memiliki tahapan yang bersifat perulangan dimana jika produk pada *Sprint* pertama belum cukup memenuhi kebutuhan, maka pada *Sprint* berikutnya dapat dikembangkan sistem yang sesuai dengan evaluasi pengguna.

Penelitian oleh Asri Rahayu, Maydina Mutiara Kasih, Opy Safitri, dan Rina Nursanti berjudul "Penerapan aplikasi deteksi dini *speech delay* pada anak batita dengan menggunakan metode pengembangan kpsp berbasis android (skrianteng-*mobile*) di posyandu wilayah kerja puskesmas taman bacaan palembang" [28] yang berfokus pada efektifitas aplikasi deteksi dini *speech delay* berbasis android. Penelitian ini mengatasi kendala kurangnya informasi mengenai deteksi dini tumbuh kembang dari tenaga kesehatan dan kurangnya minat ibu untuk mendeteksi dini perkembangan anaknya. Hasil dari penelitian tersebut seluruh subjek penelitian

mengatakan aplikasi dapat memberikan manfaat yang baik dan sangat berguna bagi ibu-ibu yang memiliki anak mulai dari 0 sampai 72 bulan, untuk dapat mendeteksi dan mengetahui secara dini adanya kemungkinan penyimpangan perkembangan bicara dan bahasa pada anaknya secara mandiri dan optimal dirumah.

Penelitian oleh Satrio Krisna Murti dan Ari Sujarwo berjudul "Membangun Antarmuka Pengguna Menggunakan *React.JS* untuk Modul Manajemen Pengguna" [29] yang berfokus pada pembuatan Antarmuka (Website Admin) untuk Manajemen Pengguna. Penelitian ini membahas tentang pengembangan Modul Manajemen Pengguna di PT. Dua Empat Tujuh. Penulis mengungkapkan bahwa Modul ini akan dibuat mulai dari awal menggunakan *React.JS* untuk standarisasi *library* atau pustakanya karena pengembangan web sebelumnya belum menggunakan *React.JS*. Nantinya modul ini akan diaplikasikan untuk proyek proyek yang sudah ada. Dalam penelitian tersebut, penulis berfokus terhadap UX dan *Front-end* development dimana penulis menggunakan metodologi *design thinking* dan implementasi menggunakan *React.JS* dan *template* bootstrap Admin LTE.

Penelitian oleh Nasution berjudul "Perancangan Implementasi *MongoDB*, *Express* JS, *React.JS* dan Node JS (MERN) pada Pengembangan Aplikasi Formulir, Kuis, dan Survei Online" [16]. Penelitian ini dilatarbelakangi oleh masalah tidak adanya layanan yang mencangkup tiga fitur penting meliputi formulir, kuis dan survei. Untuk itu, pada penelitian tersebut mengembangkan layanan bernama Ubaform yaitu layanan pembuatan formulir, kuis dan survei pada penelitian berfokus terhadap implementasi teknologi MERN yang digunakan untuk pengembangan baik terhadap performa yang dihasilkan, kemudahan serta efisiensi waktu yang dihadirkan dari implementasi teknologi tersebut. Dalam penelitian tersebut penulis mengungkapkan bahwa 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.

Penelitian Oleh Ni Luh Ayu Sonia Ginasari, Kadek Suar Wibawa, dan Ni Kadek Ayu Wirdiani berjudul "Pengujian *Stress Testing API* Sistem Pelayanan dengan Apache JMeter" [27] yang berfokus pada implementasi *Stress Testing* dengan Apache JMeter untuk lebih mengoptimalkan sistem dan memeriksa ketahanan dari sistem. Penelitian tersebut membahas alur tahapan *Stress Testing* yang akan dilakukan dengan menggunakan Apache JMeter dan juga menunjukkan hasil dari *Stress Test* pada beberapa bidang *API* seperti laboratorium, bidang laboratorium, layanan laboratorium, berita laboratorium, dan pengujian uji sampel laboratorium menggunakan 25, 50, dan 75 sampel dengan periode *ramp up* adalah 10 detik dan *loop count* adalah 1. Hasil yang didapatkan pada pengujian adalah pada bagian rata-rata waktu pengujian *(Average*) mendapatkan nilai yang konstan, pada bagian *Throughput* mendapatkan nilai yang meningkat sebanding dengan jumlah *request* sampelnya, dan pada bagian *deviation* mendapatkan nilai yang menurun sehingga ini membuktikan bahwa sistem mempunyai ketahanan yang lumayan baik dimana hasil grafik nilai *Average*, *Throughput*, dan *deviation* mendapat nilai yang baik.

Berdasarkan studi terkait yang disajikan, dapat disimpulkan bahwa penggunaan teknologi modern seperti *REST API*, *Node.JS*, metodologi *Scrum*, *React.JS* dan MERN stack memiliki keuntungan yang signifikan dalam pengembangan aplikasi web. Beberapa hasil dapat disimpulkan bahwa Secara keseluruhan, penggunaan teknologi seperti *REST API*, *Node.JS*, metodologi *Scrum*, *React.JS* dan MERN stack berdampak positif pada pengembangan aplikasi. Itu dapat meningkatkan kinerja sistem, meningkatkan pengalaman pengguna dan mempercepat proses pengembangan. Studi-studi ini memberikan wawasan penting ke dalam pengembangan aplikasi web dan seluler yang melibatkan sistem aplikasi dan teknologi penyaringan keterlambatan bicara.

# III. METODOLOGI PENELITIAN

## 3.1 Waktu dan Tempat Penelitian

Waktu dan tempat pelaksanaan penelitian dilakukan pada :

1. Waktu penelitian : Februari 2023 sampai dengan Juli 2023
2. Tempat Penelitian : Rumah Terapi Al-Birru, Kelurahan Rajabasa Jaya, Kec. Rajabasa, Kota Bandar Lampung, Lampung

### 3.1.1 Jadwal Penelitian

Jadwal pada penelitian ini adalah sebagai berikut:

Tabel 3.1. Jadwal Penelitian

| No | Aktivitas | Feb 2023 | Mar 2023 | April 2023 | Mei 2023 | Juni 2023 | Juli 2023 |
|---|---|---|---|---|---|---|---|
| 1 | *User story* | | | | | | |
| 2 | *Product Backlog* | | | | | | |
| 3 | *Sprint* 1 | | | | | | |
| 4 | *Sprint 2* | | | | | | |
| 5 | *Sprint 3* | | | | | | |
| 6 | *Sprint 4* | | | | | | |
| 7 | *Sprint 5* | | | | | | |
| 8 | Penyusunan Laporan | | | | | | |

## 3.2 Alat Penelitian

### 3.2.1 Perangkat Lunak

Tabel 3.2. Perangkat lunak yang digunakan dalam penelitian

| No | Nama *Tool* | Deskripsi |
|---|---|---|
| 1. | Windows 11 Pro 64-bit (OS) | Sistem operasi windows digunakan untuk menjalankan beberapa perangkat lunak yang diperlukan untuk proses pengembangan . |
| 2. | *Visual Studio Code* | *Text Editor* memiliki banyak ekstensi atau plugin yang dapat digunakan untuk mengedit kode dalam berbagai bahasa pemrograman. |
| 3. | Postman | Perangkat lunak untuk menguji API. Post man mengirim permintaan *API* ke server web dan menerima *response*. |
| 4. | Swagger API | *Tool* untuk menyusun dokumentasi *RESTful API* secara otomatis. Dokumentasi diperlukan agar memudahkan pengembang *front-end* melihat endpoint API. |
| 5. | Chrome Web Browser | Digunakan untuk *preview* website dan berinteraksi dengan website tersebut. |
| 6. | Figma | Perangkat lunak pembuatan prototipe antarmuka web. |
| 7. | Trello | Perangkat lunak untuk berkolaborasi dan mengelola banyak proyek di satu tempat. |
| 8. | Git | Perangkat lunak untuk kolaborasi *code*, mengontrol versi *source code* dan terhubung dengan layanan *repositori code manager* seperti github untuk menyimpan repositori *source code*. |
| 9. | *Cyclic* | Layanan untuk platform server *hosting REST API* |
| 10. | Vercel | Layanan untuk platform *hosting* Website Admin |

| | | |
|---|---|---|
| 11. | *MongoDB Compass* | Perangkat lunak berbasis GUI untuk menanyakan, mengagregasi, dan menganalisis *database MongoDB* dalam lingkungan *visual*. *MongoDB Compass* juga dapat digunakan untuk mengakses basis data *cloud* jarak jauh pada layanan *MongoDB* Atlas. |

### 3.2.2 Perangkat Keras

Dalam penelitian ini perangkat keras yang digunakan adalah sebuah laptop dengan spesifikasi :

a) *Processor* Intel(R) Celeron(R) N4000 CPU @ 1.10GHz 1.10 GHz
b) *Installed* RAM 12,00 GB
c) *Storage* 256 GB SSD & 1 TB HDD

## 3.3 Pengumpulan Data

Pada penelitian ini ada sejumlah metode pengumpulan data yang dapat dilakukan, di antaranya adalah observasi dan wawancara. Berikut ini merupakan hasil dari observasi dan wawancara yang dilakukan di Rumah Terapi Al Birru :

a. Sistem *screening* secara *offline*

   Saat ini, sistem *screening* yang diterapkan di Klinik Rumah Terapi Albirru masih cara manual dan harus datang ke lokasi Klinik. *Screening* dilakukan oleh orang tua didampingi oleh terapis untuk melakukan *screening*. Sistemnya, orang tua mengisi formulir dengan menjawab pertanyaan berdasarkan rentang umur anak. Adapun untuk mengasilkan score dan analisa dari *screening*, terapis harus menghitungnya secara manual.

b. Kendala pada sistem *screening* secara *offline*

   Kendala yang dialami yaitu pelanggan waktu tidak fleksibel sehingga sulit menyesuaikan jadwal. Untuk menghasilkan score dan analisa harus menghitung secara manual, sehingga memungkinkan terjadinya *human error*.

c. Sistem yang diusulkan

Sistem yang diusulkan berdasarkan kendala yang dihadapi yaitu aplikasi *screening* berbasis online *bernama* "Birrussment (Al-Birru Assessment)". Sehingga dapat mengatasi kendala *human* error yang dihadapi dari pihak rumah terapis. Sistem ini dapat diakses melalui website dapat diakses dimana saja, selain itu pelanggan dapat melakukan *screening* dengan mengambil Assessment dari aplikasi. Pelanggan akan mendapatkan hasil diagnosa beserta analisa berdasarkan jawaban dari Assessment.

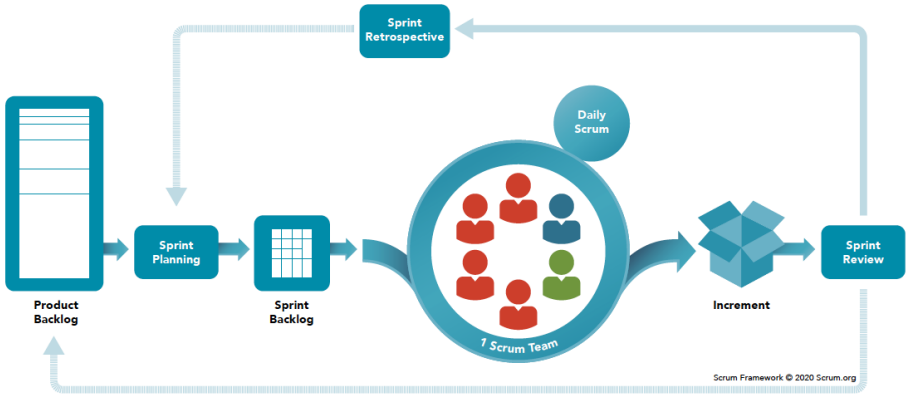
## 3.4 Tahapan Penelitian

Tahapan yang dilakukan dalam penelitian ini mengikuti model pengembangan perangkat lunak *Scrum*. Metode ini sesuai saat digunakan dalam pengembangan perangkat lunak secara tim karena dapat menjaga koordinasi yang baik antar masing-masing anggota tim yang memiliki berbagai tugas yang berbeda. Tim-tim yang terdapat pada penelitian ini adalah sebagai berikut.

Tabel 3.3. Tabel *role scrum* pengembangan aplikasi Birrussment

| Role | Nama | Keterangan |
|---|---|---|
| *Product Owner* | Ari Damayanti, S.Tr.Kes. | Direktur Rumah Terapi Al-birru.<br>Tugas dari *Product Owner* dalam penelitian ini adalah untuk memastikan bahwa developer aplikasi memenuhi tujuan bisnis dari proyek tersebut |
| *Scrum Master* | Selvia Eldina | *Scrum Master* bertanggung jawab untuk memperkenalkan dan mendukung implementasi Metode *Scrum* agar dapat berjalan dengan baik |
| Developers | Selvia Eldina | *Front-end* Developer |
| | Silvia Naim | *Backend* Developer |

Adapun tahapan-tahapan beserta ilustrasi dari fase pengembangan Aplikasi Birrussment adalah sebagai berikut :




Gambar 3.1. Fase-fase pengembangan dengan metode *Scrum*

### 3.4.1 *User story*

*User story* dikumpulkan untuk mewakili kebutuhan pengguna dalam pengembangan perangkat lunak dan juga sebagai alat komunikasi antara developer dan *Product Owner* untuk memahami kebutuhan yang harus dipenuhi. Dalam *Scrum*, User *story* akan menjadi acuan dalam pembuatan *Product Backlog*, yaitu daftar seluruh kebutuhan atau fitur pengembangan beserta prioritasnya. *Backlog* merupakan turunan atau analisis yang berasal dari kisah pengguna itu sendiri, dan seringkali berfokus pada nilai bisnis yang ingin dicapai. Misalnya, "*Sebagai [aktor], saya ingin dapat melakukan [kegiatan] agar [hasil yang diharapkan]*".

Tabel 3.4. Daftar *user story*

| Sebagai | Saya ingin… | Agar… |
|---|---|---|
| *Front-end* Developer | Membaca dokumentasi yang jelas dan terstruktur mengenai endpoint-endpoint yang tersedia, format *request* dan *response*, serta parameter-parameter yang diperlukan | Saya dapat dengan mudah mengintegrasikan *API* ini ke dalam aplikasi yang sedang saya kembangkan. |

| | | |
|---|---|---|
| | Mengakses *environment* pengujian yang berbeda dari *environment* pengujian *production* | Saya dapat menguji dan memvalidasi permintaan dan tanggapan *API* sebelum menerapkannya ke lingkungan produksi |
| | Mengintegrasikan endpoint autentikasi (‘/*auth*’) yang aman ke aplikasi *front-end*, baik untuk *login* enduser ataupun admin | Saya dapat membuat fungsional dari fitur *login* pengguna |
| | Mengintegrasikan endpoint pengguna (‘*/users*’) ke aplikasi *front-end* saya | Saya dapat membuat fungsional dari fitur edit profile dan fitur lainnya yang membutuhkan data pengguna |
| | Mengintegrasikan endpoint questions (‘/*questions*’) ke aplikasi *front-end* saya | Saya dapat membuat fungsional dari fitur *screening* |
| | Mengintegrasikan endpoint questions (‘*/results*’) ke aplikasi *front-end* saya | Saya dapat membuat fungsional dari fitur melihat hasil *screening* |
| Admin | Melakukan *login* ke sistem Website Admin dengan memberikan kredensial yang valid, seperti nama pengguna dan kata sandi | Saya dapat mengakses fitur-fitur yang ada. |
| | *Logout* dari sistem dengan mudah | Saya dapat mengamankan akses ke fitur administratif dan melindungi informasi yang sensitif. |
| | Melihat *list* pengguna yang *screening* terbaru, jumlah pengguna, jumlah pengguna | Saya dapat dengan mudah mengambil informasi yang dibutuhkan dengan cepat |

| | | |
|---|---|---|
| | yang telah *screening*, jumlah pengguna terindikasi *speech delay* dan grafik perbandingan presentase umur anak | |
| | Melihat detail informasi dari pengguna mulai dari nama anak, tanggal lahir dll | Dapat dengan cepat melihat informasi terkait pengguna apabila dibutuhkan. |
| | Melihat halaman statistik terkait statistik perkembangan hasil *screening* anak | Saya dapat dengan cepat melihat informasi terkait statistik tersebut apabila dibutuhkan. |
| | Melihat hasil Assessment atau *screening* anak | Saya dapat dengan cepat melihat informasi terkait hasil *screening* anak apabila diperlukan |
| | Mengelola pertanyaan *screening* | Saya dapat dengan cepat mengatasi pertanyaan yang tidak sesuai |
| | Mencari pengguna, pertanyaan, dsb | Dapat memudahkan saya menemukan informasi yang saya butuhkan berdasarkan keyword yang di*input* |
| Super Admin | Menambahkan admin baru ke sistem dengan memberikan informasi kredensial *login* | Admin baru dapat masuk ke sistem |

Adapun penjelasan aktor yang terdapat dalam User *story* pengembangan aplikasi *screening speech delay* adalah sebagai berikut :

Tabel 4.1. Aktor dalam user *story*

| Aktor | Penjelasan |
|---|---|
| *Front-end* Developer | Aktor yang bertanggungjawab dalam merancang dan membangun aplikasi web *screening speech delay* yang responsif, interaktif dan u*ser friendly*. |
| Administrator | Aktor yang bertanggungjawab dalam mengelola resource yang ada di Website Admin aplikasi *screening speech delay* |
| *Superadmin* | Aktor yang memiliki semua izin dalam Website Admin termasuk hak khusus yang yakni menambahkan aktor lain |

### 3.4.2 *Product Backlog*

Pada tahap ini akan dibuat daftar semua fitur, fungsi dan kebutuhan produk yang akan dibuat. *Product Owner* dan Pengembang bertanggung jawab untuk memprioritaskan *Product Backlog* dan estimasi waktunya. Ketika mengatur prioritas, task dengan *Story Point* yang lebih tinggi memerlukan lebih banyak waktu dan usaha untuk diselesaikan. Oleh karena itu, tugas-tugas dengan *Story Point* yang lebih tinggi biasanya diutamakan, karena membutuhkan perhatian lebih dari tim. Fitur tambahan tidak diperlukan dan dapat ditambahkan dan dikembangkan di tujuan produk mendatang jika fitur inti telah terpenuhi. Dalam menentukan estimasi, terdapat *sizing rule* yang akan digunakan seperti :

a. 0.5 *Story Point* = Apapun di bawah 4 jam kerja, cepat dan mudah dilakukan
b. 1 *Story Point* = ½ hari jam kerja (4 jam)
c. 2 *Story Points* = 1 hari jam kerja (8 jam)
d. 3 *Story Points* = 2 hari (16 jam)
e. 5 *Story Points* = 3-4 hari (24-32 jam)
f. 8 *Story Points* = separuh *Sprint*/5 hari (40 jam) (nilai poin tertinggi untuk kartu untuk 1 minggu)

Berikut ini *Product Backlog* untuk aplikasi Birrussment :

Tabel 3.5. *Product backlog REST API* (fitur utama)

| No. | Item | Prioritas | Points |
|---|---|---|---|
| 1. | Inisialisasi proyek, mengatur struktur folder dan *routing* pada Backend *RESTful API* | Sedang | 2 |
| 2. | Menyiapkan atau setup *MongoDB* atlas | Rendah | 0.5 |
| 3. | Membuat pemodelan *database* untuk *collection* User | Rendah | 1 |
| 4. | Setup *API* auth seperti membuat *routing* dan *boilerplate* controllernya | Tinggi | 5 |
| 5. | API Auth*(request* untuk endpoint *register*, *login*, logout) | Tinggi | 5 |
| 6 | Menerapkan uniq email untuk *register* | Rendah | 0.5 |
| 7. | Membuat *Role Based Access Control* (RBAC) | Rendah | 1 |
| 8. | Hash Password | Rendah | 1 |
| 9 | API User | Tinggi | 5 |
| 10 | Membuat skema pemodelan *database* untuk collection Question | Rendah | 1 |
| 11 | API CRUD Question | Sedang | 3 |
| 12 | Docs *API* Auth dan User | Rendah | 0.5 |
| 13 | Mengatur cors *cookie* | Rendah | 0.5 |
| 14 | *Deployment* | Sedang | 2 |
| 15 | API CRUD Question | Sedang | 3 |
| 16 | API CRUD Result | Tinggi | 5 |
| 17 | Docs *API* Question | Rendah | 0.5 |
| 18 | Docs *API* Result | Rendah | 0.5 |
| 19 | *Testing RESTful API* dengan *Black Box* | Sedang | 3 |

Tabel 3.6. *Product backlog REST API* (fitur tambahan)

| No. | Item | Prioritas | Points |
|---|---|---|---|
| 1. | OAuth *login* dengan google | Sedang | 2 |
| 2. | OAuth *register* dengan google | Rendah | 1 |
| 3. | Endpoint ubah password (forget password) | Rendah | 1 |

Tabel 3.7. *Product backlog* Website Admin (fitur utama)

| No. | Item | Prioritas | Points |
|---|---|---|---|
| 1. | *Slicing* dan integrasi *API* komponen *Sidebar* | Sedang | 2 |
| 2. | *Slicing* dan integrasi *API* komponen *Navbar* | Sedang | 2 |
| 3. | Layout Dashboard | Sedang | 2 |
| 4. | *Routing* | Rendah | 1 |
| 5. | *Reusable table* untuk page user management, page hasil assessment, page question | Sedang | 3 |
| 6 | Pagination | Rendah | 0.5 |
| 7. | Setup *state management* zustand | Tinggi | 5 |
| 8. | Card statistik di Dashboard | Sedang | 3 |
| 9. | *chart* untuk presentase range umur anak [Website Admin] | Sedang | 2 |
| 10. | *progress chart* | Sedang | 2 |
| 11 | latest result di *Dashboard* | Rendah | 1 |
| 12 | skleton tabel | Rendah | 1 |
| 13 | Improve UI login page | Rendah | 1 |
| 14 | Menampilkan *no data* pada *table* and *chart* ketika data kosong | Rendah | 1 |
| 15 | Menambakan setting page | Sedang | 3 |
| 16 | *Testing* Website Admin dengan *Cypress* | Sedang | 3 |

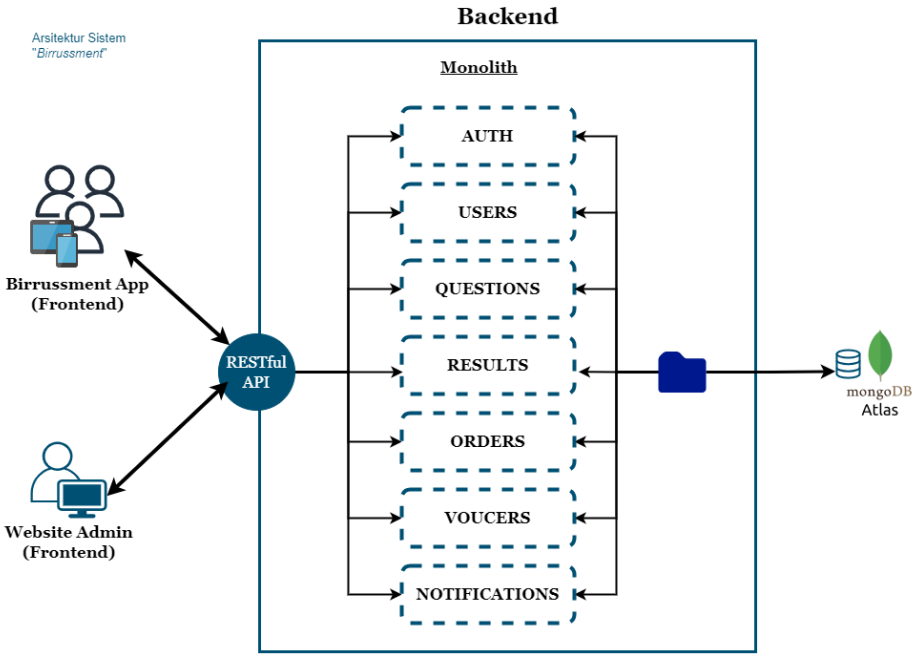
Tabel 3.8. *Product backlog* Website Admin (fitur tambahan)

| No. | Item | Prioritas |
|---|---|---|
| 1. | Ubah password | Rendah |
| 2. | Edit profile | Tinggi |
| 3. | Tambah admin | Sedang |
| 4. | Detail result pengguna | Rendah |

Berikutnya merupakan desain sistem berupa gambaran desain arsitektur, fitur dan fungsionalitas terhadap aplikasi yang akan dikembangkan. Pada tahap ini akan dibuat beberapa desain meliputi :

a) Desain Arsitektur Sistem

Design arsitektur sistem digunakan untuk menjelaskan aturan komunikasi sistem, dan hubungan struktur keseluruhan sistem perangkat lunak yang mencakup desain komponen *API*, *database*, gaya arsitektur, dll. Berikut desain Arsitektur Sistem yang akan dibuat dapat dilihat pada gambar :

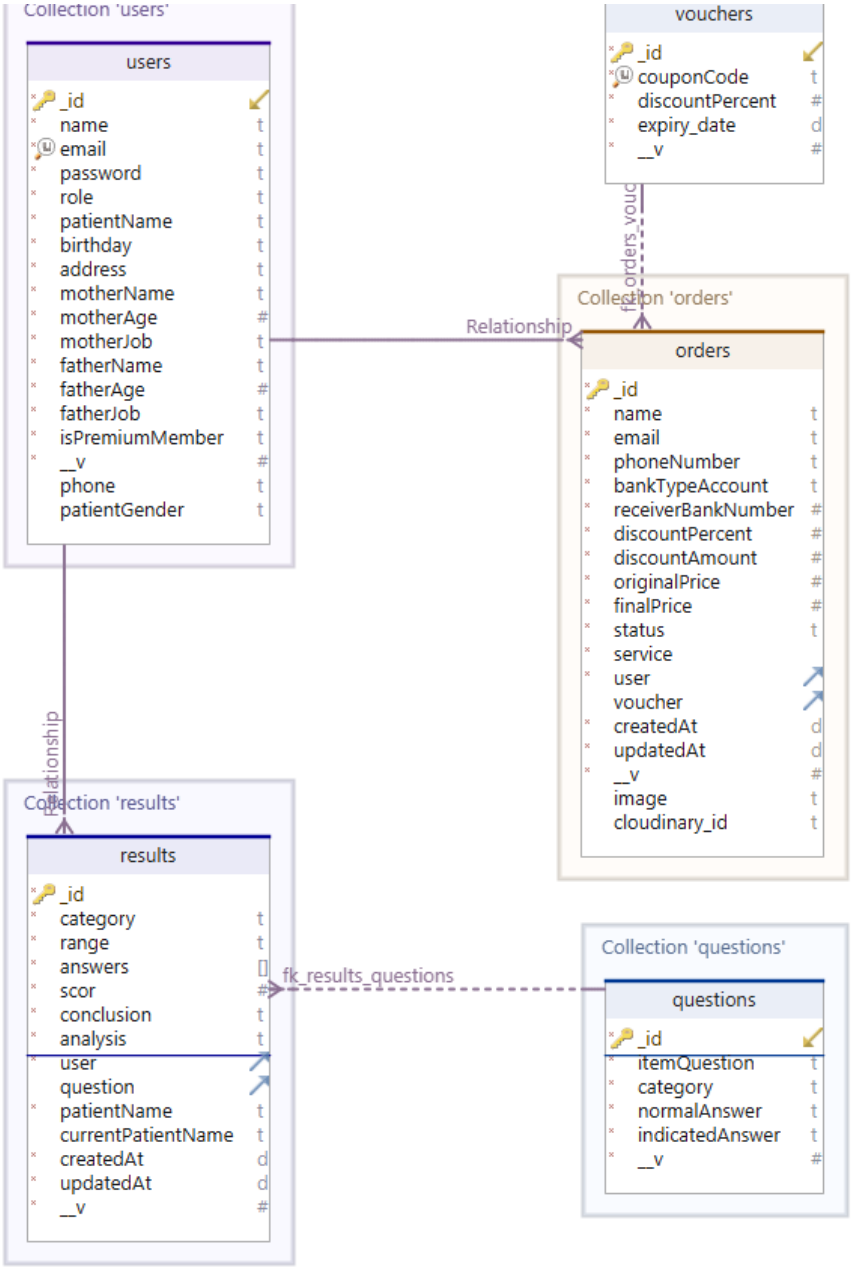



Gambar 3.2. Ilustrasi arsitektur sistem

Berdasarkan Ilustrasi gambar 3.2, dapat dilihat bahwa terdapat *client-side*, *database*, *server-side* (dengan gaya arsitektur *monolith*), dan *RESTful API*. Cara kerjanya, *client-side* (sebagai antarmuka aplikasi berbasis web) mengirim permintaan ke *server-side* melalui *RESTful API* menggunakan protokol *HTTP*. Selanjutnya *server-side* menerima permintaan dari *client-side* tersebut, memproses logika bisnis tersebut, mengakses dan *memanipulasi* data dalam *database* lalu mengirimkan *response* kembali kepada *client-side* melalui *API*. *Server-side* dalam arsitektur *monolith* mengintegrasikan logika bisnis, *database*, Dan API.

b) Desain *Database*

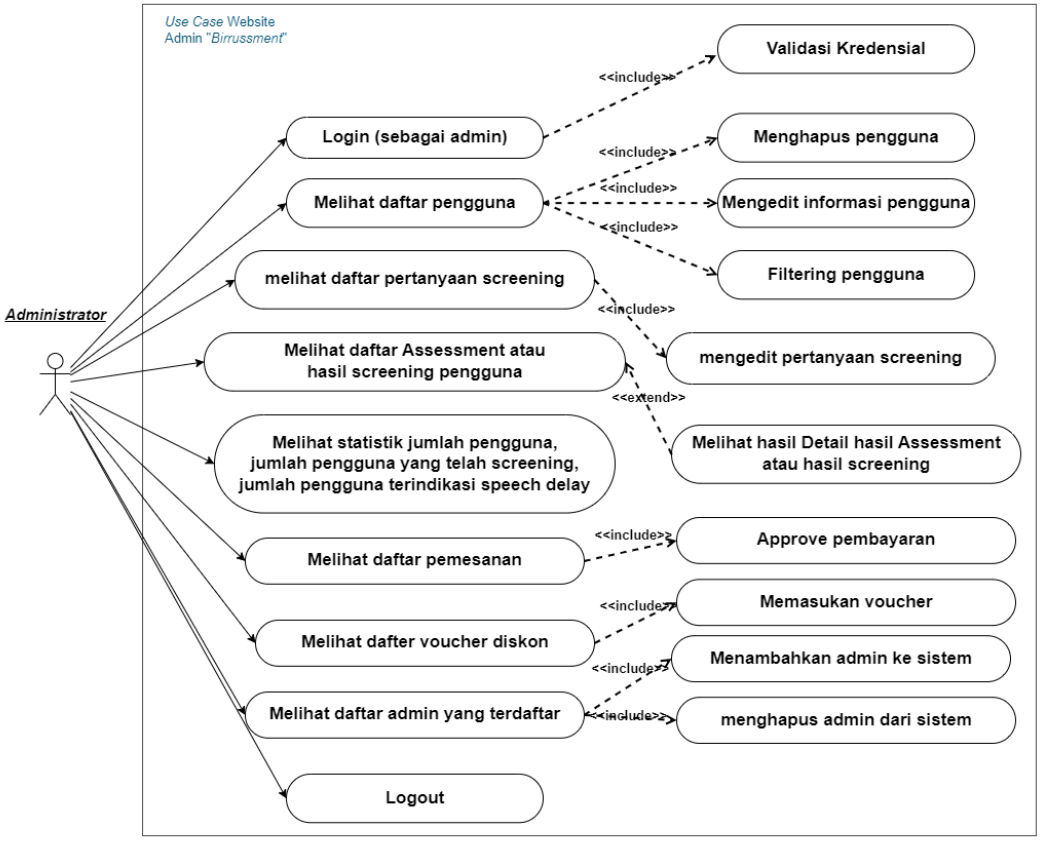
Untuk memudahkan penggambaran data relasional atau relasional maka perlu dibuat suatu desain *database*. Salah satu alat untuk menggambarkan hubungan adalah ERD (*Entity Relationship Diagram*) yang memungkinkan untuk memodelkan dan menggambarkan hubungan antar entitas dari suatu sistem. Ini berguna untuk merancang struktur basis data. Berikut desain *database* berupa *Entity Diagram Relationship* yang akan dibuat pada gambar 3.3




Gambar 3.3. Design *database* sistem dengan ERD

c) *Use case* (Gambaran fungsional Website Admin)

*Use case* yang digunakan pada sistem didefinisikan berdasarkan *user story* yang telah didapatkan. *Use case* kemudian digambarkan dalam sebuah diagram yaitu *Use case* diagram. Berikut *Use case* untuk Gambaran fungsional Website Admin yang akan dibuat :




Gambar 3.4. *Use case* untuk gambaran fungsional Website Admin

Tabel 3.9. Penjelasan aktor *use case* administrator

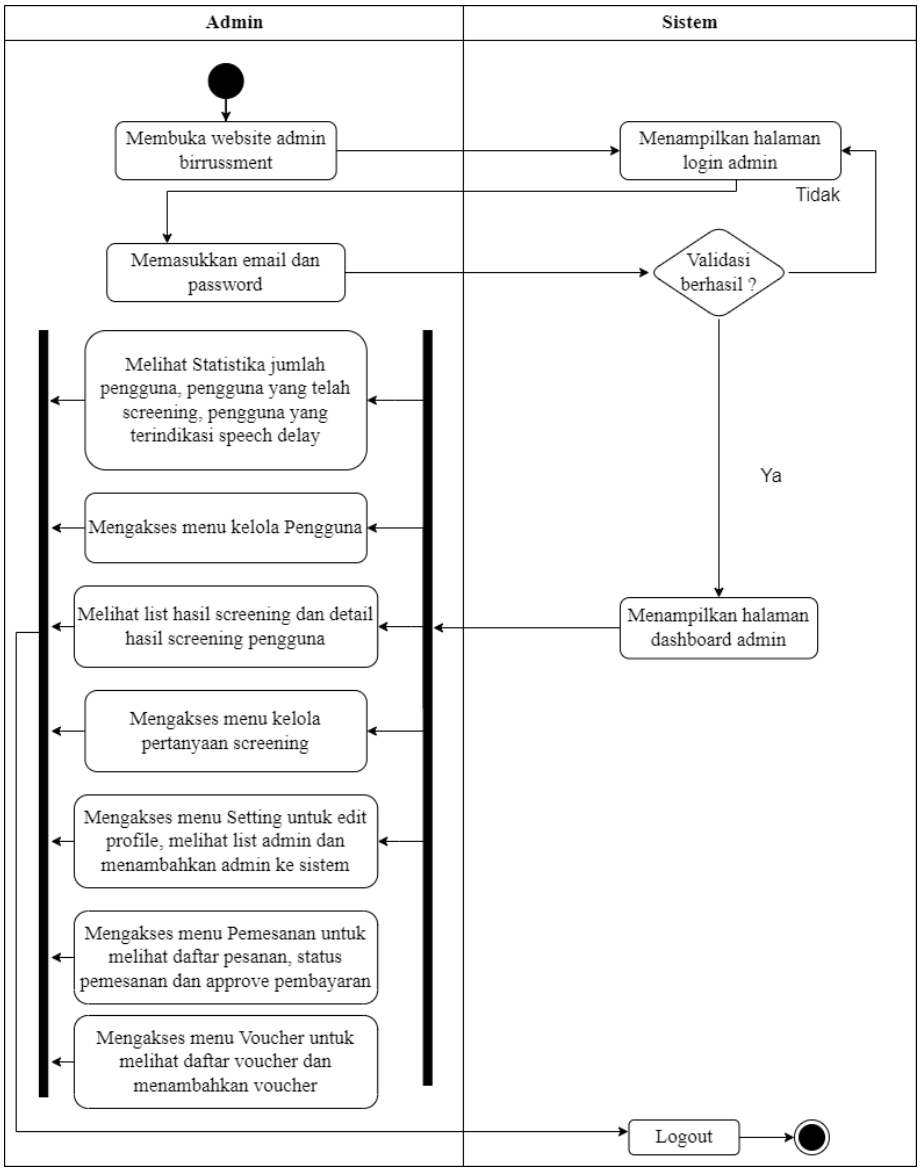
| No. | *Use case* | Dekripsi |
|---|---|---|
| 1. | *Login* (sebagai admin) | Dilakukan untuk mengakses semua menu admin website. Untuk terhubung, seorang administrator harus memasukkan email dan kata sandi untuk mendapatkan akses ke sumber daya sesuai dengan perannya |
| 2. | Validasi kredensial | Dilakukan untuk memvalidasi kredensial (misalkan email dan kata sandi) yang dimasukkan oleh pengguna saat *login*. Tujuan utama dari kasus penggunaan ini adalah untuk memastikan bahwa kredensial yang diberikan sudah benar sehingga pengguna dapat mengakses sistem dengan izin yang tepat. . |

| | | |
|---|---|---|
| 3. | Melihat daftar pengguna | memungkinkan administrator untuk melihat daftar semua pengguna terdaftar dalam sistem. Ini membantu admin untuk melacak jumlah pengguna yang menggunakan aplikasi uji keterlambatan suara dan mendapatkan informasi umum tentang mereka. |
| 4. | Melihat informasi pengguna | Dengan *Use case* ini, admin dapat melihat informasi detail tentang pengguna tertentu. Informasi ini dapat mencakup data pribadi, riwayat seleksi, dan hasil penilaian lainnya. |
| 5. | Mengedit Informasi Pengguna | *Use case* ini memungkinkan administrator untuk mengubah atau memperbarui informasi pengguna. Administrator dapat memperbarui data pribadi, mengubah status atau peran pengguna, atau memperbarui informasi relevan lainnya. |
| 6. | *Filtering* pengguna | untuk memfilter atau memfilter daftar pengguna berdasarkan kriteria tertentu. Administrator dapat menggunakan *filter* untuk mencari pengguna berdasarkan atribut seperti nama pengguna atau nama anak. |
| 7. | Melihat daftar pertanyaan *screening* | Dalam *Use case* ini, admin dapat melihat daftar pertanyaan yang digunakan dalam proses *screening speech delay*. Dan dapat mengetahui apakah ada kesalahan penulisan |
| 8. | Mengedit pertanyaan *screening* | memungkinkan administrator untuk mengubah atau memperbarui pertanyaan yang digunakan dalam proses penyaringan. Tujuannya adalah untuk memastikan bahwa pertanyaan yang diajukan kepada pengguna tetap relevan, bebas dari kesalahan ketik, dan efektif dalam menentukan keterlambatan bicara. |
| 9. | Melihat | Dengan *Use case*, Administrator dapat melihat detail hasil filter individu untuk pengguna. |

| | | |
|---|---|---|
| | detail hasil *screening* pengguna | |
| 10. | Melilhat daftar hasil *screening* pengguna | Memungkinkan administrator melihat daftar hasil filter untuk semua pengguna. Administrator dapat melihat skor, status hasil penilaian, dan informasi terkait lainnya. |
| 11. | Melihat statistika pengguna | Admin dapat melihat semua statistik pengguna. Ini termasuk data seperti jumlah pengguna dan informasi penting lainnya. tujuannya adalah untuk memberikan gambaran lengkap tentang karakteristik pengguna umum kepada administrator. |
| 12. | Melihat daftar admin yang terdafter | Memungkinkan administrator untuk melihat daftar administrator terdaftar di sistem. administrator dapat melihat informasi dasar tentang administrator lain yang memiliki akses ke situs admin. |
| 13. | Menambahkan admin ke sistem | Dalam *Use case* ini, super admin dapat menambahkan administrator baru ke sistem. Admin baru akan mendapatkan hak akses tertentu untuk mengelola situs admin |
| 14. | Menghapus admin ke sistem | Kasus penggunaan ini memungkinkan superadmin untuk menghapus administrator yang tidak aktif atau tidak diperlukan dari sistem. Admin yang dihapus akan kehilangan akses ke situs admin. |
| 15. | *Logout* | *Use case* ini digunakan ketika admin ingin memutuskan sambungan dari sesi *login* mereka. Saat keluar, admin akan dihapus dari situs admin dan tidak lagi memiliki akses ke fitur dan sumber daya terkait. |

a) *Activity Diagram*

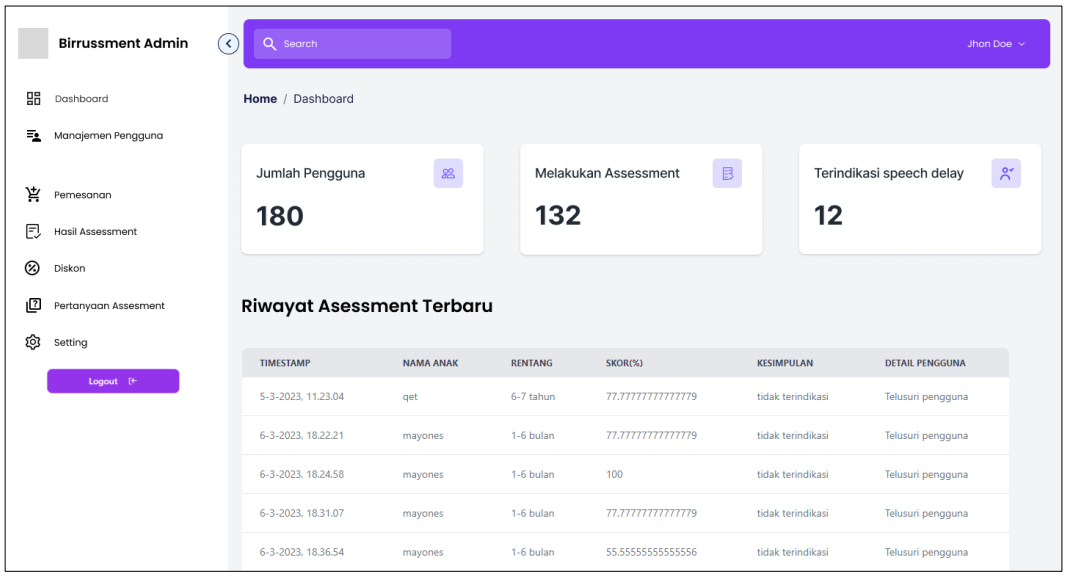
Sebelum melakukan pengembangan perlu untuk menggambarkan aliran kerja atau alur aktivitas dalam dari sistem Website Admin. Dalam penelitian ini, *Activity Diagram* akan digambarkan agar memudahkan developer memahami alur aktivitas admin dari Website Admin yang akan dibuat. Berikut *Activity Diagram* untuk Website Admin yang akan dibuat :



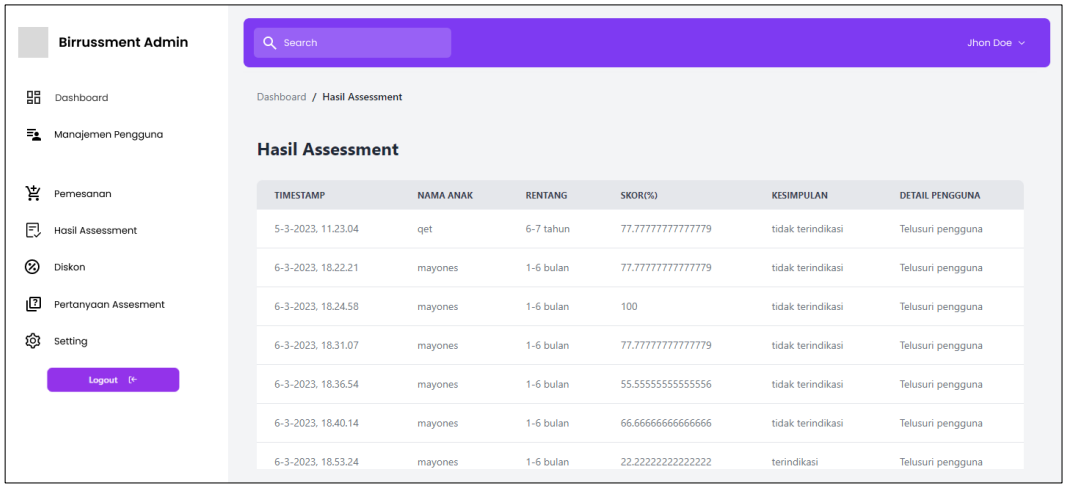

Gambar 3.5. *Activity diagram* Website Admin

*a) Mockup*

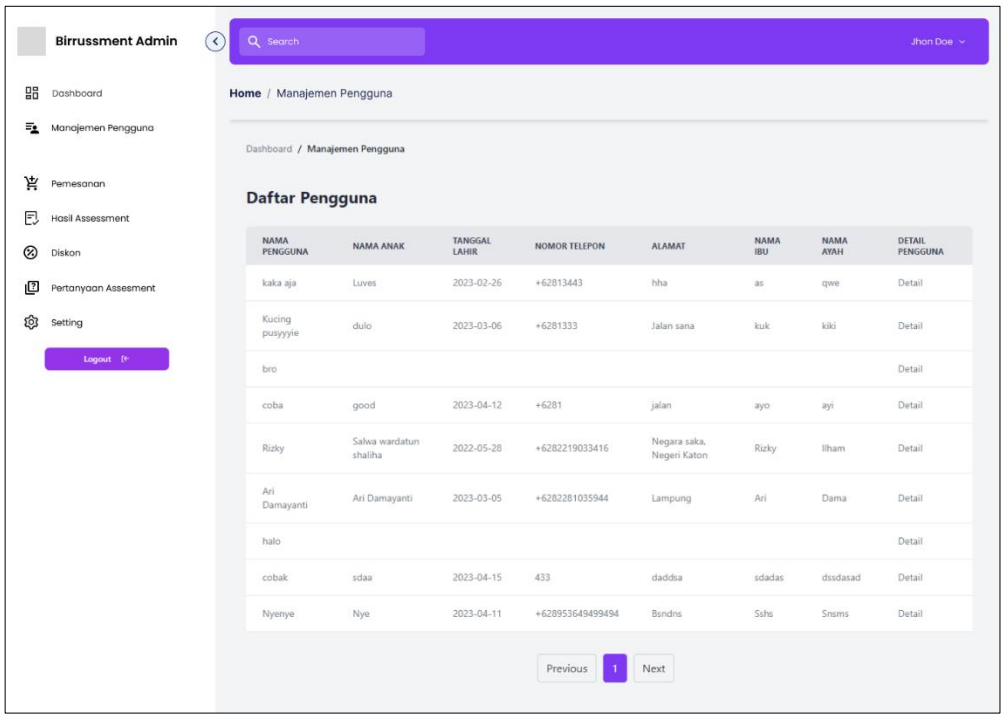
Perancangan antarmuka meliputi mewujudkan antarmuka dari setiap halaman yang ditampilkan pada aplikasi. Fase desain dimulai ketika desain sistem selesai. Memang, untuk menghindari pembuatan kode berulang, harus didefinisikan dengan tegas dalam desain sistem, tanpa modifikasi apa pun, di awal proses pengembangan. Perancangan antarmuka dilakukan dengan menggunakan software figma *open-source*. Berikut *Mockup* Website Admin yang akan dibuat :



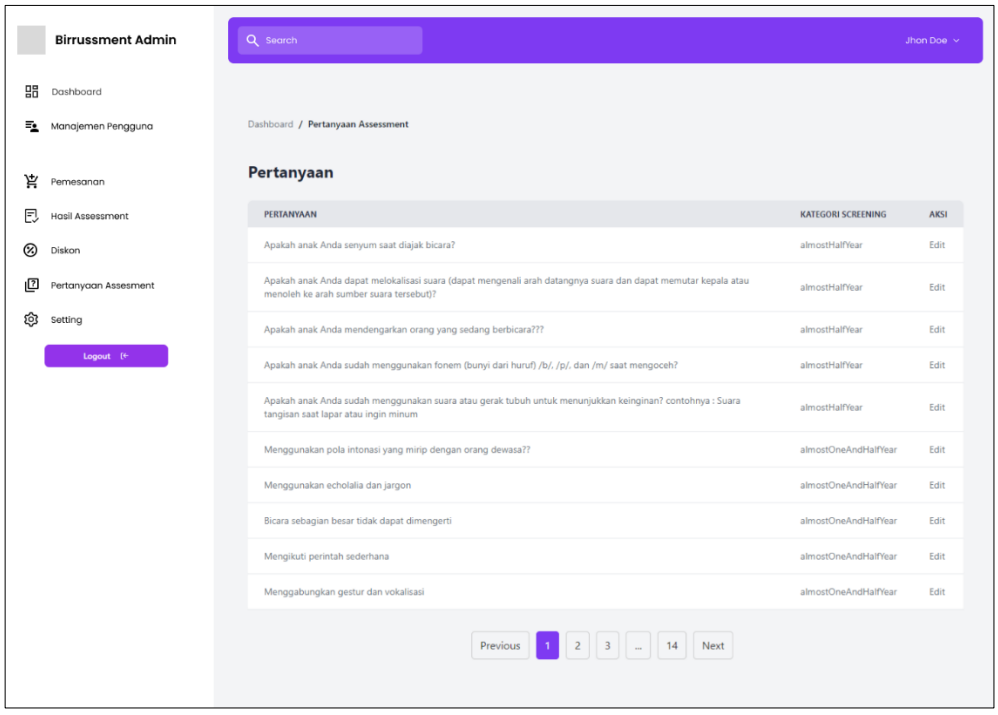

Gambar 3.6. *Mockup* menu *dashboard*



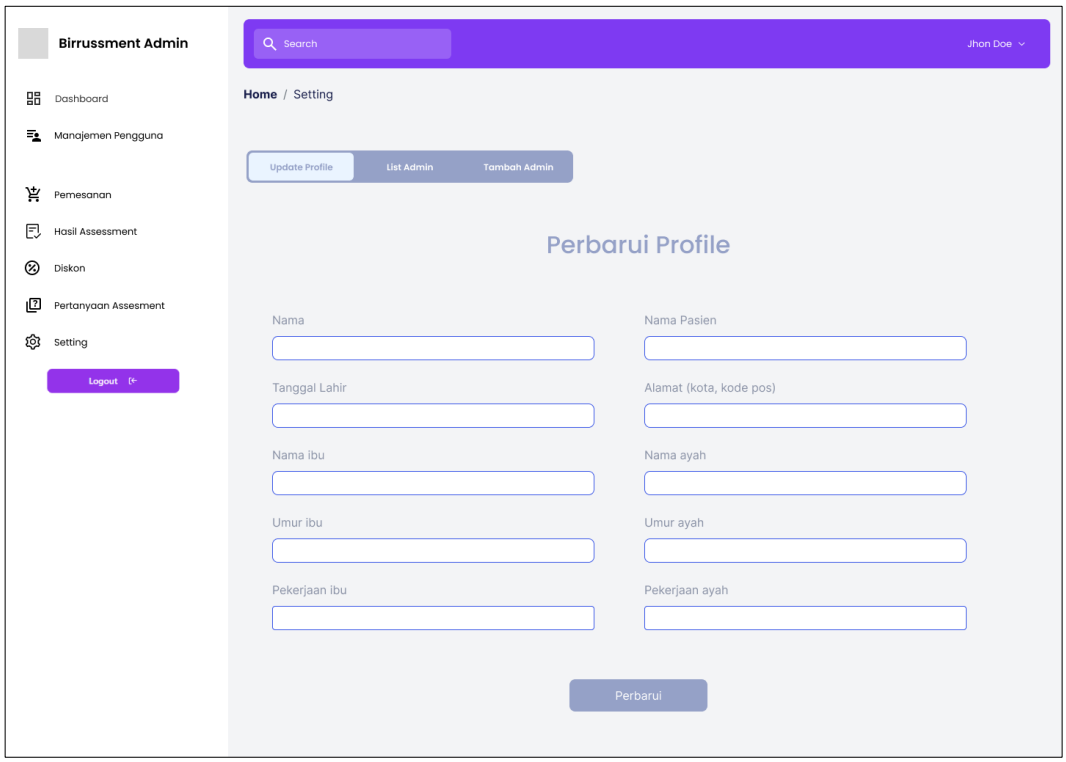

Gambar 3.7. *Mockup* menu hasil assessment




Gambar 3.8. *Mockup* menu manajemen pengguna




Gambar 3.9. *Mockup* menu list pertanyaan




Gambar 3.10. *Mockup* menu *setting*

### 3.4.3 *Sprint Planning*

Perencanaan *Sprint* dilakukan pada awal *Sprint* dan didukung oleh *Product Owner* dan developer untuk merencanakan *Sprint* yang akan berlangsung dan menentukan daftar produk yang akan diproduksi. *Scrum Master* harus memastikan bahwa acara berlangsung dan para peserta memahami tujuannya. Hasil perencanaan *Sprint* datang dalam bentuk tumpukan *Sprint* dengan tujuan *Sprint*. *Sprint Backlog* adalah daftar tugas atau fitur yang diselesaikan selama *Sprint* untuk mencapai tujuan *Sprint*.

### 3.4.4 *Daily Scrum*

*Daily Scrum* atau *daily standup* merupakan aktivitas diskusi yang diadakan oleh *Scrum Master* dan Development Team setiap hari selama *Sprint* dan berdurasi 15 menit. Tetapi dalam praktiknya, waktu pelaksanaan daily standup itu tergantung kebutuhan, dapat juga dilakukan beberapa kali dalam satu putaran *Sprint*. Aktivitas

ini bertujuan untuk mengoptimalkan kolaborasi dan performa dengan melakukan inspeksi pekerjaan hari sebelumnya dan membuat rencana pekerjaan selama satu hari ke depan. Topik yang dibahas dalam *Daily Scrum* biasanya menjawab pertanyaan berupa : “Apa yang kamu selesaikan kemarin?”, “Apa yang akan kamu kerjakan hari ini?”, “Apakah kamu ada *blocker* untuk hari ini?” .

Pada penelitian ini, pengujian dilakukan pada fase iterasi *Sprint*. Pengujian atau testing yang dilakukan menggunakan metode *End-to-end Testing*. *End-to-end Testing*, sering disingkat dengan E2E, adalah sebuah teknik *Testing* yang menguji seluruh produk atau keseluruhan suatu perangkat lunak. Pengujian ini dilakukan menggunakan skenario yang ditentukan untuk memastikan alur aplikasi berperilaku seperti yang diharapkan.

Pada pengujian E2E *REST API*, alat yang digunakan untuk pengujian menggunakan *Black Box*. Teknik pengujian *Black Box* yang digunakan adalah dengan menggunakan tabel keputusan *Test*. Misalnya, memastikan semua endpoint berfungsi dengan benar, memverifikasi bahwa permintaan *HTTP* yang Valid mengembalikan *response* yang benar, memastikan bahwa *API* memberikan *response* yang diinginkan untuk permintaan yang salah, dan seterusnya.

Dalam pengujian E2E Webmaster, alat yang digunakan untuk pengujian menggunakan *Cypress*. *Cypress* adalah salah satu perangkat lunak yang digunakan untuk pengujian otomasi. Contoh skenario yang dapat diuji dengan *Cypress* seperti Pastikan tautan pada halaman web mengarah ke halaman yang benar, Masukkan teks dalam formulir *input* dan periksa hasilnya, Pastikan halaman terlihat web responsif pada perangkat dan lebar layar yang berbeda, dll.

### 3.4.5 *Sprint Review*

Tahapan *Sprint Review* yang akan dilakukan dalam studi ini diadakan pada akhir *Sprint* untuk meninjau dan mengevaluasi pekerjaan yang telah diselesaikan dalam *Sprint* yang baru saja diselesaikan. Selama peninjauan *Sprint*, tim *Scrum* dan pemangku kepentingan berkolaborasi tentang apa yang telah dilakukan dalam *Sprint*. Tinjauan *Sprint* juga bisa disebut "demo" di mana tim dapat mendemonstrasikan pekerjaan yang dilakukan dalam iterasi dan mendapatkan umpan balik langsung dari pemangku kepentingan proyek.

### 3.4.6 *Sprint Retrospective*

Langkah ini menjadi kesempatan bagi tim untuk mengintrospeksi diri dan membuat rencana perbaikan yang akan diterapkan di *Sprint* selanjutnya. Selama fase ini, setiap anggota tim menerima umpan balik cepat untuk lebih meningkatkan produk dan budaya pengembangan. Perbaikan membantu tim memahami apa yang bekerja dengan baik dan apa yang tidak di *Sprint* berikutnya .

## 3.5 Penulisan Laporan

Pada akhir proses penelitian dan pengembangan perangkat lunak. Langkah selanjutnya adalah menulis laporan yang akan menjadi referensi untuk penelitian selanjutnya serta literatur untuk penelitian ini.

# V. SIMPULAN DAN SARAN

## 5.1 Simpulan

Berdasarkan penelitian yang telah dilakukan dapat diambil kesimpulan bahwa :

1. Berhasil dibangun *RESTful API* dan menghasilkan fitur berupa *API* Authentication, *API* CRUD User, *API* CRUD Question, dan *API* CRUD Result untuk diintegrasikan dengan Aplikasi *Screening Speech delay*.
2. Berhasil dibangun Website Admin yang dapat memudahkan administrator melacak data pengguna, hasil test dan grafik perkembangan skor assessment *speech delay*.
3. Proses pelaksanaan proyek *RESTful API* dan Website Admin menggunakan *framework Scrum* metode *Agile* yang memiliki 19 *user story* lalu dilakukan *breakdown* hingga menghasilkan 42 *Product Backlog*. *Product Backlog* ini kemudian didistribusikan lagi untuk 5 *Sprint Backlog*.
4. Berdasarkan hasil pengujian *RESTful API* menggunakan metode *Black Box*, fitur-fitur yang dikembangkan telah berjalan sesuai dengan *product goals*. Ada total 18 skenario yang berhasil diuji pada fitur *API* Authentication, *API* CRUD User, *API* CRUD Question, dan *API* CRUD Result.
5. Berdasarkan hasil pengujian Website Admin menggunakan *automation testing* dengan *cypress*, fitur-fitur yang dikembangkan telah berjalan sesuai dengan *product goals*. Ada total 12 skenario yang berhasil diuji meliputi pada fitur *login*, navigasi halaman, *search* dan *logout*.
6. Berdasarkan hasil pengujian performansi *RESTful API* menggunakan metode *Stress Test*, tingkat kinerja *RESTful API* untuk Aplikasi *Screening Speech delay* akan optimal apabila maksimal diakses oleh 120 *request* dalam periode *ramp up* 1 detik dan *loop count* adalah 1. Lalu server akan mengalami penurunan kinerja secara drastis apabila diakses oleh lebih dari 600 *request*.

## 5.2 Saran

Saran yang didapatkan dari penelitian ini adalah sebagai berikut :

1. Melakukan pengembangan *RESTful API* lebih lanjut dengan menambahkan fitur *Authentication* dengan *Google* agar memberikan pengalaman *login* yang cepat dan mudah bagi pengguna.
2. Melakukan pengembangan Website Admin lebih lanjut dengan menambahkan fitur *preview* hasil assessment anak agar administrator dapat melihat dengan lebih detail apa yang telah dijawab oleh orang tua yang mengikuti deteksi dini *speech delay*. Ini akan membantu administrator dalam mengevaluasi dan memantau perkembangan anak secara lebih efektif.
3. Melakukan *upgrade* ke versi berbayar pada server hosting *back-end* sebagai solusi untuk meningkatkan kinerja pada *RESTful API*. Versi berbayar menyediakan sumber daya server yang lebih baik, dukungan teknis yang lebih cepat, dan fitur tambahan yang dapat meningkatkan kecepatan *response* aplikasi.

# DAFTAR PUSTAKA

# DAFTAR PUSTAKA


[1] S. Desiarna, U. Nafila, Restiani, and Fatmawati, "Gangguan Keterlambatan Berbicara (Speech Delay) pada Anak Usia Dini," *Jurnal Penelitian dan Pengabdian Sastra, Bahasa, dan Pendidikan*, vol. 2, no. 2, 2023, doi: https://doi.org/10.25299/s.v2i2.11743.

[2] A. Aulia, A. Rahma, and A. Hulwah, "Strategi Guru dalam Meningkatkan Perkembangan Bahasa Anak Usia 5-6 Tahun Di TK Al-Kautsar," *Jurnal Pendidikan Islam Anak Usia Dini dan Al-Qur'an*, vol. 1, no. 1, pp. 48–57, 2022, doi: https://doi.org/10.33511/ash-shobiy.v1n1.48-57.

[3] Masitoh, "Gangguan Bahasa Dalam Perkembangan Bicara Anak," *Jurnal Edukasi Lingua Sastra*, vol. 17, no. 1, pp. 40–54, 2019, doi: https://doi.org/10.47637/elsa.v17i1.105.

[4] J. C. Duby *et al.*, "Identifying infants and young children with developmental disorders in the medical home: An algorithm for developmental surveillance and screening," *Pediatrics*, vol. 118, no. 1, pp. 405–420, Jul. 2006, doi: 10.1542/peds.2006-1231.

[5] S. Surahman and E. B. Setiawan, "Aplikasi Mobile Driver Online Berbasis Android Untuk Perusahaan Rental Kendaraan," *ULTIMA InfoSys*, vol. 8, no. 1, pp. 35–42, 2017, doi: https://doi.org/10.31937/si.v8i1.554.

[6] I. Kurniawan and F. Rozi, "REST API Menggunakan NodeJS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android," *Jurnal Ilmiah Teknologi Sistem Informasi*, vol. 1, no. 4, pp. 127–132, 2020, doi: https://doi.org/10.30630/jitsi.1.4.18.

[7] D. V. Kornienko, S. V. Mishina, S. V. Shcherbatykh, and M. O. Melnikov, "Principles of securing RESTful API web services developed with python frameworks," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Nov. 2021, pp. 1–11. doi: 10.1088/1742-6596/2094/3/032016.

[8] nodejs.dev, "Introducing Node.js." https://nodejs.dev/ (accessed May 19, 2023).

[9] N. Hestiyana *et al.*, "Deteksi Kejadian Speech Delayed Pada Anak Dengan Algoritma Id3," *Dinamika Kesehatan Jurnal Kebidanan dan Keperawatan*, vol. 12, no. 2, pp. 2549–4058, 2021, doi: 10.33859/dksm.v12i2.

[10] H. Nur, M. Maritje, W. Tairas, and W. Hendriani, "The Experience of Hope for Mothers with Speech-Language Delay Children," *Journal of Educational, Health and Community Psychology*, vol. 7, no. 2, pp. 104–117, 2018, doi: http://dx.doi.org/10.12928/jehcp.v7i2.8936.

[11] K. G. Shipley and J. G. McAfee, "Assessment in Speech-Language Pathology," 5th edition.Boston: Cengage Learning, 2016, pp. 263–267.

[12] S. Gautam, "Deno-A new Node.js?," University of Applied Sciences, Helsinki, 2021. Accessed: Aug. 27, 2023. [Online]. Available: https://urn.fi/URN:NBN:fi:amk-202105178893

[13] A. Mubariz *et al.*, "Perancangan Back-End Server Menggunakan Arsitektur Rest dan Platform Node.JS (Studi Kasus: Sistem Pendaftaran Ujian Masuk Politeknik Negeri Ujung Pandang)," in *Proc. Seminar Nasional Teknik Elektro dan Informatika (SNTEI)*, 2020, pp. 72–77. Accessed: Sep. 03, 2023. [Online]. Available: https://dspace.uii.ac.id/handle/123456789/38607

[14] npmjs.com, "About npm." https://www.npmjs.com/ (accessed May 22, 2023).

[15] expressjs.com, "Express : Fast, unopinionated, minimalist web framework for Node.js." https://expressjs.com/ (accessed May 22, 2023).

[16] Nasution, "Implementasi MongoDb, ExpressJs, ReactJs, Dan NodeJs (Mern) Pada Pengembangan Aplikasi Formulir, Kuis, Dan Survei Online," Universitas Islam Indonesia, Yogyakarta, 2021.

[17] L. Schaefer, "NoSQL vs. SQL Databases." https://www.mongodb.com/ (accessed May 25, 2023).

[18] react.dev, "React : The library for web and native user interfaces." https://react.dev/ (accessed May 25, 2023).

[19] P. Lestari, "Single Page Application (SPA) atau Aplikasi Halaman Tunggal," *ILKOM Jurnal Ilmiah*, vol. 10, no. 1, pp. 38–43, 2018, doi: https://doi.org/10.33096/ilkom.v10i1.204.38-43.

[20] M. F. Santoso, "Teknik Single Page Application (Spa) Layout Web Dengan Menggunakan React Js Dan Bootstrap," *Jurnal Khatulistiwa Informatika*, vol. 9, no. 2, pp. 107–114, 2021, doi: https://doi.org/10.31294/jki.v9i2.11357.

[21] S. Hadji and M. Taufik, "Implementasi Metode Scrum Pada Pengembangan Aplikasi Delivery Order Berbasis Website (Studi Kasus Pada Rumah Makan Lombok Idjo Semarang)," in *Proc. Konferensi Ilmiah Mahasiswa Unissula (Kimu) 2*, 2019, pp. 32–43.

[22] K. Schwaber and J. Sutherland, *Panduan Scrum*, 4th edition. Creative Commons, 2020. Accessed: Aug. 31, 2023. [Online]. Available:

https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Indonesian.pdf

[23] J. Lian Min, A. Istiqomah, A. Rahmani, P. Negeri Bandung, and P. P. Tester Padepokan Tujuh Sembilan-Bandung, "Evaluasi Penggunaan Manual Dan Automated Software Testing Pada Pelaksanaan End-To-End Testing," *Jurnal Teknologi Terapan)* |, vol. 6, no. 1, pp. 18–25, 2020, doi: https://doi.org/10.31884/jtt.v6i1.256.

[24] Y. Dwi Wijaya and M. Wardah Astuti, "Pengujian Blackbox Sistem Informasi Penilaian Kinerja Karyawan PT Inka (Persero) Berbasis Equivalence Partitions Blackbox Testing Of PT Inka (Persero) Employee Performance Assessment Information System Based On Equivalence Partitions," *Jurnal Digital Teknologi Informasi*, vol. 4, no. 1, pp. 22–26, 2021, doi: https://doi.org/10.32502/digital.v4i1.3163.

[25] F. C. Ningrum, D. Suherman, S. Aryanti, H. A. Prasetya, and A. Saifudin, "Pengujian Black Box pada Aplikasi Sistem Seleksi Sales Terbaik Menggunakan Teknik Equivalence Partitions," *Jurnal Informatika Universitas Pamulang*, vol. 4, no. 4, pp. 125–130, 2019, doi: http://dx.doi.org/10.32493/informatika.v4i4.3782.

[26] cypress.io, "Cypress Docs." https://www.cypress.io/ (accessed Jun. 08, 2023).

[27] N. Luh, A. S. Ginasari, K. S. Wibawa, N. Kadek, and A. Wirdiani, "Pengujian Stress Testing API Sistem Pelayanan dengan Apache JMeter," *Jurnal Ilmiah Teknologi dan Komputer*, vol. 2, no. 3, 2021, [Online]. Available: https://ojs.unud.ac.id/index.php/jitter/article/view/79652

[28] A. Rahayu, M. M. Kasih, O. Safitri, R. Nursanti, P. Taman, and B. Palembang, "Penerapan Aplikasi Deteksi Dini Speech Delay Pada Anak Batita Dengan Menggunakan Metode Pengembangan Kpsp Berbasis Android (Skrianteng-Mobile) Di Posyandu Wilayah Kerja Puskesmas Taman Bacaan Palembang," in *Repository Poltekkes Kemenkes Palembang*, 2018. Accessed: Sep. 03, 2023. [Online]. Available: https://repository.poltekkespalembang.ac.id/items/show/2506

[29] S. K. Murti, J. Informatika, T. Industri, A. Sujarwo Badan, and S. Informasi, "Membangun Antarmuka Pengguna Menggunakan ReactJs untuk Modul Manajemen Pengguna," *Journal Portal Universitas Islam Indonesia*, vol. 2, no. 2, pp. 1–6, 2021, Accessed: Sep. 03, 2023. [Online]. Available: https://journal.uii.ac.id/AUTOMATA/article/view/19443

.