

**PERANCANGAN REPOSITORI INSTITUSI MENGGUNAKAN
PROGRESSIVE WEB APPS DENGAN METODE *EXTREME*
PROGRAMMING
(STUDI KASUS REPOSITORI SKRIPSI DIGITAL
TEKNIK INFORMATIKA UNPAD)**

SKRIPSI

Diajukan kepada Program Studi S1 Teknik Informatika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Untuk Menyusun Tugas Akhir S1

VEGA SAVERA YUANA
NPM 140810160053



UNIVERSITAS PADJADJARAN
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI TEKNIK INFORMATIKA
SUMEDANG
2020

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan penyusunan skripsi yang berjudul “Perancangan Repositori Institusi Menggunakan Progressive Web Apps dengan Metode Extreme Programming (Studi Kasus Repositori Skripsi Digital Teknik Informatika Unpad)” sebagai salah satu syarat menempuh sarjana pada Program Studi S-1 Teknik Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.

Dalam proses penyusunan dan penulisan skripsi ini tidak terlepas dari bantuan, bimbingan, serta dukungan dari berbagai pihak. Oleh karena itu dalam kesempatan ini penulis mengucapkan terima kasih kepada Bapak Dr. Juli Rejito, M.Kom, sebagai pembimbing utama, Bapak Aditya Pradana, S.T., M.Eng, sebagai pembimbing pendamping yang telah meluangkan waktu dan pikirannya sehingga penulis dapat menyelesaikan skripsi ini. Ucapan terima kasih juga diberikan kepada keluarga penulis yang selalu memberikan motivasi dan doa yang menjadi pendorong dalam penyelesaian skripsi ini. Penulis juga mengucapkan terima kasih sebanyak-banyaknya kepada:

1. Dosen-dosen Teknik Informatika Unpad yang telah mengajar dan memberikan ilmu kepada penulis selama masa perkuliahan yang membawa penulis pada posisi sekarang ini.

2. Teman-teman Unfaedah, Reynaldi Noer, Afifah Khoeriah, Rafid ghadah, Santo Joosten, Dzakia Rayhana, dan Abieza Pradana yang telah menjadi *support system* bagi penulis selama masa perkuliahan.
3. Asep Nur Muhammad, Muhamad Yusrizan, Fidriyanto Rizkillah, Bariq Mbani, Eko Fajar Putra, dan Ibnu Ahsani yang telah banyak membantu penulis dalam proses menyelesaikan skripsi ini.
4. Teman-teman Cyber 2016 yang telah menempuh masa perkuliahan bersama-sama dan saling membantu dalam prosesnya.

Akhir kata, penulis berharap semoga skripsi ini dapat bermanfaat bagi pembaca.

ABSTRAK

Di era digital ini, banyak universitas yang menyediakan web repositori dimana mahasiswa dapat mengakses berkas digital dari skripsi atau karya ilmiah milik sivitas akademika. Digitalisasi ini tentunya memudahkan mahasiswa dalam memperoleh referensi dan sumber bacaan tanpa perlu mengunjungi perpustakaan. Dari pernyataan tersebut dibangun web repositori skripsi teknik informatika unpad untuk memudahkan akses skripsi milik universitas. Optimalnya, web repositori harus responsif terhadap *smartphone* dan memiliki pengalaman pengguna yang baik. Pengalaman pengguna dapat ditingkatkan dengan membuat web yang dapat digunakan seperti *native app* dan tetap memiliki tampilan dan informasi disaat *offline*. Hal ini dapat dicapai dengan mengimplementasikan PWA. Dalam prosesnya pengembangannya, diperlukan metode perangkat lunak yang tepat. *Extreme programming* adalah salah metode pengembangan perangkat lunak *Agile* sehingga bersifat gesit dan iteratif. Metode ini cocok untuk digunakan oleh proyek kecil dengan tim yang kecil pula. Dengan siklus singkat dan berfokus pada permintaan pengguna, aplikasi terus mendapat *feedback* sehingga meningkatkan kualitas aplikasi. Dari hasil peneitian didapat nilai likert 88.53. Dari hasil tersebut dapat disimpulkan bahwa sistem repositori yang dibangun dengan keunggulan PWA dan metode *extreme programming* berjalan dengan baik dan memiliki nilai kepuasan yang sangat baik.

Kata Kunci : ExpressJs, *Extreme Programming*, PWA, ReactJs, Repositori

ABSTRACT

In the digital era, many Universities provided web repository where students can access digital files of theses or scientific works belonging to academics. This digitalization certainly made it easier for students to obtain references and sources of reading without the need to visit the library. From this statement, the web repository was built to facilitate access to the university's scientific works. Optimally, the web repository must be responsive to smartphones and had a good user experience. The user experience could be improved by creating a web that can be used as a native app and still has an UI and information when offline. This could be achieved by implementing PWA. In the process, the right software development method was needed. Extreme programming is one of the Agile software development methods so it is agile and iterative. This method is suitable for small projects with small teams. With a short cycle and focus on user requests, the application continued to get feedback thereby improving the quality of the application. The research results obtained a Likert value of 88.53. These results could be concluded that the repository system that was built with the advantages of PWA and extreme programming methodology runs well and has a very good satisfaction value.

Keywords : *ExpressJs, Extreme Programming, PWA, ReactJs, Repository*

DAFTAR ISI

KATA PENGANTAR	iii
ABSTRAK	v
ABSTRACT	vi
DAFTAR ISI	vii
DAFTAR TABEL	x
DAFTAR GAMBAR	xi
DAFTAR LAMPIRAN	xiii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Identifikasi Masalah	4
1.3 Batasan Masalah	4
1.4 Maksud dan Tujuan Penelitian	5
1.5 Manfaat Penelitian	5
1.6 Metodologi Penelitian	6
1.7 Sistematika Penulisan	6
BAB II TINJAUAN PUSTAKA	8
2.1 Repositori Institusi	8
2.2 <i>Extreme Programming</i>	8
2.3 <i>Progressive Web Apps</i>	10
2.4 ReactJS	11
2.5 Express	12
2.6 Javascript	13
2.7 Bootstrap	14
2.8 MySQL	15
2.9 <i>Unified Modeling Language (UML)</i>	16
2.9.1 <i>Use Case Diagram</i>	16

2.9.2	<i>Activity Diagram</i>	17
2.9.3	<i>Deployment Diagram</i>	18
2.10	<i>Entity Relationship Diagram (ERD)</i>	19
2.11	<i>Black Box Testing</i>	20
2.12	<i>Usability Testing</i>	21
BAB III ANALISIS DAN PERANCANGAN		23
3.1	Fase Eksplorasi	23
3.1.1	Kebutuhan Pengguna	24
3.1.2	Kebutuhan Data	24
3.1.3	Kebutuhan Perangkat Lunak.....	25
3.1.4	Kebutuhan Perangkat Keras.....	25
3.1.5	Kebutuhan Sistem	26
3.1.6	<i>User Story</i>	28
3.2	Fase Perencanaan.....	30
3.3	Fase Iterasi	31
3.3.1	Analisis Sistem	32
3.3.2	Analisis Arsitektur Menggunakan <i>Deployment Diagram</i>	34
3.3.3	Analisis Basis Data	35
3.3.4	Desain Antarmuka	39
3.3.5	<i>Testing</i>	47
3.4	Fase Produksi.....	47
3.5	Fase Pemeliharaan dan Fase Akhir.....	50
BAB IV HASIL DAN PEMBAHASAN		51
4.1	Impelementasi Program	51
4.1.1	Halaman Utama	51

4.1.2	Halaman Register.....	54
4.1.3	Menu <i>Login</i>	59
4.1.4	Halaman Unggah Skripsi.....	63
4.1.5	Halaman Menu Admin.....	67
4.1.6	Halaman Verifikasi Akun.....	69
4.1.7	Halaman Tinjau Skripsi.....	73
4.1.8	Halaman Detail Skripsi.....	77
4.1.9	Fitur Pencarian dan Penyaringan.....	78
4.1.10	Halaman Profil dan Fitur <i>Edit Password</i>	81
4.1.11	Halaman Status Skripsi.....	85
4.1.12	Implementasi PWA.....	88
4.2	Fase Produksi.....	92
4.2.1	Rilisan Kecil.....	92
4.2.2	Pengujian.....	93
4.3	Fase Pemeliharaan dan Fase Akhir.....	94
4.3.1	<i>Feedback</i>	94
4.3.2	Implementasi <i>Feedback</i>	95
BAB V KESIMPULAN DAN SARAN.....		104
5.1	Kesimpulan.....	104
5.2	Saran.....	105
DAFTAR PUSTAKA.....		106
LAMPIRAN.....		108
RIWAYAT HIDUP.....		150

DAFTAR TABEL

Tabel 2.1 Simbol-Simbol pada <i>Use Case Diagram</i>	16
Tabel 2.2 Simbol-Simbol pada <i>Activity Diagram</i>	17
Tabel 2.3 Simbol-Simbol pada <i>Deployment Diagram</i>	18
Tabel 2.4 Simbol-simbol pada ERD	20
Tabel 2.5 Titik Respon	22
Tabel 3.1 Kebutuhan Sistem Pengguna: Mahasiswa	27
Tabel 3.2 Kebutuhan Sistem Pengguna: Admin	27
Tabel 3.3 <i>User Story</i>	28
Tabel 3.4 Prioritas Fitur	31
Tabel 3.5 Atribut Tabel skripsi	37
Tabel 3.6 Atribut Tabel <i>users</i>	38
Tabel 3.7 Atribut Tabel <i>temp_users</i>	38
Tabel 3.8 Atribut Tabel <i>forums</i>	39
Tabel 3.9 Atribut Kriteria Nilai	48
Tabel 3.10 Skenario <i>Testing</i>	49
Tabel 3.11 <i>Form Usability Testing</i>	49
Tabel 4.1 <i>Black Box Testing</i> pada Fitur Registrasi	58
Tabel 4.2 <i>Black Box Testing</i> pada Fitur <i>Login</i>	63
Tabel 4.3 <i>Black Box Testing</i> pada Fitur Unggah Skripsi	67
Tabel 4.4 <i>Black Box Testing</i> pada Fitur Verifikasi Akun	73
Tabel 4.5 <i>Black Box Testing</i> pada Fitur Tinjau Skripsi	76
Tabel 4.6 <i>Black Box Testing</i> pada Fitur Pencarian dan Penyaringan	81
Tabel 4.7 <i>Black Box Testing</i> pada Fitur <i>Edit Password</i>	84
Tabel 4.8 <i>Black Box Testing</i> Fitur PWA	92
Tabel 4.9 Hasil Pengujian Rilis Kecil	93
Tabel 4.10 <i>Black Box Testing</i> Kontak Admin	100

DAFTAR GAMBAR

Gambar 2.1 Siklus Hidup XP (Krishna <i>et al.</i> , 2011)	9
Gambar 2.2 Prinsip Utama PWA (Karpagam <i>et al.</i> , 2017).....	11
Gambar 2.3 Alur <i>Request</i> pada Express (Hahn, 2016:7)	12
Gambar 3.1 Kompatibilitas <i>Browser</i> Chrome dan Mozilla pada Perangkat (Santoni, 2018)	26
Gambar 3.2 <i>Use Case Diagram</i>	32
Gambar 3.3 <i>Activity Diagram</i>	33
Gambar 3.4 <i>Deployment Diagram</i>	34
Gambar 3.5 <i>Entity Relationship Diagram</i>	35
Gambar 3.6 Model Data Fisik	36
Gambar 3.7 Desain Halaman Utama.....	39
Gambar 3.8 Desain Halaman Registrasi Bagian 1 (a) dan Bagian 2 (b).....	40
Gambar 3.9 Desain <i>Login Section</i>	41
Gambar 3.10 Desain Halaman Detail Skripsi	41
Gambar 3.11 Desain Halaman Unggah Skripsi	42
Gambar 3.12 Desain Halaman Profil Mahasiswa	43
Gambar 3.13 Desain Halaman <i>Edit Password</i>	43
Gambar 3.14 Desain Halaman Status Skripsi	43
Gambar 3.15 Desain Halaman Menu Admin	44
Gambar 3.16 Desain Halaman Verifikasi Akun Mahasiswa	45
Gambar 3.17 Desain Halaman Tinjau Skripsi.....	45
Gambar 3.18 Desain Halaman Kontak Admin	46
Gambar 3.19 Desain Halaman Lupa <i>Password</i>	46
Gambar 4.1 Halaman Utama pada <i>Desktop</i> (a) dan <i>Mobile</i> (b).....	52
Gambar 4.2 Halaman Registrasi bagian 1	54
Gambar 4.3 Halaman Registrasi Bagian 2	54
Gambar 4.4 Menu <i>Login</i>	60
Gambar 4.5 Halaman Unggah Skripsi	63
Gambar 4.6 Halaman Menu Admin	68

Gambar 4.7 Halaman Verifikasi Akun	69
Gambar 4.8 Halaman Tinjau Skripsi.....	74
Gambar 4.9 Halaman Detail Skripsi	77
Gambar 4.10 Halaman Profil	82
Gambar 4.11 Bagian <i>Edit Password</i>	82
Gambar 4.12 Halaman Status Skripsi	85
Gambar 4.13 Implementasi Fitur <i>Offline</i>	90
Gambar 4.14 Implementasi Fitur <i>add to homescreen</i>	91
Gambar 4.15 Web Dalam Bentuk <i>Native App</i>	92
Gambar 4.16 Tampilan Profil Setelah Diperbaharui.....	95
Gambar 4.17 Halaman Edit Unggahan	96
Gambar 4.18 Navigator Menu.....	98
Gambar 4.19 Menu <i>Login</i> pada <i>Smartphone</i> (a) Lama (b) Baru	98
Gambar 4.20 Kontak Admin	99
Gambar 4.21 Halaman Lupa <i>Password</i>	101

DAFTAR LAMPIRAN

Lampiran 1 Ringkasan Hasil <i>Usability Testing</i>	108
Lampiran 2 Kode Web Aplikasi	114

BAB I

PENDAHULUAN

1.1 Latar Belakang

Teknologi informasi terus berkembang dengan pesat. Perkembangan teknologi ini tentunya memberi perubahan pada gaya hidup manusia. Sejak industri 3.0 manusia sudah mulai mengubah format informasi kedalam bentuk digital. Di era industri 4.0 tentunya digitalisasi semakin banyak diimplementasikan dalam kehidupan manusia. Salah satu contoh ialah *e-book*. Manusia mulai meninggalkan kertas dan beralih ke media digital. Hal ini dikarenakan media digital lebih mudah diakses dimana saja dan kapan saja.

Pada saat ini, banyak universitas yang menyediakan web repositori universitas dimana mahasiswa dapat mengakses berkas digital dari skripsi mahasiswa, disertasi, maupun karya ilmiah dosen yang merupakan karya sivitas akademika dari perguruan tinggi tersebut. Repositori institusi memiliki artian sebagai sekumpulan set layanan yang ditawarkan universitas kepada masyarakat untuk pengelolaan dan penyebaran dari materi yang dibuat oleh institusi tersebut (Repanovici, 2009). Adanya repositori tentunya bermanfaat bagi mahasiswa. Mahasiswa tidak perlu mengunjungi perpustakaan untuk mencari referensi melainkan hanya perlu mengakses web repositori universitas.

Untuk optimalisasi penggunaannya, web repositori universitas harus responsif terhadap *smartphone* karena lebih dari setengah pengguna web menggunakan *smartphone*. Aplikasi web tersebut juga harus bekerja dengan baik

walaupun dalam kondisi koneksi yang tidak stabil. Bila web memakan waktu yang lama untuk *me-load* atau tidak memiliki tampilan disaat koneksi buruk maka akan mengurangi minat pengguna dalam menggunakan web. Hal ini akan mengakibatkan manfaat dari web tidak tercapai dengan baik. Untuk mengatasi hal tersebut, perlu diterapkan konsep *progressive web apps* (PWA)

PWA adalah aplikasi web yang menggabungkan fitur-fitur aplikasi asli dengan teknologi web. Dengan kata lain PWA adalah situs web normal yang dibuat dengan teknologi web — HTML (*Hypertext Markup Language*), CSS (*Cascading Style Sheets*), dan JavaScript — namun menawarkan pengalaman yang lebih baik kepada pengguna (Hume, 2017).

PWA memiliki keunggulan yaitu bertingkah seperti *native application*, cepat, dan tidak bergantung pada koneksi. PWA akan mendaftarkan *service worker* yang dapat mendeteksi dan bereaksi terhadap perubahan dalam koneksi pengguna sehingga dapat memberikan pengalaman penuh pada saat *online*, *offline*, atau koneksi buruk. Contoh PWA dapat bertingkah seperti *native application*, ialah PWA memungkinkan pengguna untuk menyimpan *shortcut* web ke layar beranda. Muncul seperti aplikasi asli, memberi akses mudah ke aplikasi web dengan satu sentuhan tombol. PWA akan memberi manfaat kepada pengguna, apapun perangkat yang digunakannya.

Dalam membangun perangkat lunak diperlukan metode pengembangan perangkat lunak. Metode ini berguna sebagai kerangka kerja yang menstrukturkan, merencanakan, dan mengendalikan proses pengembangan sistem informasi. Salah satu metode pengembangan perangkat lunak yang banyak digunakan, ialah *Agile*.

Agile adalah metode pengembangan perangkat lunak yang gesit dan berpusat pada gagasan pengembangan berulang (iteratif).

Extreme programming (XP) adalah salah satu metode pengembangan perangkat lunak berbasis *Agile*. XP bertujuan untuk menghasilkan perangkat lunak berkualitas tinggi, dan kualitas hidup yang lebih baik bagi tim pengembangan. Metodologi XP terutama dirancang untuk tim yang lebih kecil dengan dua hingga sepuluh anggota. XP menganjurkan rilis dalam siklus singkat untuk mendapatkan *feedback* dari pengguna dan meningkatkan produktivitas dimana respon pengguna akan diadopsikan di iterasi selanjutnya.

Metode XP akan digunakan pada pengembangan perangkat lunak repositori skripsi yang akan dibangun oleh penulis karena XP merupakan metode yang cocok untuk tim kecil, memiliki rencana iterasi yang fleksibel (dapat berubah sesuai kebutuhan), hanya mengerjakan fitur-fitur yang dianggap penting dan menambahkan fitur bila benar-benar dianggap perlu. Dengan demikian pembangunan perangkat lunak akan benar-benar terfokus pada kebutuhan. Rilis kecil pada tahapan XP juga memudahkan pengembang dalam mendapatkan *feedback* dan membangun perangkat lunak dengan cepat dan sesuai dengan kebutuhan pengguna.

Pada saat ini prodi Teknik informatika Unpad belum memiliki repositori maupun sarana yang memudahkan mahasiswa dalam mengakses karya sivitas akademika secara digital. Berdasarkan latar belakang yang telah dipaparkan, penulis mengusulkan rancangan web repositori skripsi Teknik Informatika Unpad yang dapat digunakan untuk mengakses skripsi mahasiswa Teknik Informatika

Unpad. Rancangan web ini dibangun dengan menggunakan kerangka kerja ReactJS dan konsep PWA dengan metode pengembangan perangkat lunak XP.

1.2 Identifikasi Masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, masalah yang akan dipecahkan dalam penelitian ini adalah bagaimana perancangan repositori institusi yang mengimplementasikan konsep PWA dengan metode pengembangan perangkat lunak XP yang dapat digunakan untuk memudahkan mahasiswa Teknik Informatika dalam mengakses skripsi.

1.3 Batasan Masalah

Dari permasalahan yang disampaikan sebelumnya, diberlakukan batasan masalah pada penelitian sebagai berikut:

1. Web aplikasi dibangun dengan metode pengembangan perangkat lunak XP.
2. Web aplikasi yang dibangun mengimplementasikan PWA fitur *add to homescreen* pada web sehingga dapat digunakan seperti *native app* dan fungsi tampilan saat *offline*.
3. Web aplikasi diakses menggunakan *browser* Chrome atau Mozilla dengan perangkat *desktop* maupun *smartphone* Android.
4. Target pengguna adalah mahasiswa Teknik Informatika Unpad dan Admin yaitu pegawai tata usaha Teknik Informatika.
5. Materi digital yang dapat diakses pada web repositori adalah skripsi mahasiswa Teknik Informatika Unpad.
6. Mahasiswa yang memiliki akun dapat mengunggah dan melihat *file* skripsi.

1.4 Maksud dan Tujuan Penelitian

Maksud dari penelitian ini adalah melihat hasil dari perancangan repositori skripsi Teknik Informatika Unpad yang mengimplementasikan PWA dengan metode pengembangan perangkat lunak XP.

Tujuan yang ingin dicapai oleh penulis dari penelitian ini adalah:

1. Dapat menghasilkan repositori institusi yang memudahkan mahasiswa Teknik Informatika Unpad untuk mengakses skripsi baik melalui *desktop* maupun *smartphone*.
2. Dapat mengimplementasikan metode XP pada pengembangan web repositori skripsi Teknik Informatika Unpad
3. Dapat mengimplementasikan PWA pada web repositori sehingga web aplikasi yang bertingkah seperti *native application* pada perangkat yang digunakan dan tetap menampilkan informasi disaat *offline*.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah :

1. Memudahkan mahasiswa Teknik Informatika Unpad untuk mengakses skripsi melalui *desktop* maupun *smartphone*.
2. Menambah pengetahuan dan pembelajaran mengenai metode XP pada pengembangan web repositori.
3. Menghasilkan web yang bertingkah seperti *native application* pada perangkat yang digunakan dan memiliki tampilan disaat *offline* dengan mengimplementasikan PWA.

1.6 Metodologi Penelitian

Jenis penelitian yang akan digunakan ialah *Research and Development*, yaitu penelitian dengan menerapkan langkah-langkah yang ada untuk menghasilkan sebuah produk perangkat lunak. Tahapan-tahapan yang akan dilalui adalah sebagai berikut:

1. Studi literatur. Pada tahap ini penulis mencari referensi dan informasi dari buku, jurnal, maupun artikel yang mendukung penelitian.
2. Penerapan metode XP. Pada tahap ini penulis membangun aplikasi sesuai dengan tahapan pada metode XP
 - a. Eksplorasi, yaitu menentukan kebutuhan dan membuat *user story*.
 - b. Perencanaan, yaitu menetapkan prioritas pengerjaan fitur.
 - c. Iterasi untuk dirilis, yaitu melakukan analisis, desain, dan *testing*.
 - d. Produksi, yaitu melakukan rilis kecil untuk mendapatkan *feedback*.
 - e. Pemeliharaan, yaitu melakukan pembaharuan dan rilis.
 - f. Final, yaitu melakukan rilis final.
3. Penulisan laporan. Pada tahap ini penulis menuliskan hasil penelitian ke dalam laporan skripsi

1.7 Sistematika Penulisan

Untuk memberi gambaran yang jelas tentang penelitian ini, maka disusunlah sistematika penulisan yang berisi materi yang akan dibahas pada setiap bab. Sistematika dalam penulisan tugas akhir ini adalah sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini dijelaskan tentang latar belakang dari topik penulisan skripsi, pokok permasalahan berupa identifikasi dan batasan masalah, tujuan dan manfaat yang diharapkan dari penulisan skripsi, metodologi yang digunakan serta sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Pada bab ini dijelaskan seluruh landasan teori yang berhubungan dengan penelitian, yaitu tentang metode pengembangan perangkat lunak yang digunakan, penjelasan teoritis mengenai bahasa pemrograman dan *framework* yang digunakan dalam proses pengimplementasian aplikasi, serta teori lainnya guna memahami permasalahan yang dibahas.

BAB III ANALISIS DAN PERANCANGAN

Pada bab ini dijelaskan tentang metode pengembangan aplikasi yang digunakan meliputi analisis kebutuhan sistem, perancangan aplikasi, diagram pemodelan sistem, model perancangan data dan rancangan antarmuka pengguna.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini dijelaskan tentang implementasi aplikasi yang telah dibangun, tampilan aplikasi, pengujian aplikasi, serta hasil dari penggunaan metode XP dan PWA pada web.

BAB V KESIMPULAN DAN SARAN

Bab ini merupakan penutup yang berisi kesimpulan dan saran dari penelitian yang sudah dilakukan

BAB II

TINJAUAN PUSTAKA

2.1 Repositori Institusi

Repositori adalah tempat penyimpanan sesuatu. Repositori biasanya merujuk pada perpustakaan atau tempat penyimpanan arsip. Repositori institusi adalah sekumpulan set layanan yang ditawarkan universitas kepada masyarakat untuk pengelolaan dan penyebaran dari materi yang dibuat oleh institusi tersebut (Repanovici, 2009).

Manfaat dari repositori institusi antara lain mengumpulkan karya ilmiah dalam suatu tempat agar mudah ditemukan kembali oleh mesin pencari seperti Google dan lainnya, sebagai sarana promosi, menyebarkan luaskan karya sivitas akademika dengan tempat dan waktu yang tidak terbatas (Sutedjo, 2014). Dengan demikian repositori institusi dapat mendukung penyebaran hasil penelitian baik dari mahasiswa maupun dosen.

2.2 *Extreme Programming*

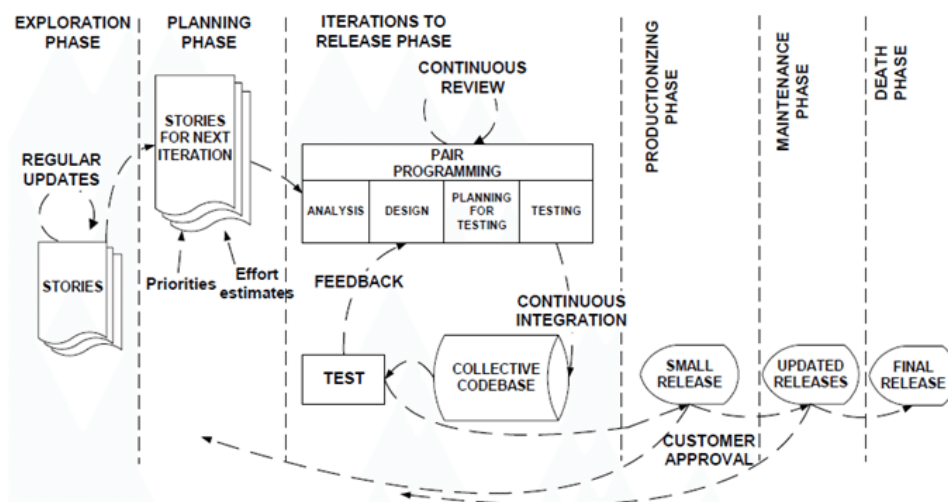
Extreme programming (XP) adalah metode pengembangan perangkat lunak yang diciptakan oleh Kent Beck. XP merupakan salah satu metode pengembangan *Agile*. Metode pengembangan ini cocok digunakan ketika tim dihadapkan dengan *requirement* yang tidak jelas maupun jika terjadi perubahan *requirement* yang sangat cepat. (Prabowo, dkk., 2013).

XP memiliki siklus hidup yang mencakup 6 fase yaitu: eksplorasi, perencanaan, iterasi untuk dirilis, produksi, pemeliharaan, dan akhir. Pada fase

eksplorasi, kebutuhan pengguna dikumpulkan dan *user story* dibuat. Pada fase perencanaan, tim pengembang memberikan prioritas pengerjaan pada *user story* yang sudah dibuat. Penetapan prioritas didasari oleh hal berikut:

1. Semua *story* segera diimplementasikan (dalam beberapa minggu)
2. *Story* dengan *value* tertinggi akan dipindahkan dari jadwal dan diimplementasikan pertama.
3. *Story* dengan resiko paling tinggi akan diimplementasikan terlebih dulu.

Fase iterasi untuk dirilis adalah fase yang paling penting. Pada fase ini analisis, desain, dan pengujian ini berlangsung melalui *pair programming*. Tiga tahapan tersebut dilakukan secara berurut dan berulang. Pada fase produksi, rilisan kecil ditunjukkan kepada pelanggan untuk diminta *feedback* dan persetujuan dari sistem yang dibangun. Pada fase pemeliharaan, pembaruan proyek dilakukan berdasarkan *feedback* yang didapat dari fase produksi. Kemudian dirilis dengan persetujuan pelanggan. Terakhir, pada fase akhir, produk final akan dirilis (Krishna *et al.*, 2011).



Gambar 2.1 Siklus Hidup XP (Krishna *et al.*, 2011)

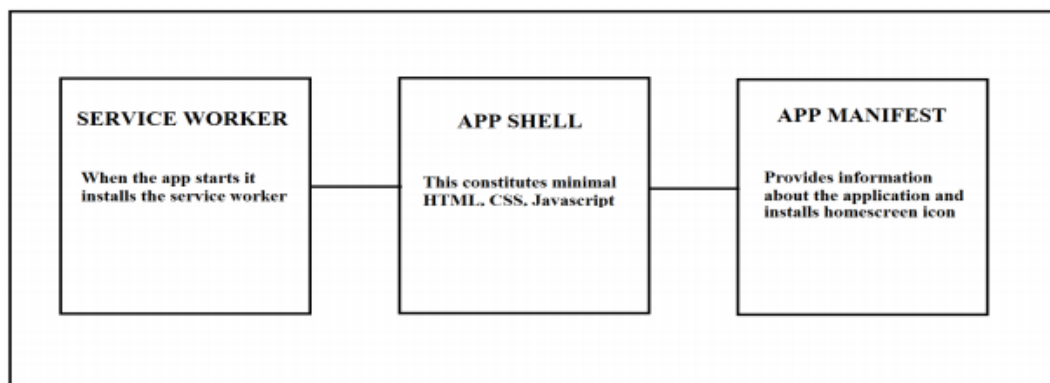
XP memiliki nilai-nilai yaitu kesederhanaan (*simplicity*), komunikasi (*communication*), umpan balik (*feedback*), dan keberanian (*courage*). Metode pengembangan ini juga memiliki manfaat yaitu: fokus terhadap pelanggan, ditekankan pada kerja tim, komunikasi, dan tanggung jawab terhadap kualitas, pengukuran berkelanjutan, pengembangan bertahap, desain sederhana, tinjauan berkelanjutan. Penerapan XP dapat menghasilkan aplikasi dalam kurun waktu lebih cepat dengan jumlah anggota tim yang sedikit.

2.3 *Progressive Web Apps*

Aplikasi Web Progresif (PWA) adalah ide yang pertama kali didukung oleh insinyur Google, Alex Russell pada Juni 2015 (Karpagam *et al.*, 2017). Aplikasi web progresif adalah generasi baru aplikasi web yang menggabungkan manfaat aplikasi asli dengan dengan keunggulan web. PWA mengubah situs web menjadi sesuatu yang lebih seperti aplikasi tradisional asli. (Ater, 2017:15).

Selain memiliki tampilan seperti *native app*, PWA memiliki keunggulan lain seperti tidak bergantung pada koneksi seperti situs web tradisional. Dengan menggunakan *service worker*, web dapat melakukan *cache* bagian situs secara selektif untuk memberikan pengalaman walaupun disaat sedang *offline*, *online*, atau pada koneksi yang buruk (Hume, 2017:5). Dengan *service worker*, web dapat di-*load* dengan lebih cepat. PWA juga dapat menambahkan *shortcut ke homescreen* perangkat. PWA menunjuk ke *file* yang dikenal sebagai *file* manifes yang berisi informasi tentang situs web, termasuk ikonnya, layar latar belakang, warna, dan orientasi standar.

PWA memiliki 3 prinsip utama yaitu: *service worker*, *app shell*, dan *app manifest*. *Service worker* menyediakan fungsionalitas *offline*, *push notification*, pembaruan konten latar belakang, *caching* konten, dan banyak lagi lainnya. *App shell* fokus dalam menjaga *shell* UI aplikasi dan konten di dalamnya terpisah, dan di-*cache* secara terpisah. *App manifest* menyediakan kemampuan untuk menyimpan *bookmark* situs ke layar beranda perangkat tanpa meng-*install* seperti aplikasi asli (Karpagam *et al.*, 2017).



Gambar 2.2 Prinsip Utama PWA (Karpagam *et al.*, 2017)

2.4 ReactJS

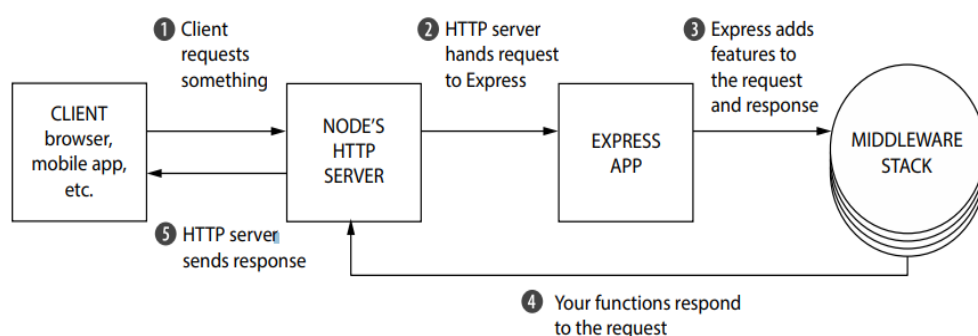
ReactJS adalah kerangka kerja (*framework*) *open-source* yang fleksibel dan kuat untuk mengembangkan aplikasi sisi klien. ReactJS membutuhkan isyarat dari pengembangan sisi *server* dan menerapkannya pada elemen HTML, hal itu menciptakan pondasi yang memudahkan *developer* membangun aplikasi web (Freeman, 2019:31).

ReactJS merupakan teknologi web sisi klien berbasis JavaScript yang dikembangkan oleh Facebook. Web yang dibangun akan memiliki UI yang dibagi kedalam beberapa komponen (*component based*). ReactJS memiliki keunggulan

yaitu cepat dan efisien karena berbasis komponen sehingga ReactJS hanya perlu *re-render resource* yang berhubungan dengan data yang berganti, tanpa perlu *re-render* keseluruhan halaman. ReactJS juga menciptakan Virtual DOM untuk mempercepat urusan perubahan DOM (*Document Object Model*). Semua operasi dikerjakan di dalam Virtual DOM, setelah operasi selesai ReactJS akan menulis perubahan tersebut di dalam DOM.

2.5 Express

Express adalah kerangka kerja relatif kecil yang berada di atas fungsionalitas *server* web Node.js untuk menyederhanakan API dan menambahkan fitur baru yang bermanfaat. Express memudahkan pengaturan fungsionalitas aplikasi dengan *middleware* (fungsi-fungsi penanganan permintaan yang lebih kecil) dan perutean, menambah utilitas untuk objek HTTP Node.js, memfasilitasi rendering tampilan HTML dinamis, mendefinisikan standar yang mudah diimplementasikan (Hahn, 2016:6).



Gambar 2.3 Alur *Request* pada Express (Hahn, 2016:7)

Alih-alih membuat satu fungsi penanganan permintaan monolitik yang besar, Express meminta *developer* untuk menulis banyak fungsi yang lebih kecil (banyak di antaranya bisa merupakan fungsi pihak ketiga). Kemudian fungsi dieksekusi

untuk setiap permintaan. Fungsi-fungsi penanganan permintaan yang lebih kecil ini disebut fungsi *middleware*. Express meminimaliskan pengkodean pada Node.js dengan menambahkan fitur-fitur. Hal ini membuat kerja *developer* menjadi lebih mudah dan sederhana. Express meningkatkan NodeJS seperti halnya Bootstrap untuk HTML / CSS.

2.6 Javascript

Javascript adalah bahasa *scripting* yang ditafsirkan (*interpreted scripting language*) dengan kata lain Javascript adalah jenis bahasa pemrograman yang memungkinkan kode yang dibuat langsung dijalankan sebagai program secara dinamis. Bahasa yang ditafsirkan memiliki interpreter yang akan menjalankan program secara langsung dari *source code* yang dibuat, menerjemahkan baris per baris menjadi kode mesin dan langsung mengeksekusi kode pada saat itu juga. Tidak seperti *compiled language* yang menerjemahkan seluruh *source code* menjadi kode mesin dalam sekali proses kompilasi dan menghasilkan *executable file*, *interpreted language* tidak menghasilkan *executable file*.

JavaScript adalah bagian dari tiga serangkai teknologi yang harus dipelajari oleh semua pengembang web: HTML untuk menentukan konten halaman web, CSS untuk menentukan presentasi halaman web, dan JavaScript untuk menentukan perilaku halaman web. Contoh utama JavaScript adalah kemampuan untuk menambahkan interaktivitas ke situs web. Hal tersebut dapat terjadi karena interpreter tertanam ke dalam *web browser*. Oleh karena itu JavaScript menjadi bahasa yang mudah diakses karena hanya membutuhkan teks editor dan *browser* (Ferguson, 2019:3)

Pada buku *Beginning JavaScript. The Ultimate Guide to Modern JavaScript Development* (Ferguson, 2019:3) dijelaskan bahwa, Javascript pada sisi klien dapat menambahkan tingkat interaktivitas seperti merespons klik tombol, memvalidasi konten formulir dan menggunakan *application programming interfaces* (APIs) yang dibangun ke dalam *browser*. Kasus penggunaan lain adalah untuk mengeksekusi JavaScript di *server*, menggunakan lingkungan seperti Node.js. Contoh JavaScript sisi *server* adalah kemampuan untuk melakukan hal-hal seperti membuat permintaan dari *database* dan menanggapi permintaan HTTP dan membuat *file*.

2.7 Bootstrap

Bootstrap adalah produk *open source* dari Mark Otto dan Jacob Thornton. Keduanya adalah karyawan Twitter pada saat merilis Bootstrap di tahun 2011. Pada saat itu dibutuhkan standarisasi perangkat *frontend* untuk semua insinyur di perusahaan. Bootstrap dibangun untuk mengatasi ketidakkonsistenan di antara aplikasi individual yang menjadi hambatan dalam mengukur dan memelihara aplikasi.

Bootstrap adalah kerangka kerja *frontend* untuk mengembangkan situs dan aplikasi web yang responsif dan mengimplementasikan *mobile-first design*. Bootstrap adalah kombinasi HTML, CSS, dan kode JavaScript untuk membangun komponen antarmuka pengguna yang dapat dipanggil melalui kelas. Bootstrap menyediakan sistem grid 12 kolom yang responsif dan kelas yang telah ditentukan sebelumnya yang memudahkan tata letak (*layouting*). Bootstrap memiliki lusinan komponen *prestyled* yang dapat digunakan kembali dan *plugin* jQuery khusus,

seperti tombol, peringatan, *dropdown*, modal, *pagination*, *carousal*, *badge*, dan ikon (Singh and Bhatt, 2016:13-14). Bootstrap memberikan kemudahan kepada pengembang dalam membangun web tanpa menambahkan banyak *code*.

2.8 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data (DBMS) yang open source dan menggunakan bahasa SQL (*Structured Query Language*). MySQL termasuk ke dalam jenis RDBMS (*Relational Database Management System*) sehingga dalam implementasinya menggunakan istilah baris, kolom, dan tabel. Dalam sebuah *database* MySQL terdapat beberapa tabel untuk menyimpan data dimana tiap tabel memiliki relasi satu sama lain sehingga data dari beberapa tabel dapat diambil dan dikombinasikan. Sistem semacam inilah yang disebut dengan RDBMS (Nadia, dkk., 2018)

Sistem manajemen basis data MySQL populer karena berbagai alasan yaitu MySQL terkenal karena kinerjanya, kemudahan untuk digunakan, dan keandalannya. MySQL menjadi pilihan paling umum untuk basis data relasional. Ribuan aplikasi berbasis web mengandalkan MySQL termasuk industri raksasa seperti Facebook, Twitter, dan Wikipedia. MySQL sangat fleksibel dalam hal platform, seperti RedHat, Fedora, Ubuntu, Debian, Solaris, Microsoft Windows, dan sebagainya. Ini memiliki dukungan dari API untuk terhubung dengan berbagai bahasa, seperti C, C ++, C #, PHP, Java, Ruby, dan banyak lagi. (Mehta, *et al.*, 2018)



2.9 Unified Modeling Language (UML)


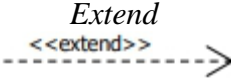

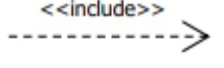
Unified Modeling Language (UML) adalah bahasa visual untuk pemodelan sistem dengan menggunakan diagram dan teks-teks pendukung. UML muncul karena adanya kebutuhan standardisasi untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML terbaru adalah UML 2.3 yang terdiri dari 13 macam diagram dan 3 kategori yaitu: *structure diagrams*, *behavior diagrams*, dan *intraction diagrams* (A.S dan Shalahuddin, 2018:137-140). Pada subbab ini penulis hanya akan membahas 3 macam diagram yang digunakan dalam penelitian.

2.9.1 Use Case Diagram

Use case diagram adalah salah satu *behavior diagram* yaitu diagram yang menggambarkan ciri-ciri tingkah/metode/fungsi dari sebuah sistem. Diagram ini menggambarkan aktor, *use case* dan relasinya sebagai suatu urutan tindakan yang memberikan nilai terukur untuk aktor. Sebuah *use case* digambarkan sebagai elips horizontal dalam suatu diagram UML *use case* (Ropianto, 2016). Perhatikan Tabel 2.1 untuk penjelasan dari simbol-simbol *use case diagram*.

Tabel 2.1 Simbol-Simbol pada *Use Case Diagram*


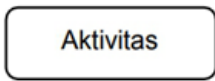
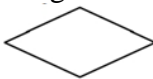
No	Simbol	Keterangan
1	<p><i>Use Case</i></p> 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor
2	<p><i>Actor</i></p> 	Orang atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat. Biasanya dinyatakan menggunakan kata benda atau nama aktor.



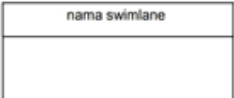
No	Simbol	Keterangan
3		Komunikasi antara aktor dengan <i>use case</i> yang berpartisipasi pada diagram
4		Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> , dimana <i>use case</i> yang ditambahkan dapat berdiri sendiri meski tanpa <i>use case</i> tambahan itu. Arah panah mengarah pada <i>use case</i> yang ditambahkan.
5		Hubungan generalisasi dan spesialisasi (umum-khusus) antar dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
6		Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> , dimana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsinya atau sebagai syarat dijalankan <i>use case</i> ini. Arah panah <i>include</i> mengarah pada <i>use case</i> yang dibutuhkan

2.9.2 Activity Diagram

Menggambarkan aktifitas-aktifitas, objek, *state*, transisi *state* dan *event*. Dengan kata lain, aktivitas diagram menggambarkan perilaku sistem, alur kerja, atau aktivitas sistem. Tabel 2.2 berisi penjelasan dari simbol-simbol *activity diagram*.

Tabel 2.2 Simbol-Simbol pada *Activity Diagram*




No	Simbol	Keterangan
1	Status Awal 	Menandakan tindakan awal atau titik awal aktivitas untuk setiap diagram aktivitas.
2	Aktivitas 	Menunjukkan aktivitas yang dilakukan sistem
3	Percabangan / <i>decision</i> 	Asosiasi percabangan adalah keadaan dimana terdapat pilihan aktivitas lebih dari satu.



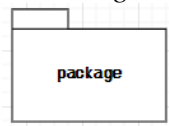
No	Simbol	Keterangan
4	Penggabungan / <i>join</i> 	Asosiasi penggabungan adalah keadaan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5	<i>Status akhir</i> 	Menunjukkan bagian akhir dari aktivitas.
6	<i>Swimlane</i> 	<i>Swimlane</i> memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.9.3 Deployment Diagram

Diagram *deployment* atau *deployment diagram* menunjukkan konfigurasi komponen dalam suatu proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan sistem tambahan seperti rancangan *device*, *node*, dan *hardware*, sistem *client/server*, sistem terdistribusi murni, dan rekayasa ulang aplikasi (A.S dan Shalahuddin, 2018:154)

Tabel 2.3 Simbol-Simbol pada *Deployment Diagram*

No	Simbol	Keterangan
1	<i>Package</i> 	<i>Package</i> merupakan sebuah bungkus dari satu atau lebih node.
2	<i>Node</i> 	<i>Node</i> mengacu pada perangkat keras atau perangkat lunak yang tidak dibuat sendiri.
3	<i>Artifact</i> 	Elemen perangkat lunak atau produk yang dihasilkan oleh perangkat lunak. Berada didalam <i>node</i>






No	Simbol	Keterangan
4		Relasi antar <i>node</i>
5	Kebergantungan / <i>dependency</i> 	Ketergantungan antar <i>node</i> , arah panah mengarah pada node yang dipakai
6		<i>Package</i> merupakan sebuah bungkus dari satu atau lebih node.

2.10 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) diagram yang menggambarkan dan menjelaskan skema pemodelan basis data. Merupakan bentuk paling awal dalam melakukan perancangan basis data relasional. Umumnya setelah perancangan ERD selesai berikutnya adalah mendesain *database* secara fisik yaitu pembuatan tabel. ERD dapat memiliki hubungan hubungan *binary* (relasi yang menghubungkan dua entitas), *ternary* (relasi dengan banyak entitas), atau *N-ary* (relasi banyak entitas) namun pada umumnya ERD memiliki hubungan *binary*. (A.S dan Shalahuddin, 2018:50-52)

ERD memiliki kardinalitas yaitu jumlah himpunan relasi antar entitas. Pemetaan kardinal terdiri dari; *one-to-one*, yaitu satu entitas A hanya dapat terhubung dengan 1 anggota entitas B, *one-to-many*, yaitu satu anggota entitas A hanya dapat terhubung dengan 1 anggota entitas B sementara 1 anggota entitas B dapat terhubung dengan beberapa anggota entitas A, dan *many-to-many*, yaitu masing-masing anggota entitas A dapat terhubung dengan beberapa anggota entitas B.

Tabel 2.4 Simbol-simbol pada ERD

No	Simbol	Keterangan
1	Entitas/ <i>Entity</i> 	Entitas mengacu pada objek yang akan disimpan datanya. Entitas akan menjadi sebuah tabel
2	Atribut 	Atribut mengacu kolom data yang perlu disimpan pada tabel entitas.
3	Atribut <i>primary key</i> 	Atribut <i>primary key</i> adalah kolom data yang digunakan digunakan untuk mengakses <i>record</i> . Atribut <i>primary key</i> bersifat unik
4	Relasi 	Relasi adalah hubungan antar entitas. Merupakan kata kerja
5	Asosiasi 	Penghubung antara atribut dengan entitas dan entitas dengan relasi

2.11 Black Box Testing

Black Box Testing merupakan teknik pengujian perangkat lunak yang berfokus pada spesifikasi fungsional dari perangkat lunak. Dalam *black box testing*, struktur bagian tidak diketahui atau tidak dipertimbangkan. Pengujian didapat dari deskripsi eksternal perangkat lunak, termasuk spesifikasi, persyaratan, dan desain. (Spillner, *et al.*, 2014 : 110). Pengujian dapat dilihat dalam hal input dan outputnya (atau karakteristik transfer), tanpa mengetahui cara kerjanya. *Black Box testing* memungkinkan pengembang perangkat lunak untuk membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program (Jaya, 2018).

Tes dengan semua kemungkinan kombinasi data input akan menjadi tes yang lengkap, tetapi ini tidak realistis karena banyaknya kombinasi. Maka selama pengujian, subset wajar dari semua kasus uji yang mungkin harus dipilih (Spillner, Linz and Schaefer, 2014:110).

Keuntungan penggunaan metode *black box testing* ialah pengujian dilakukan dari sudut pandang pengguna, sehingga dapat mengungkap ambiguitas atau inkonsistensi. Hal ini juga bermanfaat dalam menangani macam-macam skenario yang dapat dilakukan pengguna terhadap perangkat lunak.

2.12 Usability Testing

Usability atau ketergunaan adalah tingkat kualitas dari perangkat lunak yang mudah dipelajari, mudah digunakan, dan mendorong pengguna untuk menggunakan sistem sebagai alat bantu positif dalam menyelesaikan tugas. Terdapat lima unsur yang menjadi pokok *usability*, yaitu kegunaan, efisiensi, efektivitas, kepuasan, dan aksesibilitas (Handiwidjojo dan Ernawati², 2016)

Agar web mencapai tingkat ketergunaan yang ideal, terdapat 5 syarat yang harus dipenuhi, yaitu *learnability* (mudah dipelajari), *efficiency* (Efisien), *memorability* (kemudahan dalam mengingat), *errors* (pencegahan kesalahan), dan *satisfaction* (kepuasan pengguna).

Usability testing yang dilakukan akan menggunakan skala Likert untuk menghitung hasilnya. Skala Likert adalah suatu skala psikometrik yang umum digunakan dalam angket dan merupakan skala yang paling banyak digunakan dalam riset berupa survei (Maryuliana, dkk., 2016). Sewaktu menanggapi pertanyaan dalam skala Likert, responden menentukan tingkat persetujuan mereka terhadap

suatu pernyataan dengan memilih salah satu dari pilihan yang tersedia. Biasanya disediakan lima pilihan skala dengan format seperti Tabel 2.5:

Tabel 2.5 Titik Respon

Pilihan Titik Respon	Nilai
1. Sangat Buruk	1 poin
2. Buruk	2 poin
3. Cukup	3 poin
4. Bagus	4 poin
5. Sangat Bagus	5 poin

Maka, hasil penilaian dari kuesioner tersebut dapat dihitung dengan menentukan interval dan kriteria nilai terlebih dahulu dengan menggunakan Persamaan 2.1:

$$Interval = \frac{Nilai\ tertinggi}{Jumlah\ titik\ respon}$$

Persamaan 2.1 Interval

Berikutnya, untuk menentukan total skor dari setiap butir pertanyaan dapat menggunakan Persamaan 2.2:

$$Skor\ Kriteria = (Jumlah \times Nilai\ 1) + (Jumlah \times Nilai\ 2) + (Jumlah \times Nilai\ 3) + (Jumlah \times Nilai\ 4) + (Jumlah \times Nilai\ 5)$$

Persamaan 2.2 Total Skor

Karena nilai tertinggi dari pilihan titik respon adalah 5, maka dari itu interpretasi nilai hasil dapat dihitung dengan menggunakan Persamaan 2.3:

$$Nilai\ akhir\ (\%) = \frac{Total\ skor}{Jumlah\ responden} \times Interval$$

BAB III

ANALISIS DAN PERANCANGAN

Metode pengembangan perangkat lunak yang digunakan pada penelitian ini ialah *Extreme programming*. Pengembangan aplikasi web pada penelitian ini melalui tahapan-tahapan XP yang terdiri dari 6 fase yaitu fase eksplorasi, fase perencanaan, fase iterasi untuk dirilis, fase produksi, fase pemeliharaan, dan fase akhir.

3.1 Fase Eksplorasi

Pada fase eksplorasi, tujuan utama yang ingin dicapai ialah membuat *user story*. Untuk memudahkan proses pembuatan *user story*, dilakukan pengumpulan kebutuhan (*requirement*) terlebih dahulu. Penulis mengumpulkan kebutuhan untuk membangun sistem repositori skripsi Teknik Informatika Unpad dan membuat *user story* berdasarkan kebutuhan tersebut. Secara bersamaan pula pengembang mempersiapkan teknologi yang akan digunakan.

Kebutuhan pada penelitian ini dibagi menjadi empat bagian yaitu, kebutuhan pengguna, kebutuhan data, kebutuhan perangkat lunak, kebutuhan perangkat keras, dan kebutuhan sistem. Kebutuhan-kebutuhan tersebut dirancang berdasarkan tujuan dan batasan-batasan yang telah dijelaskan pada BAB I. Kebutuhan sistem dikumpulkan dengan cara melakukan survei menggunakan *google form* kepada mahasiswa Teknik Informatika Unpad. Setelah kebutuhan dikumpulkan, *user story* akan dibuat.

3.1.1 Kebutuhan Pengguna

Pada tahapan ini dilakukan perencanaan mengenai target pengguna web aplikasi repositori skripsi Teknik informatika Unpad. Target pengguna web aplikasi ialah mahasiswa Teknik Informatika Unpad. Akun yang terdaftar harus memenuhi syarat dan terbukti sebagai mahasiswa Teknik nformatika Unpad, oleh karena itu diperlukan admin untuk menangani verifikasi data. Maka web aplikasi memiliki 2 jenis pengguna yaitu:

1. Mahasiswa Teknik Informatika Unpad yang memiliki perangkat dan *browser* yang mendukung untuk mengakses web repositori skripsi dengan fitur PWA.
2. Admin web yaitu pegawai tata usaha Teknik Informatika Unpad yang bertugas untuk menverifikasi data pada web. Admin harus menggunakan *browser* pada *desktop* maupun perangkat Android dengan *browser* yang mendukung fitur PWA.

3.1.2 Kebutuhan Data

Data yang diperlukan agar web repositori skripsi dapat berjalan sesuai dengan fungsinya ialah sebagai berikut:

1. Data diri mahasiswa yang terdiri dari nama, NPM, email, foto KTM, dan *password*. Dan data admin terdiri dari nama, *username*, dan *password*
2. Data skripsi yang terdiri dari judul, penulis, tahun dipublikasi, abstrak dalam dua bahasa, berkas skripsi, kategori skripsi, dan kata kunci seputar topik skripsi.

3.1.3 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak yang digunakan pada proses pengembangan adalah sebagai berikut:

1. Visual Studio Code 1.41.1, yaitu teks editor yang digunakan untuk pengkodean.
2. ReactJS, yaitu *framework frontend* yang digunakan untuk membangun aplikasi web.
3. Express, yaitu *framework* Node.JS yang digunakan sebagai *backend* web.
4. Mysql, yaitu *database* SQL yang digunakan pada sistem
5. Dbeaver, yaitu sebuah SQL *client* dan alat administrasi basis data yang digunakan untuk mengakses *database*
6. Lighthouse, yaitu alat bantu untuk meningkatkan kualitas aplikasi web yang dapat dijalankan sebagai ekstensi Chrome. Lighthouse menjalankan serangkaian pengujian terhadap web, kemudian menghasilkan sebuah laporan mengenai seberapa bagus laman itu menjalaninya. Lighthouse digunakan untuk menguji implementasi PWA pada web.

Sementara perencanaan kebutuhan perangkat lunak bagi pengguna web reopsitori skripsi adalah *browser* yang kompatibel dengan fitur PWA, yaitu Google Chrome 73 atau lebih tinggi, Mozilla Firefox 58 atau lebih tinggi.

3.1.4 Kebutuhan Perangkat Keras

Perencanaan kebutuhan perangkat keras minimum bagi pengguna agar web aplikasi berjalan dengan baik. Berdasarkan gambar 3.1 iOS tidak mendukung *service worker* sehingga tidak dapat menjalankan fitur PWA. Diambil kesimpulan

bahwa perangkat keras yang dapat digunakan untuk mengoperasikan web repositori skripsi Teknik Informatika Unpad dengan fungsi PWA adalah sebagai berikut:

1. Perangkat Android dengan sistem Nougat 7.0 menggunakan *browser* Mozilla atau Chrome
2. Desktop dengan sistem operasi Windows, MacOS, dan Linux dengan menggunakan *browser* Mozilla atau Chrome

	MOBILE	DESKTOP			
	Android	iOS	Windows	macOS	Linux
PWA Progressive Web Apps	✓	✓	✓	✓	✓
Service Worker	✓	✗	✓	✓	✓

Gambar 3.1 Kompatibilitas *Browser* Chrome dan Mozilla pada Perangkat

(Santoni, 2018)

3.1.5 Kebutuhan Sistem

Pada tahapan ini dilakukan survei kepada target pengguna yaitu mahasiswa Teknik Informatika Unpad untuk mendapatkan kebutuhan sistem repositori skripsi yang diinginkan. Dari hasil survei dirancang fitur-fitur seperti yang tertera pada tabel 3.1 dan tabel 3.2.

Tabel 3.1 Kebutuhan Sistem Pengguna: Mahasiswa

Pengguna : Mahasiswa	
Fitur	Deskripsi
Registrasi	Pengguna mendaftarkan diri dengan menginput nama, NPM, <i>email</i> , <i>password</i> , dan foto KTM.
<i>Login</i>	Pengguna <i>login</i> dengan menginput NPM dan <i>password</i>
Cari Skripsi	Pengguna dapat mencari skripsi berdasarkan judul, nama penulis, kata kunci terkait, dan menyaring skripsi berdasarkan tahun atau kategori skripsi
Melihat Skripsi	Pengguna yang telah <i>login</i> dapat melihat <i>file</i> skripsi
Unggah Skripsi	Pengguna dapat mengunggah skripsi
Edit <i>Password</i>	Pengguna dapat mengubah kata sandi pada halaman profil
Status Skripsi	Pengguna dapat mengecek status skripsi apakah telah disetujui admin atau tidak
Tampilan <i>offline</i>	Web tetap memiliki tampilan pada saat koneksi <i>offline</i>
<i>Add to homescreen</i>	Web dapat ditambahkan ke <i>homescreen</i> dan digunakan seperti <i>native app</i>
Lupa <i>Password</i>	Pengguna mendapatkan kata sandi baru yang dikirimkan melalui <i>email</i>
Kontak admin	Pengguna dapat mengirim pesan ke admin

Tabel 3.2 Kebutuhan Sistem Pengguna: Admin

Pengguna : Admin	
Fitur	Deskripsi
<i>Login</i>	Pengguna <i>login</i> dengan menginput <i>username</i> dan <i>password</i> yang terdaftar
Tinjau Unggahan Skripsi	Pengguna dapat mengecek unggahan skripsi dan menghapus atau menyetujui unggahan
Verifikasi Akun Mahasiswa	Pengguna mengecek data registrasi mahasiswa dan memberi aktivasi terhadap akun yang memenuhi persyaratan registrasi.
Kontak mahasiswa	Pengguna dapat mengirim pesan ke mahasiswa

Kolom dengan warna abu-abu adalah kebutuhan yang ditambahkan pada fase pemeliharaan karena adanya penambahan fitur dari hasil *feedback*. Maka, kebutuhan tersebut tidak masuk dalam pengujian kepada pengguna pada fase produksi.

3.1.6 User Story

Pada tahapan ini, penulis membuat *user story* yang merupakan inti dari fase eksplorasi. Kebutuhan yang telah ditentukan sebelumnya membuat proses pembentukan *user story* menjadi lebih mudah. *User story* menghasilkan gambaran fitur-fitur yang pada web repositori skripsi Teknik Informatika Unpad. Secara umum *user story* terdiri dari tiga bagian, yaitu judul, deskripsi, dan *acceptance criteria*.

Tabel 3.3 *User Story*

Judul	Deskripsi	<i>Acceptance Criteria</i>
Register	Sebagai mahasiswa, saya ingin mendaftarkan diri agar memiliki akses terhadap aplikasi web	<ul style="list-style-type: none"> -Terdapat menu registrasi untuk mahasiswa memperoleh akun -Mahasiswa mengisi nama lengkap, NPM, <i>email</i>, <i>password</i>, dan foto KTM -Mahasiswa akan menerima <i>email</i> verifikasi -Akun akan terdaftar apabila verifikasi <i>email</i> telah dilakukan -Akun tidak dapat digunakan apabila belum diaktifkan -Akun diaktifkan oleh admin -Mahasiswa mengetahui akun telah diaktifkan melalui <i>email</i>
Login	Sebagai pengguna, saya ingin mengakses fitur yang tersedia pada aplikasi web	<ul style="list-style-type: none"> -Mahasiswa mengisi NPM dan <i>password</i> pada kolom login -Admin mengisi <i>username</i> dan <i>password</i> yang terdaftar -Sistem melakukan validasi terhadap data yang diinputkan -Pengguna mendapatkan akses ke fitur-fitur pada web aplikasi
Mencari Skripsi	Sebagai mahasiswa, saya ingin mencari skripsi berdasarkan judul, penulis, tahun, dan kategori skripsi	<ul style="list-style-type: none"> -Mahasiswa mengisi kata kunci pada kolom pencarian untuk mencari skripsi -Mahasiswa dapat menyaring skripsi berdasarkan tahun atau kategori
Unggah Skripsi	Sebagai mahasiswa, saya ingin mengunggah	<ul style="list-style-type: none"> -Fitur hanya dapat diakses oleh pengguna yang telah login

Judul	Deskripsi	Acceptance Criteria
	skripsi miliki saya ke web	-Terdapat <i>form</i> unggah skripsi. Mahasiswa mengisi identitas skripsi, dan mengunggah <i>file</i> skripsi. -Unggahan tidak muncul dihalaman utama melainkan ditinjau terlebih dahulu oleh admin
Tinjau Unggahan	Sebagai admin, saya ingin meninjau skripsi yang diunggah oleh mahasiswa	-Fitur hanya dapat diakses oleh admin -Terdapat list skripsi yang diunggah oleh mahasiswa -Admin dapat menyetujui atau menolak unggahan.
Verifikasi Akun Mahasiswa	Sebagai admin, saya ingin mengecek akun-akun yang terdaftar	-Fitur hanya dapat diakses oleh admin -Admin dapat melihat data akun yang terdaftar. -Admin dapat menghapus akun atau mengaktifkan akun yang memenuhi persyaratan -Pemberitahuan bahwa akun telah diaktifkan atau tidak diaktifkan akan dikirimkan melalui <i>email</i>
Edit Password	Sebagai mahasiswa, saya ingin mengubah kata sandi	-Fitur hanya dapat diakses oleh pengguna yang telah login -Mahasiswa dapat melihat halaman profil dan mengakses menu <i>edit password</i> -Mahasiswa dapat mengubah kata sandi
Status Skripsi	Sebagai mahasiswa, saya ingin mengetahui apakah skripsi yang saya unggah telah dipublikasikan oleh admin	-Fitur hanya dapat diakses oleh pengguna yang telah login -Mahasiswa dapat melihat status skripsi yang diunggah
PWA (<i>offline</i>)	Sebagai pengguna saya ingin web aplikasi tetap mempertahankan tampilan dan memberikan informasi saat koneksi <i>offline</i>	-Web tetap memiliki tampilan disaat <i>offline</i> dan menampilkan info bahwa pengguna sedang <i>offline</i>
PWA (<i>add to homescreen</i>)	Sebagai pengguna saya ingin menambahkan web pada <i>homescreen</i> dan menggunakannya seperti <i>native app</i>	-Pengguna dapat menambahkan web ke <i>homescreen</i> dan menggunakannya seperti <i>native app</i>
Lupa Password	Sebagai mahasiswa yang lupa <i>password</i>	-Mahasiswa menginputkan <i>email</i> dan NPM yang terdaftar

Judul	Deskripsi	Acceptance Criteria
	saya ingin melakukan permintaan untuk mengakses akun dengan <i>password</i> baru.	-Mahasiswa menerima <i>email</i> berisi <i>password</i> baru yang dapat digunakan untuk <i>login</i>
Kontak Admin & Mahasiswa	Sebagai pengguna saya ingin mengontak admin/mahasiswa melalui web	-Fitur hanya dapat diakses oleh pengguna yang telah login -Mahasiswa dapat mengirim pesan ke admin -Admin dapat mengirim pesan ke mahasiswa

Kolom dengan warna abu-abu adalah *user story* yang ditambahkan pada fase pemeliharaan karena adanya penambahan fitur dari hasil *feedback*. *User story* tersebut diimplementasikan pada fase pemeliharaan sehingga tidak masuk dalam pengujian kepada pengguna pada fase produksi.

3.2 Fase Perencanaan

Pada fase kedua, penulis memberikan prioritas pengerjaan kepada fitur-fitur yang telah dibentuk pada *user story* di fase eksplorasi. Prioritas tinggi diberikan pada fitur dengan nilai dan resiko paling tinggi. Fitur dengan prioritas tinggi diimplementasikan terlebih dulu. Halaman utama akan diimplementasikan pertama karena pada saat mengakses web, halaman inilah yang pertama kali akan muncul.

Registrasi dan *login* adalah fitur dengan resiko paling tinggi karena berhubungan dengan autentikasi akun dan hak akses ke fitur web. *Login* membutuhkan token dan proses membedakan jenis user. Maka register dan *login* akan diimplementasikan terlebih dahulu. Selanjutnya adalah fitur unggah skripsi. Unggah skripsi membutuhkan validasi masukan dan *error handling* yang baik agar berkas yang diterima sistem sesuai format dan tidak berulang.

Fitur verifikasi akun dan tinjau skripsi memiliki nilai yang tinggi karena berurusan dengan validasi data, maka kedua fitur ini diimplementasikan setelah register, *login*, dan unggah skripsi. Kemudian fitur pratinjau skripsi, fitur pencarian, *edit password*, fitur PWA, kontak admin, dan lupa *password* dan diimplementasikan setelahnya.

Tabel 3.4 Prioritas Fitur

No	Prioritas Fitur
1	Halaman utama / <i>Landing page</i>
2	Registrasi
3	<i>Login</i> (Autentikasi)
4	Unggah berkas
5	Verifikasi akun (Admin)
6	Tinjau skripsi (Admin)
7	Menampilkan pratinjau skripsi
8	Pencarian dan filter
9	Edit password
10	Status Skripsi
11	PWA (tampilan saat <i>offline</i> dan <i>add to screen shortcut</i>)
12	Kontak Admin
13	Lupa <i>password</i>

Kolom dengan warna abu-abu adalah fitur yang ditambahkan pada fase pemeliharaan dari hasil *feedback*. Fitur tersebut diimplementasikan pada fase pemeliharaan sehingga tidak masuk dalam pengujian kepada pengguna pada fase produksi.

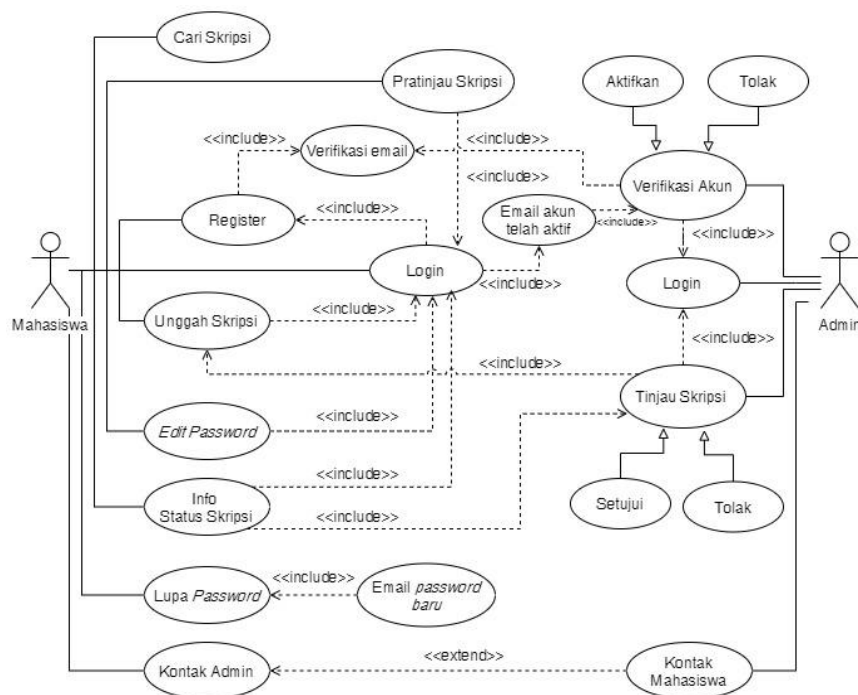
3.3 Fase Iterasi

Pada fase iterasi dilakukan analisis, desain dan *testing*. Analisis pada penelitian ini dibagi menjadi tiga bagian yaitu, Analisis analisis sistem, analisis arsitektur, analisis basis data. Desain meliputi desain antarmuka. Analisis sistem dan arsitektur menggunakan diagram UML sedangkan analisis basis data

menggunakan ERD. Pada tahap ini pula kode diimplementasikan, *testing* fungsionalitas dilakukan, dan ditinjau secara berkala.

3.3.1 Analisis Sistem

1. Use Case diagram



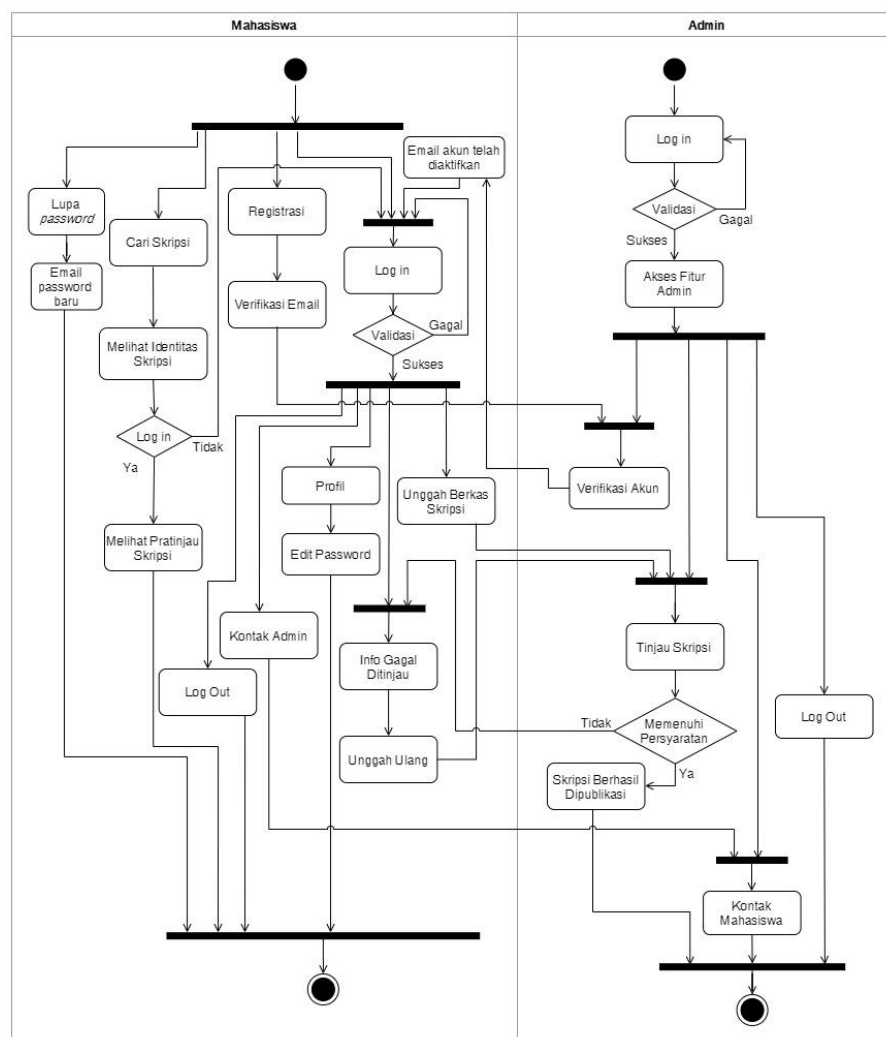
Gambar 3.2 Use Case Diagram

Use case diagram menjelaskan kelakuan dari sistem web repositori skripsi Teknik Informatika Unpad. Pada gambar 3.2, terdapat dua aktor yang merepresentasikan dua jenis pengguna, yaitu mahasiswa dan admin. Mahasiswa dapat melakukan registrasi akun, *login*, pencarian skripsi, melihat skripsi, unggah skripsi, *edit password*, dan melihat info tinjauan skripsi. Pencarian skripsi dapat dilakukan tanpa *login*, sementara fitur-fitur lainnya dapat dilakukan setelah *login*. *Login* dapat dilakukan apabila mahasiswa telah mendaftarkan akun dan akun telah

diaktifkan oleh admin. Admin dapat melakukan verifikasi akun dan tinjau skripsi apabila admin telah *login*.

Terdapat fitur tambahan yang didapat dari hasil pemeliharaan yaitu kontak admin dan lupa *password*. Fitur lupa *password* akan mengirimkan *password* baru ke pengguna melalui *email* yang terdaftar. Fitur kontak admin adalah fitur dimana mahasiswa dapat menghubungi admin melalui web. *Use case diagram* yang menjelaskan sistem web aplikasi hasil final rilis dapat dilihat pada gambar 3.2.

2. Activity diagram



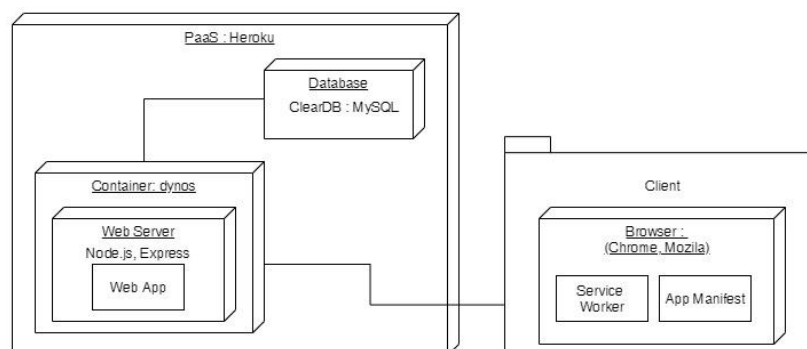
Gambar 3.3 Activity Diagram

Diagram aktivitas menggambarkan aliran kerja dari suatu sistem. Pada gambar 3.3, terdapat dua *swimlane* dengan label mahasiswa dan admin. *Swimlane* memisahkan aktivitas berdasarkan pengguna. Mulai dari status awal, mahasiswa dapat melakukan tiga aktivitas yaitu registrasi, *login*, mencari skripsi. Akun yang didaftarkan akan ditinjau oleh admin terlebih dahulu. Apabila memenuhi syarat maka akun akan diaktifkan. Setelah diaktifkan, mahasiswa dapat *login*.

Mahasiswa yang telah *login* dapat mengunggah skripsi. Skripsi yang diunggah akan ditinjau oleh admin sebelum dipublikasikan. Mahasiswa dapat melihat status dari hasil tinjauan admin, apakah skripsi telah dipublikasi atau ditolak. Mahasiswa juga dapat melihat skripsi mahasiswa lain setelah *login*.

Fitur tambahan yaitu lupa *password* dan kontak admin membuat mahasiswa dapat melakukan pemulihan akun dan mengirim pesan ke admin melalui web. Fitur ini ditambahkan pada fase pemeliharaan. Aliran kerja admin ialah, *log in* kemudian dapat memilih melakukan verifikasi akun atau tinjau skripsi. Diagram aktivitas yang menjelaskan alur kerja web aplikasi hasil final rilis dapat dilihat pada gambar 3.3.

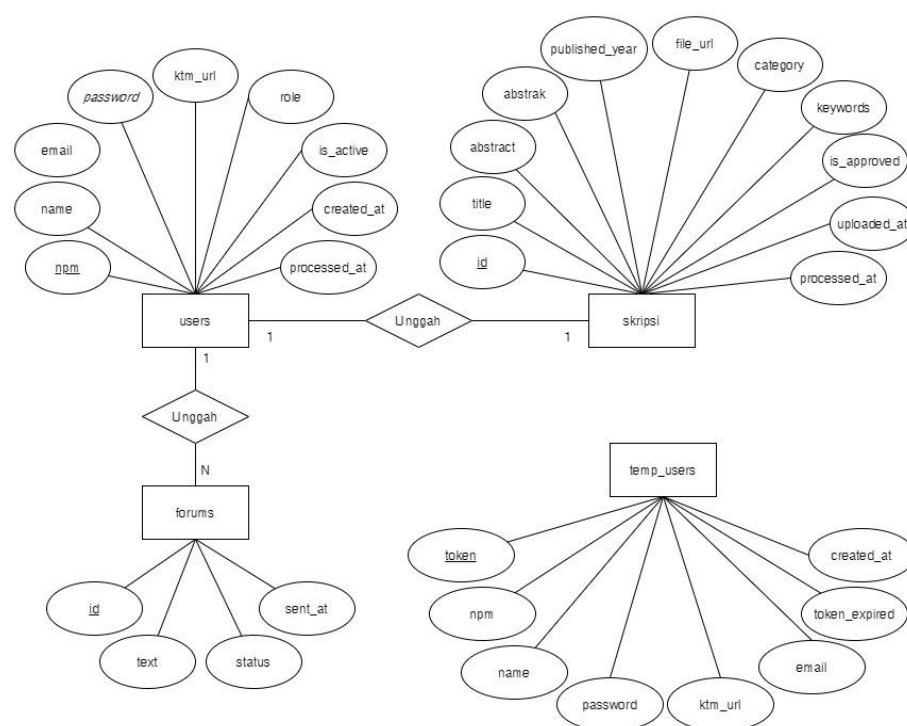
3.3.2 Analisis Arsitektur Menggunakan *Deployment Diagram*



Gambar 3.4 *Deployment Diagram*

Deployment diagram menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Pada gambar 3.4, dijelaskan bahwa pengguna harus memiliki *browser* yang kompatibel (memiliki *service worker*, dan *app manifest*) seperti Chrome atau Mozilla. Web aplikasi di-deploy di heroku menggunakan *database* add-on clearDB. Heroku menggunakan model *container* untuk menjalankan aplikasi. Wadah yang digunakan di Heroku disebut "dyno." Dynos adalah wadah Linux tervirtualisasi yang dirancang untuk mengeksekusi kode berdasarkan perintah yang ditentukan pengguna.

3.3.3 Analisis Basis Data



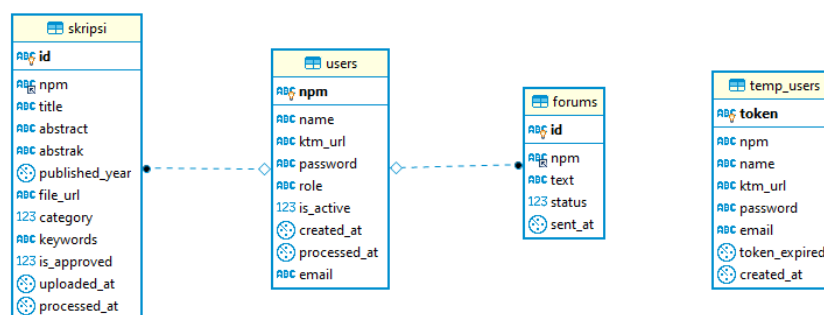
Gambar 3.5 Entity Relationship Diagram

Pada gambar 3.5 terdapat empat entitas dalam ERD yaitu *users*, *temp_user*, *skripsi*, dan *forums*. Tabel *forums* ditambahkan karena hasil *feedback* pada fase pemeliharaan yang menambahkan fitur kontak admin. Tabel *forums* berisi

data pesan yang dikirimkan pada fitur kontak admin. Gambar 3.5 adalah ERD yang menjelaskan *database* web aplikasi hasil final rilis.

Entitas *users* dan skripsi memiliki relasi *one to one*. Sementara entitas *forums* dan skripsi memiliki relasi *one to many*. Entitas *temp_users* berisi penyimpanan sementara data pengguna saat register. Apabila verifikasi *email* telah dilakukan maka data pengguna pada tabel *temp_users* akan dipindahkan ke tabel *users* dan dihapus dari tabel *temp_users*.

Relasi *one to one* diakomodasi pada model data fisik dengan aturan *foreign key* dapat ditambahkan pada tabel mana saja yang terhubung oleh relasi ini. Pada gambar 3.6, NPM ditambahkan pada tabel skripsi sebagai *foreign key*. Relasi *one to many* diakomodasikan pada tabel fisik dengan aturan *foreign key* ditambahkan pada tabel berderajat N. Model data fisik dapat dilihat pada gambar 3.6.



Gambar 3.6 Model Data Fisik

Tabel 3.5 menjelaskan mengenai tipe data, ukuran, dan keterangan dari tiap atribut pada tabel skripsi yang ada pada *database*. Tabel skripsi berisi data dari skripsi yang diunggah oleh mahasiswa. Data tersebut terdiri dari id skripsi, judul, abstrak dalam bahasa inggris dan indonesia, *url file* skripsi, kategori, kata kunci, tahun publikasi, waktu diunggah dan ditinjau oleh admin, status unggahan skripsi, serta NPM dari pengguna terkait.

Tabel 3.5 Atribut Tabel skripsi

No	Atribut	Tipe	Ukuran	Keterangan
1	id	varchar	100	<i>Primary key</i> sebagai kode identitas skripsi
2	npm	varchar	15	<i>Foreign key</i> dari tabel <i>users</i>
3	title	varchar	255	Judul skripsi
4	abstract	text	-	Abstrak skripsi dalam bahasa inggris
5	abstrak	text	-	Abstrak skripsi dalam bahasa indonesia
6	file_url	varchar	255	Url <i>file</i> skripsi
7	category	tinyint	1	Kategori skripsi berdasarkan bidang minat. Nilai 1 adalah Sistem Cerdas dan Sistem Grafika (SCSG), 2 adalah Sistem Informasi dan Rekayasa Perangkat Lunak (SIRPL), 3 adalah Minat Jaringan Komputer dan Komunikasi Data (JKKD), dan 4 adalah Ilmu Komputasi dan Metode Numerik (IKMN)
8	keywords	varchar	255	Kata kunci terkait dengan skripsi untuk memudahkan pencarian
9	published_year	year	-	Tahun skripsi dipublikasikan
10	is_approved	tinyint	1	Terdiri dari 0 (ditolak admin), 1 (disetujui oleh admin), dan 2 belum di proses. <i>Default</i> awal adalah 2
11	uploaded_at	timestamp	-	Waktu skripsi diunggah
12	processed_at	timestamp	-	Waktu skripsi ditinjau oleh admin

Tabel 3.6 menjelaskan mengenai tipe data, ukuran, dan keterangan dari atribut pada tabel *users* yang ada pada *database*. Tabel *users* berisi data dari mahasiswa yang sudah terdaftar. Data tersebut terdiri dari NPM, nama, *email*, URL KTM url, *password*, jenis *user*, status aktivasi akun, waktu daftar, dan waktu diaktifkan oleh admin.

Tabel 3.6 Atribut Tabel *users*

No	Atribut	Tipe	Ukuran	Keterangan
1	npm	varchar	15	<i>Primary key</i> berupa NPM pengguna / <i>username</i> bagi admin. Bersifat unik
2	name	varchar	255	Nama lengkap pengguna
3	email	varchar	100	<i>Email</i> pengguna
4	ktm_url	varchar	255	URL KTM
5	password	varchar	255	<i>Password</i> pengguna
6	role	varchar	6	Jenis pengguna yaitu 'admin' atau 'user'
7	is_active	tinyint	1	Terdiri dari 0 (ditolak admin), 1 (disetujui oleh admin), dan 2 belum di proses. <i>Default</i> awal adalah 2
8	created_at	timestamp	-	Waktu akun didaftarkan
9	processed_at	timestamp	-	Waktu akun diaktifkan oleh admin

Tabel 3.7 menjelaskan mengenai tipe data, ukuran, dan keterangan dari atribut pada tabel *temp_users* yang ada pada *database*. Tabel *temp_users* digunakan sebagai penyimpanan sementara data pengguna saat pertama kali didaftarkan. Setelah *email* diverifikasikan, data dipindahkan ke tabel *users* dan dihapus dari tabel *temp_users*. Atribut pada tabel *temp_users* mirip seperti tabel *users* dengan tambahan atribut token, dan *token_expired*.

Tabel 3.7 Atribut Tabel *temp_users*

No	Atribut	Tipe	Ukuran	Keterangan
1	token	varchar	100	<i>Primary key</i> berupa token yang digunakan untuk verifikasi <i>email</i> .
2	npm	varchar	15	Npm pengguna Bersifat unik
3	name	varchar	255	Nama lengkap pengguna
4	email	varchar	100	<i>Email</i> pengguna
5	ktm_url	varchar	255	URL KTM
6	password	varchar	255	<i>Password</i> pengguna
7	token_expired	timestamp	-	Batas waktu aktifnya token. Data dengan token yang telah <i>expired</i> akan dihapus
8	created_at	timestamp	-	Waktu akun didaftarkan

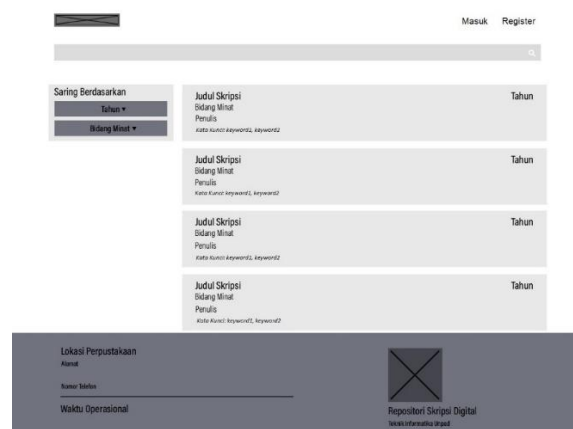
Tabel 3.8 menjelaskan mengenai tipe data, ukuran, dan keterangan dari atribut pada tabel *forums* yang ada pada *database*. Tabel *forums* ditambahkan pada *database* karena adanya penambahan fitur pada fase pemeliharaan. Tabel *forums* berisi pesan dari mahasiswa maupun admin. Tabel *forums* berisi atribut id pesan, id pengirim, text, status pengiriman, dan waktu dikirim.

Tabel 3.8 Atribut Tabel *forums*

No	Atribut	Tipe	Ukuran	Keterangan
1	id	varchar	20	<i>Primary key</i> . 12 digit pertama berisi NPM mahasiswa + 5 digit id untuk pembeda tiap pesan
2	npm	varchar	15	id pengirim pesan. <i>Foreign key</i> dari tabel <i>users</i>
3	text	text	-	pesan yang dikirimkan
4	status	tinyint	1	status berubah bila pesan telah dibalas admin
5	sent_at	timestamp	-	Waktu pesan dikirim

3.3.4 Desain Antarmuka

1. Halaman Utama



Gambar 3.7 Desain Halaman Utama

Rancangan desain antarmuka untuk halaman utama dapat dilihat pada Gambar 3.7. Pada halaman utama, disajikan beberapa skripsi yang ada dalam

database. Mahasiswa juga dapat melakukan pencarian skripsi berdasarkan judul, penulis maupun kata kunci terkait dengan skripsi. Skripsi juga dapat disaring berdasarkan tahun publikasinya atau bidang minat. Bila mahasiswa meng-klik salah satu skripsi yang ada, maka pengguna dibawa ke halaman detail dari skripsi tersebut. Tombol masuk dan register pada pojok kanan atas halaman *web* akan terus ada selama pengguna belum *login*.

2. Registrasi

The image shows two wireframe designs for a registration page, labeled (a) and (b). Both designs include a header with 'Masuk' and 'Register' links. Design (a) features a 'Register' form with fields for 'Name', 'NPM', 'Password', and 'Konfirmasi Password', along with a 'Next' button. Design (b) features a 'Register' form with a 'Foto KTM' field, a 'Choose file' button, and a 'Submit' button. Both designs also include a footer with 'Lokasi Perpustakaan', 'Alamat', 'Kontak', and 'Waktu Operasional' fields, and a 'Repositori Skripsi Digital' logo.

Gambar 3.8 Desain Halaman Registrasi Bagian 1 (a) dan Bagian 2 (b)

Rancangan desain antarmuka untuk halaman registrasi dapat dilihat pada gambar 3.8. Pada halaman registrasi, mahasiswa dapat melakukan registrasi dengan cara mengisi data diri pada *form* yang sudah disajikan. *Form* terbagi menjadi dua bagian, *form* data diri dan unggah foto KTM. Data yang harus dilengkapi yaitu nama, NPM, *email*, *password*, dan konfirmasi *password*. Apabila data sudah dilengkapi dan mahasiswa dapat menekan tombol lanjut untuk maju ke bagian kedua. Apabila data diterima oleh sistem dengan sukses maka mahasiswa dapat lanjut ke bagian kedua. Pada bagian kedua mahasiswa mengunggah foto KTM.

Setelah mengklik tombol *submit*, akan muncul pesan bahwa email verifikasi telah dikirimkan.

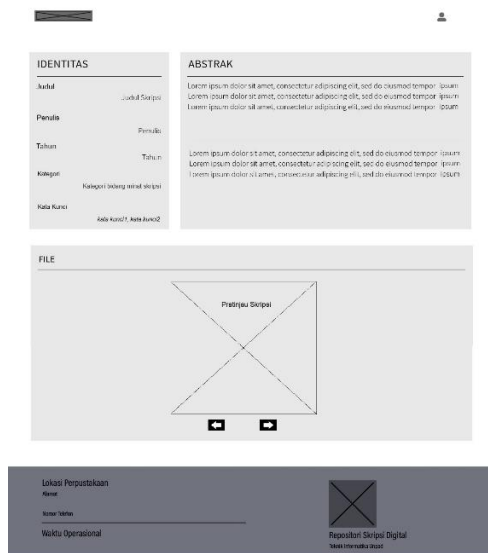
3. Login (Autentikasi)

Rancangan desain antarmuka untuk bagian *login* dapat dilihat pada gambar 3.9. *Form login* dapat diakses di semua halaman apabila pengguna belum *login* dengan cara mengklik tombol masuk pada *navbar*. Pengguna harus mengisi NPM atau *username* bagi admin dan *password* untuk melakukan *login*.



Gambar 3.9 Desain *Login Section*

4. Detail Skripsi



Gambar 3.10 Desain Halaman Detail Skripsi

Rancangan desain antarmuka untuk halaman detail skripsi dapat dilihat pada gambar 3.10. Detail skripsi dibagi menjadi tiga bagian, yaitu identitas, abstrak, dan *file* skripsi. Apabila pengguna telah *login*, maka pengguna dapat melihat *file* skripsi

dibawah identitas skripsi. Sebaliknya, saat belum *login* pengguna hanya dapat melihat identitas dan abstrak skripsi.

5. Unggah Skripsi

The image shows a web form titled 'Unggah' (Upload). It contains several input fields: 'Judul' (Title), 'Tahun' (Year), 'Abstrak' (Abstract), 'Abstract', 'File * (Maks 20Mb)' with a 'Choose file' button, 'Bidang Minat Skripsi' (Thesis Interest), and 'Kata Kunci' (Keywords). A 'Submit' button is located at the bottom of the form. Below the form, there is a footer section with 'Lokasi Perpustakaan', 'Alamat', 'Nomor Telepon', 'Waktu Operasional', and a logo for 'Repositori Skripsi Digital'.

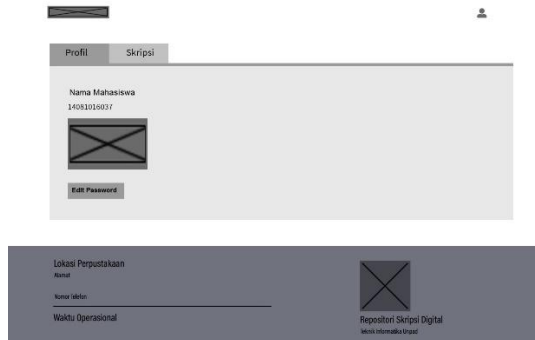
Gambar 3.11 Desain Halaman Unggah Skripsi

Rancangan desain antarmuka untuk halaman unggah skripsi dapat dilihat pada gambar 3.11. Mahasiswa yang telah menyelesaikan skripsi dapat mengunggah skripsinya dengan mengisi *form* data skripsi. Data yang harus diisi ialah judul skripsi, tahun publikasi, abstrak dalam bahasa Indonesia dan Inggris, *file* skripsi, bidang minat yang terdiri dari Sistem Cerdas dan Sistem Grafika (SCSG), Sistem Informasi dan Rekayasa Perangkat Lunak (SIRPL), Minat Jaringan Komputer dan Komunikasi Data (JKKD), dan Ilmu Komputasi dan Metode Numerik (IKMN), serta kata kunci terkait dengan skripsi yang diunggah.

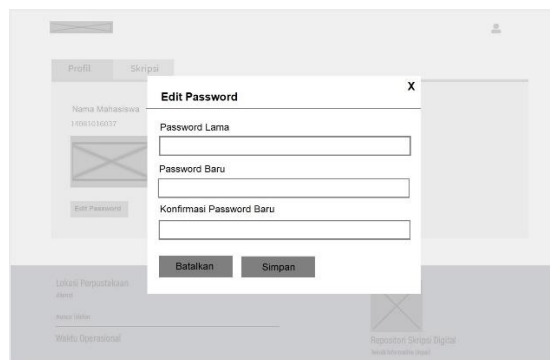
6. Profil

Rancangan desain antarmuka untuk halaman profil dapat dilihat pada gambar 3.12. Pada halaman ini mahasiswa dapat melihat nama, NPM, foto KTM, dan menu mengubah *password*. Saat mahasiswa mengklik tombol *edit password* maka akan

muncul modal seperti yang ditunjukkan pada gambar 3.13. Mahasiswa harus mengisi *password* lama, *password* baru dan mengisi konfirmasi *password*.



Gambar 3.12 Desain Halaman Profil Mahasiswa



Gambar 3.13 Desain Halaman *Edit Password*

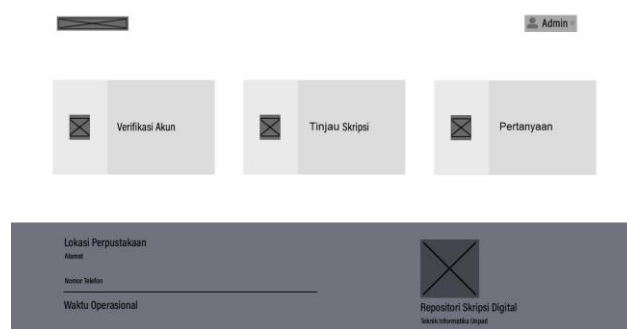
7. Status Skripsi



Gambar 3.14 Desain Halaman Status Skripsi

Rancangan desain antarmuka untuk halaman status skripsi dapat dilihat pada gambar 3.14. Data yang diisi pada saat unggah skripsi akan ditampilkan beserta waktu unggah, status skripsi, dan waktu publikasi oleh admin. Tombol unggah ulang ada pada halaman selama skripsi belum dipublikasikan oleh admin.

8. Menu Admin



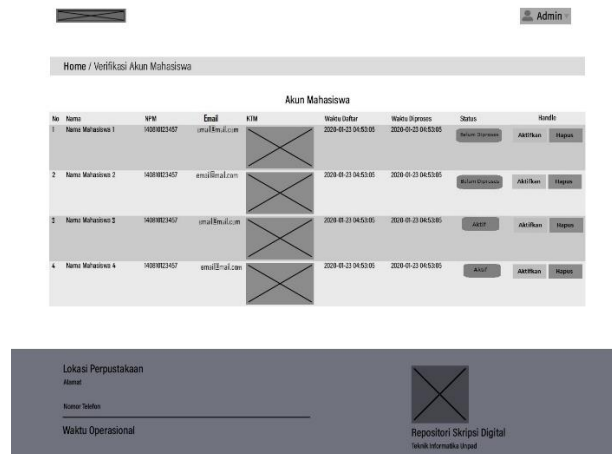
Gambar 3.15 Desain Halaman Menu Admin

Apabila pengguna yang *login* memiliki *role* admin, maka pengguna akan dibawa ke halaman menu admin. Rancangan desain antarmuka untuk halaman menu admin dapat dilihat pada gambar 3.15. Menu admin terdiri dari dua pilihan yaitu verifikasi akun dan tinjau skripsi. Pada fase pemeliharaan, menu admin ditambah menjadi 3 pilihan yaitu menu kontak mahasiswa untuk menjawab pertanyaan yang dikirimkan mahasiswa.

9. Verifikasi akun

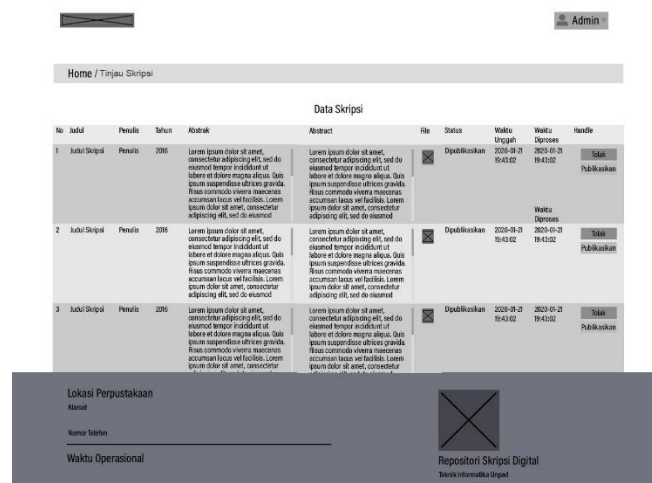
Rancangan desain antarmuka untuk halaman verifikasi akun dapat dilihat pada gambar 3.16. Halaman verifikasi akun hanya dapat diakses oleh admin. Pada tabel terdapat nama, NPM, *email*, KTM mahasiswa, waktu daftar, waktu proses, status, dan tombol aksi. Admin dapat melihat data registrasi mahasiswa. Apabila data valid admin mengaktifkan akun mahasiswa dan mengubah status akun dengan

mengklik tombol aktifkan. Dan sebaliknya bila data tidak valid maka admin dapat mengklik tombol hapus untuk menghapus data akun tersebut.



Gambar 3.16 Desain Halaman Verifikasi Akun Mahasiswa

9. Tinjau Skripsi



Gambar 3.17 Desain Halaman Tinjau Skripsi

Rancangan desain antarmuka untuk halaman tinjau skripsi dapat dilihat pada gambar 3.17. Halaman tinjau skripsi hanya dapat diakses oleh admin. Pada tabel terdapat data skripsi yang diunggah mahasiswa, yaitu judul, nama penulis, tahun, abstrak, *abstract*, *file* skripsi, status, waktu unggah, waktu proses, dan tombol aksi.

Admin dapat akan melakukan pengecekan pada data skripsi dan memutuskan apakah data skripsi valid dan siap untuk dipublikasikan.

10. Kontak Admin

Gambar 3.18 Desain Halaman Kontak Admin

Menu kontak admin adalah salah satu fitur yang ditambahkan pada fase pemeliharaan. Halaman kontak admin berisi kolom pesan yang dikirimkan oleh mahasiswa ke admin. Desain halaman kontak admin dapat dilihat pada gambar 3.18.

11. Lupa Password

Gambar 3.19 Desain Halaman Lupa Password

Lupa password adalah salah satu fitur yang ditambahkan pada fase pemeliharaan. Lupa password dapat diakses dibawah form login. Pengguna dibawa

kehalaman lupa *password* untuk mengisi *email* dan NPM yang terdaftar. Desain antarmuka halaman lupa *password* dapat dilihat pada gambar 3.19.

3.3.5 Testing

Pada tahapan ini dilakukan *testing*. *Testing* yang dimaksud adalah pengujian terhadap kode dan fungsi-fungsi pada web repositori skripsi Teknik Informatika Unpad. Pengujian terhadap pengguna tidak dilakukan pada tahap ini. Proses pengujian fungsional dilakukan setelah tiap fitur selesai dikerjakan.

Metode pengujian yang digunakan pada tahapan ini adalah metode *black box testing*, dimana pengujian dilakukan dengan cara memberi beberapa masukan dan memerhatikan output yang dihasilkan. Hal ini dilakukan untuk menguji berbagai skenario yang dapat dilakukan pengguna terhadap web repositori skripsi Teknik Informatika Unpad dengan harapan web dapat berjalan dengan baik pada setiap skenario. Web dapat memasuki fase produksi setelah tahapan *testing* selesai. Pengujian fungsionalitas fitur akan dijelaskan lebih lanjut pada BAB IV bagian implementasi fitur.

3.4 Fase Produksi

Pada fase ini rilisan kecil dihasilkan dan diujicobakan pada pengguna. Web repositori skripsi Teknik Informatika Unpad siap untuk dirilis apabila fitur-fitur pada yang dijelaskan pada fase eksplorasi telah berhasil diimplementasikan dan berjalan dengan baik. Fitur-fitur tersebut adalah register, *login*, unggah skripsi, verifikasi akun, tinjau skripsi, melihat pratinjau skripsi, pencarian dan penyaringan, *edit password*, cek status skripsi, dan fitur PWA. Fitur PWA yang dimaksud ialah, web memiliki tampilan pada saat *offline* dan tombol *add to homescreen*.

Dengan kata lain, rilis kecil menghasilkan produk yang memenuhi *user story*. Pada fase ini pula dilakukan pengujian terhadap *user* untuk mendapatkan *feedback* terhadap aplikasi. Pengujian dilakukan dengan menggunakan *usability testing*. *Testing* yang dilakukan berfokus pada pencarian *feedback*.

Usability testing ini menggunakan skema penilaian 5 titik respon dari skala Likert untuk mengukur hasil respon dari setiap pernyataan yang disajikan dalam kuesioner yang ditujukan kepada para calon pengguna. Lima tingkatan jawaban atau respon dari kuesioner yang akan digunakan untuk *Usability Testing* terdapat pada Tabel 2.5. Maka, hasil penilaian dari kuesioner tersebut dapat dihitung dengan menentukan interval dan kriteria nilai terlebih dahulu dengan menggunakan Persamaan 2.1:

$$Interval = \frac{100}{5} = 20$$

Persamaan 3.1 Interval Pengujian

Sehingga didapatkan kriteria nilai untuk pengujian ini dapat dilihat pada Tabel 3.9:

Tabel 3.9 Atribut Kriteria Nilai

Hasil	Keterangan
0 – 19.99	Sangat buruk
20 – 39.99	Buruk
40 – 59.99	Cukup
60 – 79.99	Baik
80 – 100.0	Sangat Baik

Berikutnya, untuk menentukan total skor dan nilai hasil dapat menggunakan Persamaan 2.2 dan Persamaan 2.3. Pengujian dilakukan dengan skenario *usability testing* pada tabel 3.10.

Tabel 3.10 Skenario *Testing*

No	Task	Fitur yang diuji
1	Pengguna diminta untuk mengakses link <i>repositori-skripsi.herokuapp.com</i> melalui <i>device</i> yang diinginkan melalui Chrome atau Mozilla	Halaman utama / <i>Landing page</i>
2	Pengguna diminta untuk mencari skripsi dengan bidang minat dan tahun tertentu	Pencarian dan filter
3	Pengguna diminta untuk mendaftarkan diri	Registrasi
4	Pengguna diminta untuk <i>login</i> dengan akun yang telah didaftarkan	<i>Login</i> (Autentikasi)
5	Pengguna diminta untuk melihat <i>file</i> skripsi tertentu	Melihat skripsi
6	Pengguna diminta untuk mengunggah skripsi	Unggah berkas
7	Pengguna diminta untuk mengedit unggahan	
8	Pengguna diminta untuk mengecek status unggahannya	Status skripsi
9	Pengguna diminta untuk mengubah <i>password</i>	<i>Edit password</i>
10	Pengguna diminta untuk menambahkan web pada <i>homescreen</i>	PWA (tampilan saat <i>offline</i> dan <i>add to screen shortcut</i>)
11	Pengguna diminta untuk <i>offline</i> dan merefresh halaman	

Setelah responden selesai melakukan skenario pada tabel 3.10, responden diminta untuk mengisi form penilain terhadap fitur-fitur dari segi fungsi dan UI menggunakan sekala likert. Tabel 3.11 berisi pertanyaan pada form.

Tabel 3.11 *Form Usability Testing*

No	Pertanyaan	Fitur / Fungsi
1	Bagaimana penilaian tampilan halaman utama	<i>Landing Page</i>
2	Bagaimana penilaian fungsi pencarian dan filter	Pencarian
3	Bagaimana penilaian fungsi registrasi	Register
4	Bagaimana penilaian tampilan registrasi	
5	Bagaimana penilaian fungsi <i>login</i>	<i>Login</i>
6	Bagaimana penilaian tampilan <i>login</i>	
7	Bagaimana penilaian tampilan halaman detail skripsi	Pratinjau Skripsi
8	Bagaimana penilaian fungsi unggah skripsi	Unggah skripsi
9	Bagaimana penilaian fungsi <i>edit</i> unggahan	
10	Bagaimana penilaian tampilan unggah skripsi	
11	Bagaimana penilaian tampilan profil mahasiswa	<i>Edit password</i>
12	Bagaimana penilaian fungsi <i>edit password</i>	

No	Pertanyaan	Fitur / Fungsi
13	Bagaimana penilaian tampilan edit <i>password</i>	
14	Bagaimana penilaian fungsi pada halaman status skripsi	Cek status skripsi
15	Bagaimana penilaian tampilan halaman status skripsi	
16	Bagaimana penilaian terhadap fungsi <i>offline</i> ?	PWA
17	Bagaimana penilaian terhadap tampilan saat <i>offline</i> ?	
18	Bagaimana penilaian terhadap fungsi <i>add to homescreen</i> ?	
19	Apakah fitur <i>add to homescreen</i> membuat tampilan nyaman seperti <i>native app</i> ? Berikan skala penilaian	
20	Adakah <i>feedback</i> terhadap fitur-fitur web maupun web secara keseluruhan? (berupa text yang tidak dinilai dengan skala likert)	-

3.5 Fase Pemeliharaan dan Fase Akhir

Pada fase ini, pemeliharaan web aplikasi dilakukan berdasarkan *feedback* dari hasil pengujian terhadap *user* pada fase produksi. Pemeliharaan meliputi perbaikan *bugs* yang ditemukan saat pengujian dan implementasi saran yang masih mendukung tujuan utama dari web repositori skripsi Teknik Informatika Unpad. Secara umum pemeliharaan akan mengulangi fase iterasi. Saran dan *bugs* akan diimplementasikan seperti fitur-fitur diimplementasikan pada fase iterasi. Apabila web terus mendapat masukan dari pengguna maka akan terjadi pengulangan yang menghasilkan rilis yang diperbaharui. Penelitian hanya melakukan 1 iterasi maka rilis yang diperbaharui akan menjadi produk final.

Fase akhir dicapai apabila pengguna tidak memiliki *story* tambahan lagi. Pada fase ini, web repositori skripsi Teknik Informatika Unpad harus sudah memenuhi tujuan dan batasan-batasan yang dijelaskan pada BAB I. Produk final akan dirilis sebagai hasil dari fase final.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Impelementasi Program

Dalam metode XP, Implementasi program dilakukan pada fase iterasi. Implementasi program mengikuti urutan prioritas fitur sebagaimana yang dijelaskan pada tahap perencanaan. Pada tahapan ini, dilakukan implementasi *user interface (frontend)* yaitu melakukan pengkodean tampilan sesuai dengan rancangan desain antarmuka. Implementasi tampilan dilakukan dengan menggunakan ReactJS.

Setelah implementasi tampilan selesai, dilanjutkan dengan implementasi *backend* menggunakan Express. Sistem yang telah diimplementasikan akan diuji fungsionalitasnya. Web repositori skripsi Teknik Informatika Unpad dapat dirilis bila fungsi-fungsi berjalan dengan baik memenuhi *acceptance criteria* pada *user story*. Pada bagian ini dijelaskan tentang pengkodean dan pengujian fitur. Berikut penjelasan mengenai implementasi untuk setiap fitur pada web skripsi Teknik Informatika Unpad.

4.1.1 Halaman Utama

Halaman ini adalah halaman pertama dilihat saat mengakses web repositori skripsi Teknik Informatika Unpad. Halaman ini berisi informasi singkat dari skripsi-skripsi yang ada pada *database*. Halaman utama memiliki *pagination* yang menampilkan 10 skripsi perhalaman. Hasil implementasi halaman utama dapat dilihat pada Gambar 4.1.



Gambar 4.1 Halaman Utama pada *Desktop* (a) dan *Mobile* (b)

Proses yang terjadi pada halaman ini ialah *request* data ke *backend* dan menampilkannya. Pada *backend*, dilakukan proses membaca data dengan *method get*. Berikut adalah potongan kode pada *backend* yang digunakan untuk mendapatkan data dari *database*:

```
router.get('/list', (req, res) =>{
  let sql = `SELECT skripsi.id, skripsi.title,
skripsi.published_year, skripsi.category, skripsi.keywords,
users.name FROM skripsi join users on users.npm = skripsi.npm
where is_approved=${1} ORDER BY published_year desc,
skripsi.processed_at desc`
  db.query(sql, (err, result)=>{
    if (err) console.log(err)
    res.send(result) })
})
```

Request dipanggil dalam *componentDidMount method*, yang berarti *request* dilakukan setelah inisial *rendering* dilakukan. Berikut adalah kode *request* data skripsi pada *frontend*:


```

getSkripsi=()=>{ //fungsi request data skripsi menggunakan axios
  axios({
    method: 'get',
    url: '/skripsi/list',
  }).then(res=>{
    this.setState({
      skripsi: res.data,
      isLoading: true,
      skripsiFiltered:res.data,
      skripsiFilteredTemp:res.data,
      skripsiFilteredCat:res.data,
      skripsiFilteredYear:res.data,
      years: [...new Set(res.data.map((year)=>{
        return year.published_year
      }))]}.sort()
    })
    localStorage.setItem('list', JSON.stringify(res.data))
  }).catch((err) => {
    if(err.response) console.log(err.response)
  })
}
componentDidMount(){
  if (navigator.onLine){ //Saat online, lakukan request data
    this.getSkripsi()
    this.setState({
      offline:false
    })
  }
  else{ //Saat offline ambil data dari local storage
    if (localStorage.getItem('list')){
      let data = JSON.parse(localStorage.getItem('list'))
      this.setState({
        skripsi: data,
        isLoading: true,
        skripsiFiltered:data,
        skripsiFilteredTemp:data,
        skripsiFilteredCat:data,
        skripsiFilteredYear:data,
        years: [...new Set(data.map((year)=>{
          return year.published_year
        }))]}.sort()
      })
    }
  }
}
}

```

Data yang didapat dikirimkan sebagai *props* ke *component list* untuk di-loop dan ditampilkan ke dalam bentuk *card* pada halaman utama. Pada kode dibawah dapat dilihat bahwa data yang dikirimkan merupakan data hasil *filter* sehingga tiap

pencarian maupun penyaringan akan mengubah data yang tampil. Fungsi filter akan dibahas lebih lanjut pada bagian fitur pencarian dan penyaringan.

```
const currentPosts = skripsiFiltered.slice(indexOfFirstPost,
indexOfLastPost)
<ListCard skripsi={currentPosts} isLoaded={isLoaded}> </ListCard>
```

4.1.2 Halaman Register

Gambar 4.2 Halaman Registrasi bagian 1

Gambar 4.3 Halaman Registrasi Bagian 2

Pengguna yang belum memiliki akun harus mendaftarkan diri melalui halaman ini. Halaman register dapat diakses melalui tombol register di kanan atas *navbar* selama pengguna belum *login*. Halaman registrasi berisi *form* yang dipisahkan kedalam dua bagian. *Form* berisi data diri dengan tipe *string* diterima

terlebih dahulu untuk dilakukan pengecekan data. Implementasi halaman register dapat dilihat pada gambar 4.2 dan gambar 4.3

Tombol lanjut tidak aktif bila data tidak diisi, panjang NPM tidak sampai 12 digit, dan apabila data konfirmasi *password* berbeda dengan data pada kolom *password*. Menekan tombol lanjut akan memanggil fungsi *next()*. Fungsi *next()* akan mengirimkan isi *form* ke *backend* dengan *method post*. Berikut adalah potongan fungsi *next()*:

```
next = (e) =>{ //Fungsi mengirim data dengan Axios
  e.preventDefault()
  this.setState({
    showLoading:true,
  })
  let {npm, pass}= this.state
  let name = this.refs.name.value
  let data={
    name:name,
    npm:npm,
    password:pass
  }
  axios({ //mengirim data dengan method post axios
    method: 'POST',
    url: '/check-form',
    data: data
  }).then(res => {
    this.setState({
      message: '',
      displayForm1: 'none',
      displayForm2: 'block',
      progress: this.state.progress + 33,
      showLoading:false
    })
    scrollTopToTop()
  }).catch(err => {
    this.setState({
      showLoading:false
    })
    if (err.response) {
      this.setState({
        message: err.response.data.message,
        status: err.response.data.status,
      })
    }
    else{
      this.setState({
        message: 'Network error, Cek Koneksi anda',
        status: 500,
      })
    }
  })
}
```

```

    })
  }
})
}

```

Pada *backend*, pengecekan data dilakukan. Pengecekan yang dilakukan adalah pengecekan kolom data yang tidak boleh kosong dan pengecekan panjang NPM yang tidak boleh kurang dari 12 digit. Pengecekan NPM juga dilakukan terhadap *database* untuk mengetahui apakah NPM tersebut sudah terdaftar sebelumnya. Berikut adalah potongan kode pada *backend* yang digunakan untuk mengecek data pada *database*:

```

router.post('/check-form', (req, res) =>{
  let { name, npm, password } = req.body
  //Cek apakah semua kolom terisi
  if (!name || !npm || !password ) {
    return utils.template_response(res, 400, "Semua field harus diisi" , null)
  }
  //Cek panjang NPM
  if(npm.length<12 || npm.length>=15 ) {
    return utils.template_response(res, 422, "NPM salah. Memerlukan angka 12 digit" , null)
  }
  //cek email
  if(!email.includes('@') || !email.includes('.') ) {
    return utils.template_response(res, 422, "Email tidak valid" , null)
  }
  //Cek apakah NPM atau email sudah terdaftar sebelumnya
  let findUser = `SELECT npm FROM users where role='user' AND npm='${npm}' or email='${email}'`
  db.query(findUser, (err, data)=>{
    if (err) console.log(err.response)
    if(data.length>0){
      return utils.template_response(res, 422, "NPM sudah pernah didaftarkan" , null)
    }
    console.log('Data valid')
    return utils.template_response(res, 200, "Data valid", null)
  })
})
})

```

Apabila pengecekan sukses dilakukan, pengguna dapat melanjutkan ke bagian kedua. Pengguna mengunggah foto KTM dalam format png, jpg, atau jpeg

dengan ukuran maksimum 5 mb. *File* yang tidak memenuhi kriteria akan ditolak dan memunculkan pesan *error*. Apabila *file* memenuhi kriteria maka seluruh data diri dan *file url* akan disimpan ke tabel *temp_users*. Token di *generate* dan disimpan di tabel *temp_user*.

Pengguna akan dikirimkan *email* verifikasi yang berisi *link* API yang akan menverifikasikan token pada *link* terhadap token yang tersimpan. Apabila sesuai maka data akan disimpan ke tabel *users*. Dengan demikian pengguna telah terdaftar. Tahap selanjutnya bagi pengguna ialah menunggu aktivasi akun oleh admin. Kriteria-kriteria pada *form* akan uji dengan *black box testing* seperti yang tertera pada tabel 4.1. Berikut adalah potongan kode pada *backend* yang digunakan untuk menyimpan data ke tabel *temp_users*:

```
const upload = multer({ //Fungsi menyimpan ktm ke local folder
  storage: storage,
  limits:{fileSize: 1024 * 1024 * 5},
  fileFilter:fileFilter
}).single('ktm')
router.post('/register', (req, res) =>{
  upload(req, res, asyncHandler(async(err) => { //callback upload
    let {name, npm, password } = req.body
    let genId = npm + uuid()
    console.log(req.file)
    console.log(req.body)
    if(err){
      return utils.template_response(res, 500, err.message , null)
    }
    //Cek kolom file tidak boleh kosong
    if (!req.file){
      return utils.template_response(res, 500, 'File tidak boleh kosong' , null)
    }
    let encryptPassword = await bcrypt.hash(password, 10)
    let expired=moment().add(30, 'minutes').format()
    let data = {
      name: name,
      email:email,
      npm: npm,
      password: encryptPassword,
      ktm_url:req.file.path,
      created_at:moment().format(),
      token: uuid(),
```

```

    token_expired:expired
  }
  console.log(data) //Query menambah data ke tabel temp_users
  let sql = 'INSERT INTO temp_users SET ?'
  db.query(sql, data, (err, result)=>{
    if (err){
      console.log('Failed',err)
      return utils.template_response(res, 400, "Gagal register",
null)
    }
    console.log('Success')
    //mengirim pesan menggunakan sendgrid
    var helper = require('sendgrid').mail;
    var from_email = new helper.Email('no-reply@repositori-
skripsi.com');
    var to_email = new helper.Email(email);
    var subject = 'Verifikasi Email!';
    var emailText=`<html>
<body>
  <p>Halo ${name},</p>
  <p>Anda telah melakukan registrasi akun pada web aplikasi
repositori-skripsi</p>
  <p>Mohon verifikasi email anda dengan menklik link berikut.
link akan aktif selama 30 sejak email dikirim</p>
  <a href=${'https://repositori-skripsi.herokuapp.com/email-
verification/'+data.token}>Verifikasi Email!</a>
</body>
</html>`
    var content = new helper.Content('text/html', emailText);
    var mail = new helper.Mail(from_email, subject, to_email,
content);
    var sg = require('sendgrid')(process.env.SENDGRID_API_KEY);
    var request = sg.emptyRequest({
      method: 'POST',
      path: '/v3/mail/send',
      body: mail.toJSON(),
    });
    sg.API(request, function(error, response) {
      console.log(response.statusCode);
      console.log(response.body);
      console.log(response.headers);
    });
    return utils.template_response(res, 200, "Email verifikasi
telah dikirim", null)
  })
}))
})

```

Tabel 4.1 *Black Box Testing* pada Fitur Registrasi

Skenario	Hasil yang diharapkan	Hasil
Tidak memberi masukan apapun pada <i>form</i> data diri	Pengguna tidak dapat melanjutkan ke bagian dua karena tombol lanjut tidak aktif	Berhasil

Skenario	Hasil yang diharapkan	Hasil
Mengosongkan salah satu <i>field</i>	Pengguna tidak dapat melanjutkan ke bagian dua karena tombol lanjut tidak aktif	Berhasil
Menginputkan NPM atau <i>email</i> yang telah terdaftar	Muncul pesan bahwa NPM atau <i>email</i> telah terdaftar setelah pengguna mengklik tombol lanjut	Berhasil
Menginputkan NPM yang salah (bukan angka, !=12 digit)	Muncul pesan bahwa NPM harus 12 digit setelah pengguna selesai menulis NPM dan tombol lanjut tidak aktif	Berhasil
Tidak mengisi kolom konfirmasi <i>password</i> dengan benar	Muncul pesan bahwa <i>password</i> tidak cocok. Pengguna tidak dapat melanjutkan ke bagian dua karena tombol lanjut tidak aktif	Berhasil
Mengisi semua <i>field</i> data diri dengan benar	Pengguna dapat lanjut ke bagian dua	Berhasil
Tidak mengunggah foto KTM	Muncul pesan bahwa <i>file</i> tidak boleh kosong	Berhasil
Mengunggah <i>file</i> yang bukan jpeg atau png	Muncul pesan bahwa <i>file</i> harus png, jpg, atau jpeg	Berhasil
Mengunggah <i>file</i> yang lebih besar dari 5mb	Muncul pesan bahwa <i>file</i> terlalu besar	Berhasil
Mengunggah <i>file</i> foto KTM dengan format jpg/png dengan besar lebih kecil dari 5MB	Muncul pesan bahwa pengguna harus melakukan verifikasi melalui <i>link</i> yang dikirimkan ke <i>email</i> . <i>Email</i> dikirimkan ke pengguna.	Berhasil
Mengakses <i>link</i> yang dikirimkan ke <i>email</i>	Muncul pesan bahwa register berhasil dan pengguna diharap menunggu aktivasi dari admin. Data pengguna tersimpan di tabel <i>users</i> dan dihapus dari <i>temp_users</i>	Berhasil

4.1.3 Menu *Login*

Tombol *login* berada kanan atas halaman pada tampilan desktop dan berada dalam menu *hamburger* pada tampilan *mobile*. Pengguna memasukkan NPM dan

password yang telah didaftarkan untuk *login*. Implementasi *login* dapat dilihat pada gambar 4.4

Gambar 4.4 Menu *Login*

Data yang dikirimkan ke *backend* akan dicek terlebih dahulu. Apabila NPM dan *password* yang dimasukkan benar, *backend* akan men-generate JWT (JSON web token) yang kemudian dikirimkan sebagai salah satu *responses* ke *frontend*. JWT digunakan sebagai info bahwa pengguna telah *login*. Berikut adalah potongan kode *login* pada *backend* dan *frontend*:

```
router.post('/login', (req, res) =>{
  let {npm, password} = req.body
  if ( !npm || !password){ //Cek apakah npm dan pass terisi
    if (!npm && !password){
      return utils.template_response(res, 400, "NPM & password harus diisi" , null)
    }
    if(!npm){
      return utils.template_response(res, 400, "NPM harus diisi" , null)
    }
    else{
      return  utils.template_response(res, 400, "Password harus diisi" , null)
    }
  }
  //query untuk mencari user dalam database
  const findUser = `SELECT * FROM users WHERE npm='${npm}'`
  db.query(findUser, npm, async (err, result)=>{
    try{
      //Cek apakah user sudah terdaftar
      if (result.length < 1){
        return  utils.template_response(res, 400, "Akun belum terdaftar" , null)
      }
      //Cek apakah akun telah aktif
      let user = result[0]
      if (user.is_active != 1){
        return  utils.template_response(res, 400, "Akun belum diaktifkan. Harap tunggu admin meninjau akun", null)
      }
    }
  })
})
```



```

//Komparasi password
if( !await bcrypt.compare(password, user.password)){
  return utils.template_response(res, 400, "Password tidak
cocok", {token: '', isLogged:false})
}
//Generate JWT
var payload = {
  "iss": "repository.apps",
  "aud": "world",
  "typ": "JWT",
  "request": {
    "npm": user.npm,
    "role": user.role,
    "name": user.name,
  }
}
var secret = "repository.secret"
jwt.sign(payload, secret, { expiresIn: '1d' }, function (err,
token) {
  let bearer = 'Bearer ' + token
  if (err) {
    return utils.template_response(res, 500, "internal api
error", null)
  }
  return utils.template_response(res, 200, "Login Berhasil",
{token: bearer, isLogged:true, role:user.role})
})
}
catch(err){
  return (err)
}
})
})

```

Response yang didapat saat *login* berhasil dilakukan ialah token, *isLogged*, dan *role*. *Response* tersebut kemudian disimpan pada *state global* agar dapat diakses oleh semua *component*. *State global* disimpan menggunakan Redux. JWT yang disimpan pada *state global* nantinya digunakan oleh komponen-komponen lain saat melakukan *request* yang mengharuskan pengguna untuk *login*. Berikut adalah potongan kode pada *frontend*:

```

submitLogin = e => {
  e.preventDefault()
  this.setState({ //menampilkan modal loading
    showLoading:true
  })
  axios(//mengirim data dengan method post menggunakan axios
    method: 'post',

```

```

    url: '/login',
    data: {
      npm: this.state.npm,
      password: this.state.pass
    }
  }).then(res => {
    let loginInfo = res.data.data
    if (loginInfo.isLogged){ //dilakukan saat login sukses
      this.props.login(loginInfo) //set state global
      this.setState({
        showLoading:false, //menghilangkan modal loading
        status:res.data.status,
        justLoggedIn:true,
        showLogin:true, //menampilkan modal berhasil login
      })
      scrollToTop()
      setTimeout(() =>
        this.setState({ //menghilangkan modal login
          showLogin:false
        }), 1000)
    }
    else{
      this.setState({
        showLoading:false,
        status:500
      })
    }
  }).catch((err) => { //dilakukan saat login gagal
    if(err.response){
      this.setState({
        message:err.response.data.message,
        status:err.response.data.status,
        showLoading:false,
      })
    } else{
      this.setState({
        status: 500,
        showLoading:false,
      })
    }
  })
}
...
const mapDispatchToProps = dispatch => { //Redux
  return {
    login: (loginInfo) => dispatch(setToken(loginInfo)),
    logout: () => dispatch(delToken())
  }
}
}

```

Fitur *login* diuji menggunakan *black box testing* dengan skenario seperti yang tertera pada tabel 4.2.

Tabel 4.2 *Black Box Testing* pada Fitur *Login*

Skenario	Hasil yang diharapkan	Hasil
NPM atau <i>password</i> tidak diisi	Muncul pesan untuk mengisi NPM dan <i>password</i>	Berhasil
Memberi masukan NPM yang belum terdaftar	Muncul pesan bahwa akun belum terdaftar	Berhasil
Memberi masukan <i>password</i> yang salah	Muncul pesan bahwa <i>password</i> salah	Berhasil
Memberi masukan NPM yang belum diverifikasi oleh admin	Muncul pesan untuk menunggu akun diaktivasi oleh admin	Berhasil
Mahasiswa <i>login</i> dengan NPM dan <i>password</i> yang benar	Muncul <i>pop up</i> bahwa pengguna telah <i>login</i>	Berhasil
Admin <i>login</i> dengan <i>username</i> dan <i>password</i> yang benar	web akan <i>redirect</i> ke halaman menu admin	Berhasil

4.1.4 Halaman Unggah Skripsi

Gambar 4.5 Halaman Unggah Skripsi

Unggah berkas adalah salah satu fitur utama dari web repositori skripsi Teknik Informatika Unpad. Mahasiswa dapat mengunggah berkas setelah *login*.

Halaman unggah skripsi berisi *form* yang terdiri dari data wajib diisi dan data opsional. Data yang wajib diisi ialah judul, tahun publikasi, abstrak, abstract dan *file* skripsi. Data opsional ialah bidang minat, dan kata kunci. *File* skripsi yang diunggah harus berformat pdf dan memiliki ukuran kurang dari 20mb. Implementasi halaman unggah skripsi dapat dilihat pada gambar 4.5.

Setelah halaman melakukan inisial *rendering*, *request* data skripsi dilakukan untuk mengecek apakah pengguna sudah pernah mengunggah *file* sebelumnya. Apabila sudah, *form* tidak akan ditampilkan melainkan pesan berisi pemberitahuan untuk menuju menu profil untuk melihat status skripsi yang sudah pernah diunggah.

Proses yang terjadi pada saat pengguna mengunggah skripsi ialah *frontend* menerima data skripsi dari *form* kemudian mengirimnya ke *backend*. JWT ikut dikirimkan melalui *headers request* karena pada *backend* terdapat *middleware* untuk mengecek autentikasi pengguna. Hal ini juga berlaku pada *request-request* lain yang mengharuskan pengguna untuk *login*. Berikut adalah potongan code pada *frontend*:

```
checkSkripsi={()=>{ //fungsi cek apakah user sudah mengunggah file
  axios({
    method: 'get',
    url: `/user/skripsi/`,
    headers: {
      Authorization: this.props.token
    }
  }).then(res=>{
    this.setState({
      skripsi: res.data,
      isLoading: true
    })
  }).catch((err) => {
    if(err.response){
      console.log(err.response)
    }
  })
}
submit = (e) =>{ //fungsi submit form unggah skripsi
  e.preventDefault()
}
```

```

this.setState({ //muncul modal berisi pesan loading
  showLoading:true
})
let {title, year, abstract, category, keywords} = this.state
let {file} = this.state
const formData = new FormData()
formData.append('file', file)
formData.append('title', title)
formData.append('year', year)
formData.append('abstrak', abstrak)
formData.append('abstract', abstract)
formData.append('category', category)
formData.append('keywords', keywords)
axios({ //mengirim data ke backend menggunakan axios
  method: 'POST',
  url: `/user/upload/`,
  data: formData,
  headers:{
    'Content-Type':'multipart/form-data',
    'Authorization': this.props.token
  }
}).then((res) =>{ //dilakukan bila unggah sukses
  this.refs.uploadForm.reset() //reset form
  this.setState({
    message:res.data.message,
    status:res.data.status,
    showLoading:false
  })
}).catch((err) => { //dilakukan bila unggah gagal
  this.setState({
    showLoading:false
  })
  if( err.response){
    this.setState({
      message:err.response.data.message,
      status:err.response.data.status,
    })
  }
})
}
}

```

Data yang dikirimkan ke *backend* akan dicek sebelum disimpan di *database*.

Pengecekan yang dilakukan adalah pengecekan kolom data wajib yang tidak boleh kosong dan pengecekan ekstensi dan ukuran *file* yang diunggah. NPM juga di cek untuk mengetahui apakah pengguna telah mengunggah skripsi sebelumnya. Hal ini dilakukan untuk mengatasi unggah skripsi lebih dari satu kali. Berikut adalah potongan kode unggah skripsi pada *backend*:

```

const upload = multer({ //menyimpan file ke local folder
  storage: storage,
  limits:{fileSize: 1024 * 1024* 20},
  fileFilter:fileFilter
}).single('file')

router.post('/upload/', (req, res) =>{
  upload(req, res, (err) => { //callback unggah file
    //Cek error pada upload middleware
    if(err){
      console.log(err)
      return utils.template_response(res, 500, err.message , null)
    }
    //Cek apakah kolom-kolom data diisi
    let { title, year, abstrak, abstract, category, keywords} = req.body
    if (!title || !year || !abstract ) {
      return utils.template_response(res, 400, "Judul, tahun, dan a
bstrak harus diisi" , null)
    }
    if(!req.file){
      return utils.template_response(res, 400, "File skripsi harus
diunggah" , null)
    }
    if (keywords && keywords.length>=255){
      return  utils.template_response(res, 400, "Keywords terlalu
panjang" , null)
    }
    //mendapatkan npm dengan men-decode token
    let bearer = req.get('Authorization')
    let token = bearer.split(' ')[1]
    let payload = jwt.decode(token, secret).request
    //Cek apakah user sudah pernah mengunggah skripsi
    let checkSkripsi = `SELECT * FROM skripsi WHERE
npm='${payload.npm}' LIMIT 1`
    db.query(checkSkripsi, (err, skripsi)=>{
      if (err){
        return utils.template_response(res, 400, err.response, null)
      }
      if(skripsi.length>0){
        return  utils.template_response(res, 422, "Pengguna sudah
pernah mengunggah file" , null)
      }
      let path_url = req.file.path
      let post = {
        id: uuid(),
        npm: payload.npm,
        title: title,
        abstrak:abstrak,
        abstract: abstract,
        published_year: year,
        file_url: path_url,
        category: category,
        keywords: keywords,
        uploaded_at:moment().format()
      }
    })
  })
})

```

```

//query menambahkan data skripsi ke database
let sql = 'INSERT INTO skripsi SET ?'
db.query(sql, post, (err, result)=>{
  if(err){
    console.log(err)
    return utils.template_response(res, 500, err.message , null)
  }
  console.log('success!')
  return utils.template_response(res, 200, "Unggah Berhasil",
null)
})
})
})
})
})

```

Fitur unggah skripsi akan diuji menggunakan *black box testing* dengan skenario seperti yang tertera pada tabel 4.3.

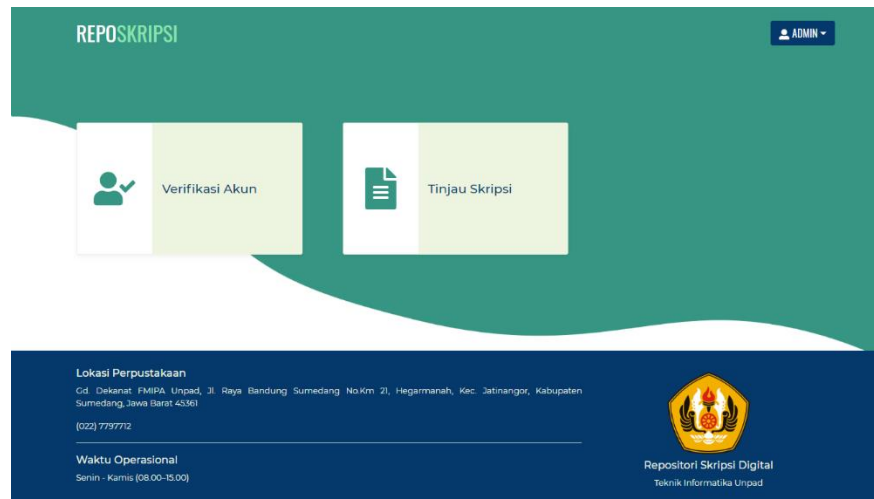
Tabel 4.3 *Black Box Testing* pada Fitur Unggah Skripsi

Skenario	Hasil yang diharapkan	Hasil
Mengosongkan semua kolom data	Pengguna tidak dapat submit data karena tombol submit tidak aktif	Berhasil
Tidak mengisi salah satu kolom data wajib	Pengguna tidak dapat submit data karena tombol submit tidak aktif	Berhasil
Tidak mengisi kolom data opsional	Unggah berhasil dan akan muncul pesan sukses unggah <i>file</i>	Berhasil
Mengunggah <i>file</i> yang bukan pdf	Muncul pesan bahwa <i>file</i> harus berupa pdf	Berhasil
Mengunggah <i>file</i> yang berukuran lebih besar dari 20mb	Muncul pesan bahwa <i>file</i> terlalu besar	Berhasil
Mengisi semua <i>field</i> dengan benar	Unggah berhasil. Muncul pesan sukses unggah <i>file</i>	Berhasil

4.1.5 Halaman Menu Admin

Admin yang telah *login* dibawa ke halaman menu admin. Pada halaman ini terdapat dua pilihan aktivitas yang dapat dilakukan oleh admin yaitu verifikasi akun dan tinjau skripsi. Pada *navbar* terdapat *dropdown* berisi menu untuk masuk ke menu admin dan *logout*. Logo pada kiri *navbar* membawa admin ke halaman utama

web dan melihat skripsi yang telah dipublikasi. Implementasi halaman menu admin dapat dilihat pada gambar 4.6.



Gambar 4.6 Halaman Menu Admin

Kedua menu admin hanya dapat diakses oleh pengguna dengan *role* admin. Oleh karena itu *request* yang terjadi pada verifikasi akun maupun tinjau skripsi akan melalui *middleware* terlebih dahulu. JWT akan dikirimkan melalui *headers authorization* pada *request* yang ada pada menu admin. Kemudian *middleware* mengecek JWT tersebut. Apabila *role* berupa admin proses akan dilanjutkan, bila tidak maka proses tidak akan dilanjutkan dan halaman akan *redirect* ke halaman utama. Berikut adalah potongan kode *middleware* untuk mengakses menu admin:

```
const auth = {
  admin (req, res, next){
    let bearer = req.headers.authorization
    let token = bearer.split(' ')[1]
    jwt.verify(token,secret, function (err, decodedPayload) {
      if (err) {
        console("failed to authorize token")
        return utils.template_response(res, 401, "failed to authorize token", null)
      }
      if (decodedPayload.request.role !== 'admin'){
        console("role not allowed")
        return utils.template_response(res, 401, "role not allowed", null)
      }
    })
  }
}
```

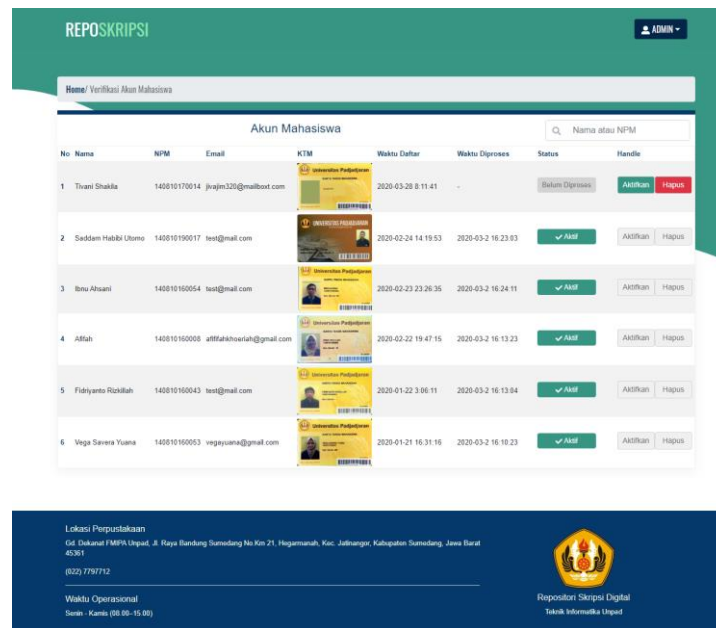


```

    }
    next()
  })
},
...
}

```

4.1.6 Halaman Verifikasi Akun



Gambar 4.7 Halaman Verifikasi Akun

Halaman verifikasi akun hanya dapat diakses oleh admin. Halaman ini berisi tabel data mahasiswa yang telah registrasi dan verifikasi *email*. Pada tabel terdapat nama, NPM, *email*, KTM mahasiswa, waktu daftar, waktu proses, status, dan tombol aksi. Proses yang terjadi pada halaman ini ialah *frontend* melakukan *request* data mahasiswa setelah inisial *rendering*. Kemudian data dari tabel *users* ditampilkan ke halaman web. Implementasi halaman verifikasi akun dapat dilihat pada gambar 4.7.

Pada tabel terdapat dua tombol aksi. Tombol aksi terdiri dari tombol aktifkan dan hapus. Tombol aktifkan akan mengubah nilai *is_active* pada data pengguna

terkait dari 0 menjadi 1. Akun dengan nilai *is_active* = 1 dapat melakukan *login*.

Berikut adalah potongan kode fungsi aktifkan pada *frontend* dan *backend*:

```
activated = (id) => {
  this.setState({ //muncul modal berisi pesan loading
    showLoading:true
  })
  axios({ //mengubah data dengan method put
    method: 'put',
    url: `/admin/activated/${id}`,
    headers:{
      Authorization: this.props.token
    }
  }).then(res=>{ //dilakukan bila verifikasi berhasil
    this.setState({
      showLoading:false
    })
    this.getData()
  }).catch((err) => { //dilakukan bila verifikasi gagal
    this.setState({
      showLoading:false,
      showAlert:true
    })
    if(err.response){
      console.log(err.response)
      this.setState({
        message:err.response.data.message
      })
    }
  })
}
```

```
router.put('/activated/:id', (req, res) =>{
  const npm = req.params.id
  let time=moment().format()
  //query mengubah data pengguna
  let sql = `UPDATE users SET is_active=${true},
processed_at='${time}' where npm='${npm}'`
  db.query(sql, (err, result)=>{
    if (err) {
      console.log(err)
      return utils.template_response(res, 500, 'Gagal' , null)
    }
    //mengambil data nama dan email pengguna
    let findUser = `SELECT name, email FROM users where role='user'
AND npm='${npm}'`
    db.query(findUser, (err, data)=>{
      if (err) console.log(err.response)
      console.log(data[0].email)
      //mengirim email pemberitahuan aktivasi akun
      var helper = require('sendgrid').mail
      var from_email = new helper.Email('no-reply@repositori-
skripsi.com')
      var to_email = new helper.Email(data[0].email)
```

```

var subject = 'Akun Telah Diaktifkan'
var emailText=`<html>
  <body>
    <p>Halo ${data[0].name},</p>
    <p>Anda telah melakukan register akun pada web aplikasi
repositori-skripsi.
    Akun anda telah ditinjau oleh admin dan telah diaktifkan.
    Silahkan login menggunakan npm yang didaftarkan.
  </p>
</body>
</html>`
var content = new helper.Content('text/html', emailText)
var mail = new helper.Mail(from_email, subject, to_email,
content)
var sg = require('sendgrid')(process.env.SENDGRID_API_KEY)
var request = sg.emptyRequest({
  method: 'POST',
  path: '/v3/mail/send',
  body: mail.toJSON(),
})
sg.API(request, function(error, response) {
  console.log(response.statusCode)
  console.log(response.body)
  console.log(response.headers)
})
return utils.template_response(res, 200, 'Berhasil' , null)
})
})
})

```

Tombol hapus akan menghapus data pengguna dari *database*. Saat diklik muncul *pop-up* untuk mencegah menghapus akun secara tidak sengaja. Berikut adalah potongan fungsi kode tombol hapus pada *frontend* dan *backend*:

```

deleteAcc= (id) => {
  this.setState({ //memunculkan modal berisi pesan loading
    showLoading:true
  })
  axios({ //menghapus pengguna dengan method delete axios
    method: 'delete',
    url: `/admin/delete-acc/${id}`,
    headers: {
      Authorization: this.props.token
    }
  }).then(()=>{ //dilakukan bila proses berhasil
    this.setState({
      showModal:false,
      showLoading:false
    })
    this.getData()
  }).catch((err) => { //dilakukan bila proses gagal
    this.setState({
      showModal:false,

```

```

        showLoading:false,
        showAlert:true
    })
    if(err.response){
        console.log(err.response)
        this.setState({
            message:err.response.data.message
        })
    }
    })
}

```

```

router.delete('/delete-acc/:id', (req, res) =>{
    const npm = req.params.id
    let find = `SELECT ktm_url, name, email from users WHERE
role='user' and npm='${npm}' LIMIT 1`
    db.query(find, (err, result)=>{
        if (err) {
            console.log(err)
            return utils.template_response(res, 500, 'Gagal menghapus foto
KTM' , null)
        }
        else{
            let data=result[0]
            //menghapus file ktm dari local folder
            let file_url=data.ktm_url
            fs.unlink(file_url, (err) => {
                if (err) console.log(err)
                else console.log(file_url, 'was deleted')
            })
            //query menghapus user dari database
            let sql = `delete from users where npm='${npm}'`
            db.query(sql, (err, result)=>{
                if (err) {
                    console.log(err)
                    return  utils.template_response(res, 500, 'Akun  gagal
Dihapus' , null)
                }
                console.log('Success')
                //mengirim email pemberitahuan akun tidak diaktifkan
                var helper = require('sendgrid').mail
                var from_email = new helper.Email('no-reply@repositori-
skripsi.com')
                var to_email = new helper.Email(data.email)
                var subject = 'Akun Gagal Diaktifkan'
                var emailText=`<html>
<body>
<p>Halo ${data.name},</p>
<p>Anda telah melakukan registrasi akun pada web aplikasi
repositori-skripsi.
Akun anda telah ditinjau oleh admin namun data yang anda
inputkan tidak benar sehingga akun tidak dapat diaktifkan.
</p>
<p>Silahkan melakukan registrasi ulang pada web, dan
pastikan untuk menginputkan npm dan foto ktm yang valid</p>
</body>

```

```

</html>`
    var content = new helper.Content('text/html', emailText)
    var mail = new helper.Mail(from_email, subject, to_email,
content)
    var sg = require('sendgrid')(process.env.SENDGRID_API_KEY)
    var request = sg.emptyRequest({
        method: 'POST',
        path: '/v3/mail/send',
        body: mail.toJSON(),
    })
    sg.API(request, function(error, response) {
        console.log(response.statusCode)
        console.log(response.body)
        console.log(response.headers)
    })
    return utils.template_response(res, 200, 'Berhasil' , null)
})
}
})
})

```

Fitur verifikasi akun duji menggunakan *black box testing* dengan skenario seperti yang tertera pada tabel 4.4.

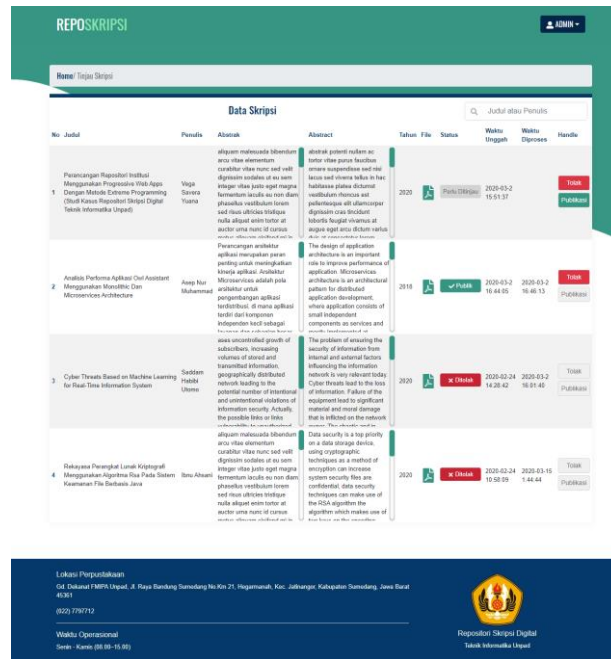
Tabel 4.4 *Black Box Testing* pada Fitur Verifikasi Akun

Skenario	Hasil yang diharapkan	Hasil
Mengklik tombol aktifkan	Tombol aksi akan dinonaktifkan. <i>Email</i> pemberitahuan akun diaktifkan dikirim ke <i>email</i> pengguna	Berhasil
Mengklik tombol hapus	Muncul <i>pop-up</i> untuk konfirmasi tindakan admin. Bila yakin akun tidak terverifikasi, akun akan dihapus dari tabel. Pengguna dikirim <i>email</i> pemberitahuan bahwa akun tidak diaktifkan	Berhasil

4.1.7 Halaman Tinjau Skripsi

Halaman tinjau skripsi adalah halaman yang hanya dapat diakses oleh admin. Halaman ini berisi data skripsi yang telah diunggah mahasiswa. Pada tabel terdapat judul, penulis, tahun dipublikasi, abstrak, abstract, *file* skripsi, status, waktu unggah, waktu proses, dan tombol aksi. Proses yang terjadi pada halaman ini ialah

frontend melakukan *request* data. Data yang ditampilkan merupakan hasil *join* tabel skripsi dan *users*. Data yang diambil dari tabel *users* ialah nama penulis. Data lainnya diambil dari tabel skripsi. Implementasi halaman tinjau skripsi dapat dilihat pada gambar 4.8



Gambar 4.8 Halaman Tinjau Skripsi

Pada tabel terdapat dua tombol aksi. Tombol aksi terdiri dari tombol tolak dan publikasi. Tombol publikasi mengubah nilai *is_approved* pada data skripsi terkait dari 2 menjadi 1. Nilai 2 menandakan kondisi awal dimana skripsi belum ditinjau. Berikut adalah potongan kode fungsi *approved* pada *frontend* dan *backend*:

```
approved = (id) => { //fungsi tombol publikasi
  this.setState({ //memunculkan modal berisi pesan loading
    showLoading:true
  })
  axios({ //mengubah status skripsi menggunakan method put axios
    method: 'put',
    url: `/admin/approved/${id}`,
    headers: {
      Authorization: this.props.token
    }
  }).then(res=>{ //dilakukan bila proses sukses
```

```

    this.setState({
      showLoading:false
    })
    this.getData()
  }).catch((err) => { //dilakukan bila proses gagal
    this.setState({
      showLoading:false,
      showAlert:true
    })
    if(err.response){
      console.log(err.response)
      this.setState({
        message:err.response.data.message
      })
    }
  })
}
}

```

```

router.put('/approved/:id', (req, res) =>{
  const id = req.params.id
  let time=moment().format()
  //query mengubah nilai is_approved pada database
  let sql = `UPDATE skripsi SET is_approved=${true},
processed_at='${time}' where id='${id}'`
  db.query(sql, (err, result)=>{
    if (err) {
      console.log(err)
      return utils.template_response(res, 500, 'Gagal' , null)
    }
    console.log('approved')
    return utils.template_response(res, 200, 'Berhasil' , null)
  })
})

```

Tombol tolak mengubah nilai *is_approve* pada data skripsi terkait menjadi 0.

Berikut adalah kode fungsi *unapproved* pada *frontend* dan *backend*:

```

unapproved = (id) => { //fungsi tombol tolak
  this.setState({ //memunculkan modal loading
    showLoading:true
  })
  axios({ //mengubah status skripsi dengan method put axios
    method: 'put',
    url: `/admin/unapproved/${id}`,
    headers: {
      Authorization: this.props.token
    }
  }).then(()=>{ //dilakukan bila proses sukses
    this.setState({
      showModal:false,
      showLoading:false
    })
    this.getData()
  }).catch((err) => {

```

```

        this.setState({ //dilakukan bila proses gagal
            showModal:false,
            showLoading:false,
            showAlert:true
        })
        if(err.response){
            console.log(err.response.statusText)
            this.setState({
                message:err.response.data.message
            })
        }
    })
}

```

```

router.put('/unapproved/:id', (req, res) =>{
    const id = req.params.id
    let time=moment().format()
    //query mengubah nilai is_approved pada database
    let sql = `UPDATE skripsi SET is_approved=${false},
processed_at='${time}' where id='${id}'`
    db.query(sql, (err, result)=>{
        if (err) {
            console.log(err)
            return utils.template_response(res, 500, 'Gagal' , null)
        }
        console.log('unapproved')
        return utils.template_response(res, 200, 'Berhasil' , null)
    })
})

```

Fitur tinjau skripsi diuji menggunakan *black box testing* dengan skenario yang tertera pada tabel 4.5.

Tabel 4.5 *Black Box Testing* pada Fitur Tinjau Skripsi

Skenario	Hasil yang diharapkan	Hasil
Mengklik tombol publikasi	Tombol publikasi akan di nonaktifkan, status skripsi akan berubah, dan skripsi akan muncul dihalaman utama	Berhasil
Mengklik tombol tolak	Pop-up konfirmasi tindakan akan muncul bila disetujui kedua tombol aksi akan dinonaktifkan dan status skripsi akan berubah	Berhasil

4.1.8 Halaman Detail Skripsi



Gambar 4.9 Halaman Detail Skripsi

Halaman detail skripsi dapat diakses saat mengklik salah satu skripsi pada halaman utaman. Halaman ini berisi info detail dari skripsi yang dipilih. Info skripsi dibagi menjadi tiga bagian yaitu identitas yang berisi judul, pengarang, tahun, kategori, dan kata kunci. Bagian kedua berisi abstrak dan bagian ketiga berisi *file* skripsi. Bagian ketiga yang berisi pratinjau skripsi hanya dapat dilihat saat pengguna telah *login*.

Hasil implementasi halaman detail skripsi dapat dilihat pada gambar 4.9. Berikut adalah potongan kode halaman detail skripsi pada *frontend*. Apabila telah

login, *request* data akan mengirimkan otorisasi. Sebaliknya saat tidak *login*, web melakukan *request* ke url berbeda tanpa mengirimkan otorisasi.

```

getData =()=>{
  let id = this.props.match.params.id
  if(this.props.token){
    axios({
      method: 'get',
      url: `/skripsi/detail/`,
      params:{
        id : id
      },
      headers: {
        Authorization: this.props.token
      }
    }).then(res=>{
      this.setState({
        skripsi: res.data[0],
        isLoading: true
      })
    }).catch(err=>{
      if(err.response){
        console.log(err.response)
      }
    })
  }
  else{
    axios({
      method: 'get',
      url: `/skripsi/info/`,
      params:{
        id : id
      }
    }).then(res=>{
      this.setState({
        skripsi: res.data[0],
        isLoading: true
      })
    }).catch(err=>{
      if(err.response){
        console.log(err.response)
      }
    })
  }
}

```

4.1.9 Fitur Pencarian dan Penyaringan

Fitur yang diimplementasikan selanjutnya ialah fitur pencarian dan penyaringan. Pada halaman utama terdapat kolom pencarian skripsi dan pilihan

penyaringan skripsi berdasarkan tahun maupun bidang minat. Proses penyaringan dilakukan di sisi klien. Fitur pencarian dan penyaringan diuji dengan skenario pada tabel 4.6. Berikut adalah kode fitur pencarian dan penyaringan:

```

onChange =(e)=>{ //fungsi pencarian
  let text = e.target.value.toLowerCase().trim()
  let {skripsiFilteredTemp} = this.state
  const filteredData = skripsiFilteredTemp.filter(item => {
    if (item.keywords){
      return item.title.toLowerCase().includes(text) ||
item.name.toLowerCase().includes(text) ||
item.keywords.toLowerCase().includes(text)
    }
    return item.title.toLowerCase().includes(text) ||
item.name.toLowerCase().includes(text)
  })
  this.setState({
    skripsiFiltered:filteredData,
  })
}
yearFilter = (e)=>{ //filter berdasarkan tahun
  let year = e.target.id
  let {skripsi, cat} = this.state
  this.setState({
    currentPage:1
  })
  //All
  if(year==='Tahun'){
    if (cat){
      let filteredData = skripsi.filter(item => {
        return item.category===cat
      })
      this.setState({
        skripsiFiltered:filteredData,
        skripsiFilteredTemp:filteredData,
      })
    }
    else{
      this.setState({
        skripsiFiltered:skripsi,
        skripsiFilteredTemp:skripsi,
      })
    }
  }
  this.setState({
    year:null,
    yearSelection:'Tahun',
  })
}
//Costumize
else{
  let filteredData
  if(cat){
    filteredData = skripsi.filter(item => {

```

```

        return item.published_year == year && item.category== cat
    })
}
else{
    filteredData = skripsi.filter(item => {
        return item.published_year == year
    })
}
this.setState({
    year:year,
    yearSelection:year,
    skripsiFiltered:filteredData,
    skripsiFilteredTemp:filteredData,
})
}
}
categoryFilter = (e)=>{
    let cat = e.target.id
    let text = e.target.innerText
    let {skripsi, year} = this.state
    this.setState({
        currentPage:1
    })
    //All
    if (cat==='all'){
        if (year){
            let filteredData = skripsi.filter(item => {
                return item.published_year==year
            })
            this.setState({
                skripsiFiltered:filteredData,
                skripsiFiltereTemp:filteredData
            })
        }
        else{
            this.setState({
                skripsiFiltered:skripsi,
                skripsiFilteredTemp:skripsi,
            })
        }
        this.setState({
            cat:null,
            categorySelection:'Bidang Minat',
        })
    }
    //Costumize
    else{
        let filteredData
        if(year){
            filteredData = skripsi.filter(item => {
                return item.published_year == year && item.category== cat
            })
        }
        else{
            filteredData = skripsi.filter(item => {
                return item.category==cat
            })
        }
    }
}

```

```

    })
  }
  this.setState({
    cat:cat,
    categorySelection:text,
    skripsiFiltered:filteredData,
    skripsiFilteredTemp:filteredData,
  })
}
}

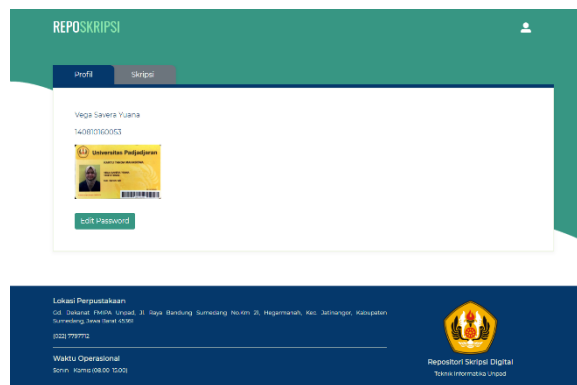
```

Tabel 4.6 *Black Box Testing* pada Fitur Pencarian dan Penyaringan

Skenario	Hasil yang diharapkan	Hasil
Memasukkan kata kunci pada kolom pencarian	Menampilkan skripsi dengan kata kunci terkait	Berhasil
Menyaring berdasarkan tahun	Menampilkan skripsi dengan tahun yang dicari	Berhasil
Menyaring berdasarkan bidang minat	Menampilkan skripsi dengan bidang minat yang dipilih	Berhasil
Menyaring berdasarkan tahun dan bidang minat	Menampilkan skripsi dengan tahun dan bidang minat yang dicari	Berhasil
Menyaring berdasarkan tahun dan bidang minat kemudian memasukkan kata kunci pada kolom pencarian dan	Menampilkan pencarian skripsi berdasarkan data yang telah disaring berdasarkan tahun dan bidang minat	Berhasil

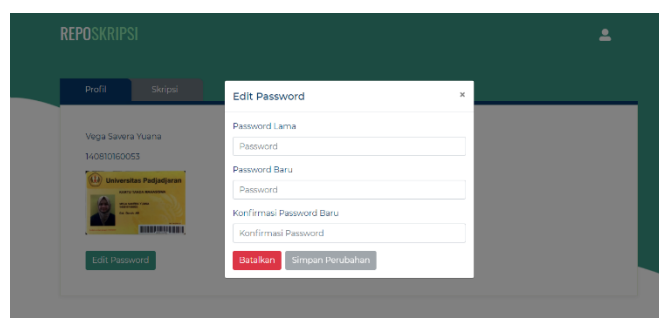
4.1.10 Halaman Profil dan Fitur *Edit Password*

Halaman profil berisi data diri mahasiswa yang diisikan pada saat registrasi. Proses yang terjadi pada halaman ini ialah *frontend* melakukan *request* data mahasiswa dan menampilkannya. Data yang ditampilkan ialah nama, NPM, dan foto KTM. Implementasi halaman profil dapat dilihat pada gambar 4.10.



Gambar 4.10 Halaman Profil

Pada halaman ini mahasiswa dapat mengubah *password*. Untuk mengubah *password*, mahasiswa harus mengisi *password* lama, *password* baru, dan konfirmasi *password*. Implementasi bagian *edit password* dapat dilihat pada gambar 4.11.

Gambar 4.11 Bagian *Edit Password*

Proses yang terjadi pada fitur *edit password* ialah permintaan mengubah *password* menggunakan *method put*. Sebelum melakukan pembaharuan, inputan dicek terlebih dahulu. *Password* lama harus sesuai dengan yang tersimpan di *database* dan *password* baru harus berbeda dengan *password* lama. Berikut adalah kode fitur *edit password*:

```
submit=(e)=>{
  e.preventDefault()
  this.setState({ //memunculkan modal berisi pesan loading
    showLoading:true
```

```

    })
    let { newPass, oldPass } = this.state
    axios({ //mengubah password dengan method put axios
      method: 'put',
      url: `/user/edit-pass`,
      headers:{
        Authorization: this.props.token
      },
      data: {
        newPass:newPass,
        oldPass:oldPass
      }
    }).then(res => { //dilakukan bila proses berhasil
      this.refs.editForm.reset();
      this.setState({
        newPass:'',
        oldPass:'',
        message:res.data.message,
        status:res.data.status,
        showLoading:false
      })
    }).catch((err) => { //dilakukan bila proses gagal
      this.setState({
        showLoading:false
      })
      if(err.response){
        this.setState({
          message:err.response.data.message,
          status:err.response.data.status,
        })
      }
    })
  }
}

```

```

router.put('/edit-pass', async(req, res) =>{
  try{
    let bearer = req.headers.authorization
    let token = bearer.split(' ')[1]
    let {newPass, oldPass} = req.body
    //cek apakah field diisi
    if (!newPass || !oldPass ) {
      return utils.template_response(res, 400, "Semua field harus diisi" , null)
    }
    //decode payload untuk mendapatkan id user
    let payload = jwt.decode(token, secret).request
    if(payload==={}){
      console.log('Need Login info')
      return
    }
    let findOldPass = `SELECT password FROM users where npm='${payload.npm}' limit 1`
    db.query(findOldPass, async(err, result)=>{
      try{
        if( await bcrypt.compare(oldPass, result[0].password)){
          console.log('Old password matches the database')

```

```

        if (oldPass===newPass){
            return utils.template_response(res, 400, "Password baru
            harus berbeda dari password lama", null)
        }
        else{
            let password = await bcrypt.hash(newPass, 10)
            //query mengubah password ke database
            let sql = `UPDATE users SET password='${password}' where
            npm='${payload.npm}'`
            db.query(sql, (err, result)=>{
                if (err) console.log(err)
                return utils.template_response(res, 200, "Edit Password
                Berhasil", null)
            })
        }
    }
    else{
        return utils.template_response(res, 400, "Password lama
        salah", null)
    }
}
catch(err){
    console.log(err.response)
}
})
}
catch(err){
    console.log(err)
}
})
})

```

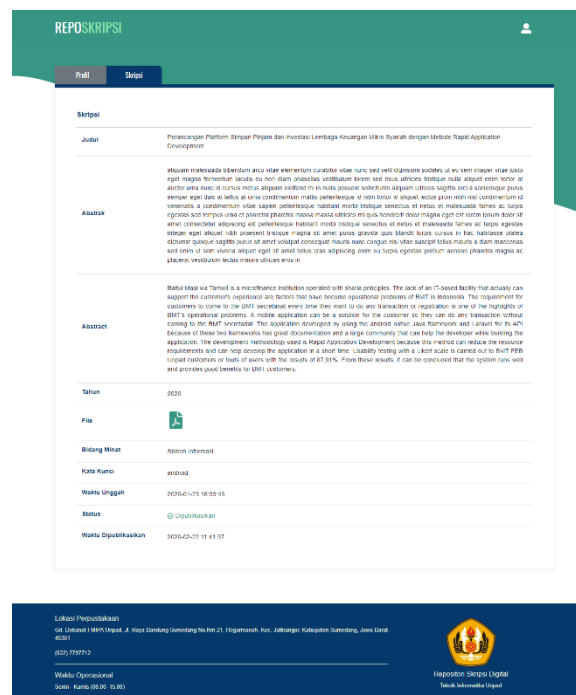
Fitur *edit password* diuji menggunakan *black box testing* dengan skenario yang dapat dilihat pada tabel 4.7.

Tabel 4.7 *Black Box Testing* pada Fitur *Edit Password*

Skenario	Hasil yang diharapkan	Hasil
Tidak mengisi semua kolom data	Tombol simpan perubahan tidak aktif. Tombol batalkan aktif	Berhasil
<i>Password lama</i> tidak diinput dengan <i>password</i> yang benar	Muncul pesan bahwa <i>password</i> lama tidak cocok dengan data yang tersimpan di <i>database</i> setelah klik simpan perubahan	Berhasil
Tidak mengisi konfirmasi <i>password</i>	Tombol simpan perubahan tidak aktif	Berhasil
Konfirmasi <i>password</i> dan <i>password</i> baru berbeda	Muncul pesan konfirmasi <i>password</i> tidak cocok setelah	Berhasil

Skenario	Hasil yang diharapkan	Hasil
	selesai input dan tombol simpan perubahan tidak aktif	
Mengisi kolom <i>password</i> lama dengan benar, mengisi <i>password</i> baru yang berbeda dengan <i>password</i> lama, dan mengisi konfirmasi <i>password</i> dengan nilai yang sama dengan <i>password</i> baru	Muncul pesan bahwa ubah <i>password</i> berhasil dilakukan	Berhasil

4.1.11 Halaman Status Skripsi



Gambar 4.12 Halaman Status Skripsi

Halaman status skripsi berisi informasi skripsi yang telah diunggah oleh mahasiswa. Bila mahasiswa belum mengunggah apapun maka halaman menampilkan pesan bahwa pengguna belum mengunggah skripsi. Pada halaman ini mahasiswa dapat melihat judul, abstrak, abstract, tahun publikasi, *file* skripsi,

bidang minat, kata kunci, waktu unggah, status skripsi, dan waktu dipublikasi. Hasil implementasi halaman status skripsi dapat dilihat pada gambar 4.12.

Skripsi dengan status belum ditinjau atau ditolak akan menampilkan tombol edit unggahan pada bagian bawah. Sementara skripsi dengan status dipublikasi tidak dapat di *edit* lagi. Berikut adalah kode pada halaman status skripsi pada *frontend* dan *backend*:

```
getSkripsi=()=>{
  axios({ //request data pengguna dengan method get axios
    method: 'get',
    url: `/user/skripsi/`,
    headers: {
      Authorization: this.props.token
    }
  }).then(res=>{ //dilakukan bila proses berhasil
    this.setState({
      skripsi: res.data,
      isLoading: true
    })
  }).catch(err=>{ //dilakukan bila proses gagal
    if(err.response){
      console.log(err.response)
    }
  })
}
```

```
router.get('/skripsi', (req, res) =>{
  let bearer = req.headers.authorization
  let token = bearer.split(' ')[1]
  let payload = jwt.decode(token, secret).request
  let sql = `SELECT * FROM skripsi WHERE npm='${payload.npm}' LIMIT 1`
  db.query(sql, (err, result)=>{
    if (err) console.log(err)
    res.send(result[0])
  })
})
```

Tombol edit unggahan mengarahkan ke *form* unggah ulang. Mahasiswa akan melakukan proses yang sama dengan proses unggah skripsi. Namun melainkan membuat *record* baru seperti pada fungsi unggah, unggah ulang akan meng-*update*

record lama. *File* yang disimpan sebelumnya akan dihapus dan digantikan dengan *file* baru. Berikut adalah kode API unggah ulang skripsi:

```
router.put('/reupload/', (req, res) =>{
  upload(req, res, (err) => { callback upload file
    //Cek error pada upload middleware
    if(err){
      console.log(err)
      return utils.template_response(res, 500, err.message , null)
    }
    //Cek apakah kolom data diisi
    let { title, year, abstract, category, keywords} = req.body
    console.log(req.body)
    if (!title || !year || !abstract || !req.file) {
      return utils.template_response(res, 400, "Semua field harus
diisi" , null)
    }
    if(!req.file){
      return utils.template_response(res, 400, "File skripsi harus
diunggah" , null)
    }
    if(keywords && keywords.length>=255){
      return utils.template_response(res, 400, "Kata kunci terlalu
banyak", null)
    }
    let bearer = req.get('Authorization')
    let token = bearer.split(' ')[1]
    //decode jwt untuk mendapatkan id user
    let payload = jwt.decode(token, secret).request
    //mengambil data skripsi yang sudah diunggah sebelumnya
    let checkSkripsi = `SELECT * FROM skripsi WHERE
npm='${payload.npm}' LIMIT 1`
    db.query(checkSkripsi, (err, skripsi)=>{
      console.log('data', skripsi[0])
      if (err){
        console.log('err', err)
        return utils.template_response(res, 400, err.response, null)
      }
      if(skripsi.length===0){
        return utils.template_response(res, 422, "Pengguna belum
mengunggah file" , null)
      }
      if(skripsi[0].is_approve===1){
        return utils.template_response(res, 422, "File sudah
dipublikasikan" , null)
      }
      let old_file=skripsi[0].file_url
      console.log('Old File', old_file)
      //hapus file lama daro local folder
      fs.unlink(old_file, (err) => {
        if (err) console.log(err);
        console.log(old_file, 'was deleted');
      })
      let id = skripsi[0].id
```

```

let path_url = req.file.path
console.log('new file', path_url)
let post = {
  title: title,
  abstract: abstract,
  published_year: year,
  file_url: path_url,
  category: category,
  keywords: keywords
}
//query mengubah unggahan
let sql = `UPDATE skripsi SET
uploaded_at='${moment().format()}', is_approved=${2}, ? where
id='${id}'`
db.query(sql, post, (err, result)=>{
  if(err){
    console.log(err)
    return utils.template_response(res, 500, err.message , null)
  }
  console.log('Success!')
  return utils.template_response(res, 200, "Unggah Ulang
berhasil", null)
})
})
})
})

```

4.1.12 Implementasi PWA

Fitur PWA yang diimplementasikan pada web repositori skripsi Teknik Informatika Unpad ialah kemampuan web untuk tetap mempertahankan tampilannya pada saat *offline* dan fitur *add to homescreen*. Langkah pertama ialah mengubah fungsi *service worker default* CRA (Create-react-app) dari *unregister* menjadi *register*. Fungsi tersebut dipanggil pada halaman *index.js*

```

import * as serviceWorker from './serviceWorker'
ReactDOM.render(<App />, document.getElementById('root'))
serviceWorker.register()

```

PWA memiliki *lifecycle* yaitu *register*, *install*, dan *activate*. File *serviceWorker* berisi fungsi yang menjalankan *lifecycle* tersebut. Ketika keseluruhan halaman telah di-*load*, *service worker* akan di *register*. Fungsi `window.addEventListener('load', () => {...})` dipanggil untuk mengetahui apakah

halaman telah di-load. Selanjutnya *service worker* akan di *install* dan di aktifkan. *App shell* akan di *cache* oleh *service worker* sehingga saat *offline* sekalipun web tetap memiliki tampilan. *App shell* adalah HTML, CSS, dan JavaScript minimal yang diperlukan untuk memberi antarmuka pengguna.

Pada saat *offline*, filter *grayscale* ditambahkan pada *body* dan pesan bahwa pengguna sedang *offline* akan muncul selama 5 detik. Berikut adalah kode menambahkan class *offline* pada *body*:

```
function checkOnline(){
  if (navigator.onLine) {
    document.body.classList.remove('offline')
  } else {
    document.body.classList.add('offline')
  }
}
checkOnline()
function handleNetworkChange(event) {
  checkOnline()
}
window.addEventListener('online', handleNetworkChange)
window.addEventListener('offline', handleNetworkChange)
```

```
body.offline{
  -moz-filter: grayscale(100%);
  -webkit-filter: grayscale(100%);
  filter: gray;
  filter: grayscale(100%);
}
```

Pada saat *offline*, muncul pemberitahuan bahwa pengguna sedang *offline*. Perubahan CSS dilakukan untuk membedakan keadaan *offline* dan keadaan *online*. Pesan dapat ditampilkan saat *offline* dengan memanipulasi CSS. Implementasi fitur *offline* dapat dilihat pada gambar 4.13. Berikut adalah kode untuk menampilkan pesan *offline*:

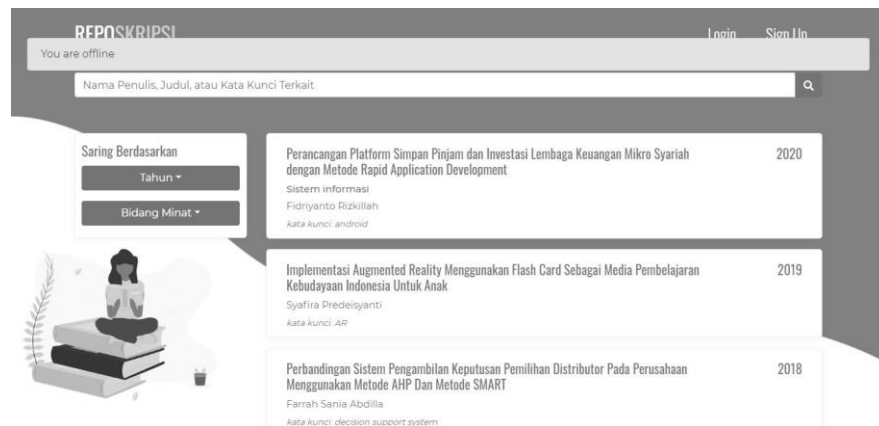
```
<div className="alert alert-secondary alert-offline" id="alert-offline" role="alert">Anda sedang offline</div>
```

```
.offline .alert-offline{
  display: block;
```

```

    animation: cssAnimation 0s 5s forwards;
    transition: 0.5s;
    opacity: 1;
  }
  @keyframes cssAnimation {
    to { opacity: 0; }
  }

```



Gambar 4.13 Implementasi Fitur *Offline*

Untuk membuat fitur tambahan ke *homescreen*, manifest perlu ditambahkan kedalam program. Manifest aplikasi web adalah file JSON sederhana yang memberi tahu *browser* tentang aplikasi web dan bagaimana seharusnya web berperilaku ketika 'diinstal' pada perangkat. Kode manifest ditambahkan pada halaman `index.html` melalui link tag. Berikut adalah kode manifest:

```
<link rel='manifest' href='%PUBLIC_URL%/manifest.json' />
```

```

{
  "short_name": "Repo Skripsi",
  "name": "Repositori Skripsi Teknik Informatika Unpad",
  "icons": [
    {
      "src": "./logo-192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "./logo-512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": "/",

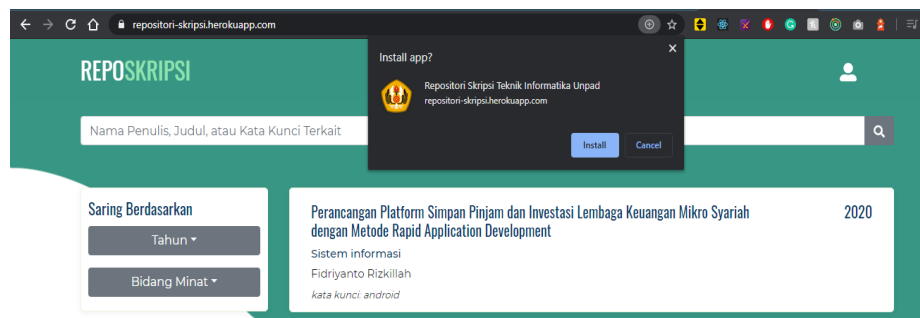
```

```

"display": "standalone",
"theme_color": "#379683",
"background_color": "#379683"
}

```

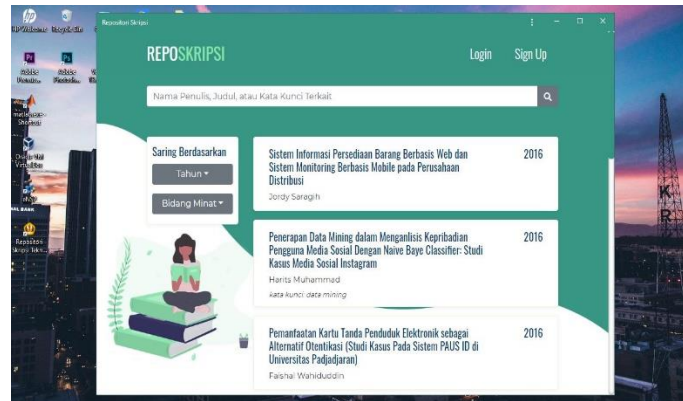
Short_name dan *name* digunakan sebagai nama aplikasi, judul saat *launcher/splash screen*, dan *prompt install*. *Short_name* digunakan apabila *name* terlalu panjang untuk ruang judul yang terbatas. *Icons* digunakan sebagai *icon app* pada *homescreen* dan gambar pada *splash screen*. *start_url* memberi tahu *browser route* dimana aplikasi mulai. *background_color* digunakan sebagai warna *splash screen*. *theme_color* akan memberi warna kustom pada *toolbar device* sehingga aplikasi lebih berkesan *native*. *Display* berisi info tampilan web sebagai *native app*. Ada beberapa jenis *display* yang dapat dipilih. Untuk menampilkan *prompt add to homescreen*, tampilan harus diatur ke *standalone*. Implementasi fitur *add to homescreen* dapat dilihat pada gambar 4.14.



Gambar 4.14 Implementasi Fitur *add to homescreen*

Tombol tambahkan ke *homescreen* terdapat pada *address bar*. Saat di klik muncul pesan seperti gambar 4.15. Dengan mengklik *install*, *shortcut* ditambahkan pada *homescreen*. Web Repositori Skripsi Teknik Informatika Unpad dapat diakses melalui *shortcut* tersebut. Web tampil dalam bentuk aplikasi seperti yang terlihat

pada gambar 4.15. Fitur PWA diuji dengan skenario *black box testing* pada tabel 4.8.



Gambar 4.15 Web Dalam Bentuk *Native App*

Tabel 4.8 *Black Box Testing* Fitur PWA

Skenario	Hasil yang diharapkan	Kesimpulan
Pengguna yang sebelumnya telah mengakses halaman, <i>me-refresh</i> halaman saat koneksi <i>offline</i>	Halaman tetap mempertahankan tampilannya dan memberikan pemberitahuan bahwa pengguna sedang <i>offline</i>	Berhasil
Pengguna ingin menambahkan aplikasi web ke <i>homescreen</i> perangkat	Terdapat fitur add to homescreen	Berhasil

4.2 Fase Produksi

4.2.1 Rilis kecil

Rilis kecil merupakan hasil pertama dari metode XP. Rilis kecil adalah web aplikasi yang telah memenuhi kriteria *User story* pada iterasi pertama. Pada penelitian ini, rilis kecil adalah web aplikasi dengan fitur dan fungsi sebagai berikut: register, *login*, unggah skripsi, verifikasi akun, tinjau skripsi, detail skripsi, pencarian dan penyaringan, *edit password*, cek status skripsi, dan fitur PWA yaitu tampilan pada saat *offline* dan fungsi *add to homescreen*.

Rilisan kecil diujikan kepada pengguna menggunakan *usability testing*. Web aplikasi di *deploy* menggunakan heroku untuk memudahkan proses pengujian dan sebagai alat bantu demo.

4.2.2 Pengujian

Pengujian dilakukan kepada 30 orang mahasiswa teknik Informatika Unpad dengan menggunakan skenario *usability testing* seperti yang dijelaskan pada tabel 3.8. Setelah melakukan skenario tabel 3.8, responden mengisi *form* penilaian yang dapat dilihat pada tabel 3.9. Hasil dari form kemudian dihitung menggunakan skala likert. Hasil pengujian dapat dilihat pada tabel 4.9.

Tabel 4.9 Hasil Pengujian Rilis Kecil

[illegible]

Dari hasil *usability testing* dapat dilihat bahwa rata-rata fitur-fitur pada web repositori skripsi Teknik Informatika Unpad telah mencapai nilai likert diatas 80. Web aplikasi memiliki nilai rata-rata 88.53, masuk ke kategori sangat baik. Maka dapat dikatakan fitur-fitur yang diimplementasikan sudah sangat baik secara fungsi maupun tampilan bagi pengguna.

Pertanyaan 9 dan 11 memiliki nilai likert 79.3 dan 78. Pertanyaan 9 merupakan penilaian terhadap fungsi edit unggahan, sementara pertanyaan 11 merupakan penilaian terhadap tampilan halaman profil pengguna. Poin ini dijadikan *feedback* untuk diperbaiki pada fase pemeliharaan. Selain itu, dari *form* pengujian didapat *feedback* lainnya pada kolom pertanyaan terakhir. *Feedback* ini akan dijadikan poin-poin pada fase pemeliharaan

4.3 Fase Pemeliharaan dan Fase Akhir

Fase pemeliharaan adalah implementasi *feedback* yang didapat dari pengujian terhadap responden. Masukan dari pembimbing juga diimplementasikan pada fase ini. Karena penelitian hanya melakukan 1 iterasi maka hasil dari fase pemeliharaan adalah final produk.

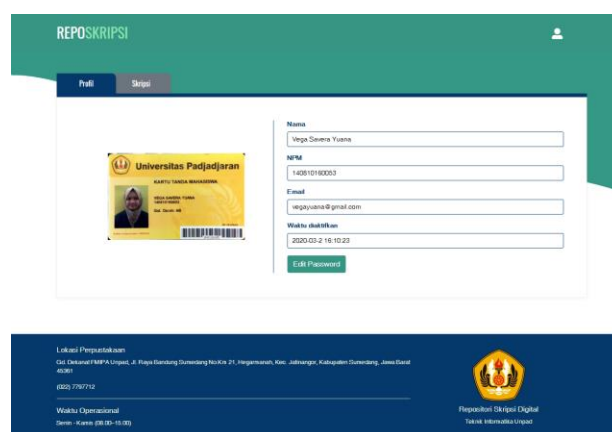
4.3.1 Feedback

Pada form *usability testing* terdapat pertanyaan mengenai *feedback* terhadap fitur-fitur web maupun web secara keseluruhan. Hasil pertanyaan ini dirangkum dan diambil poin-poin penting yang berguna untuk meningkatkan kepuasan pengguna terhadap web. Secara keseluruhan tidak ada penambahan *user story* pada web. Berikut adalah poin-poin yang diperbaiki pada fase pemeliharaan:

1. Memperbaiki tampilan pada halaman profil
2. Menghilangkan tombol *search* pada kolom pencarian untuk menghindari ambiguitas
3. Pada saat unggah ulang, data unggahan awal tetap ada pada *form*
4. Menambahkan *indicator* pada menu navigasi di-*smartphone*
5. Memperbaiki tampilan *form login* pada *smartphone* karena pada beberapa *device*, *form login* tertutup oleh keyboard.
6. Menambahkan forum untuk komunikasi antara admin dan mahasiswa
7. Menambahkan fitur lupa *password*

Fitur tambahan yang didapat dari hasil *feedback* menghasilkan penambahan *user story*, dan desain antarmuka serta menghasilkan perubahan pada analisis sistem, ERD dan *database* dari perencanaan awal. Perubahan dan penambahan yang terjadi telah dicantumkan pada BAB III.

4.3.2 Implementasi *Feedback*



Gambar 4.16 Tampilan Profil Setelah Diperbaharui

Dilakukan implementasi program dari hasil *feedback* yang didapat pada fase pemeliharaan. Banyak responden yang tidak menyukai tampilan profil karena

dirasa terlalu kosong dan tidak menampilkan informasi dengan menarik. Maka dilakukan perubahan tampilan pada halaman profil menjadi seperti gambar 4.16.

Poin selanjutnya ialah tombol *search*. Tombol *search* pada kolom pencarian memberikan ambiguitas kepada pengguna karena pencarian dilakukan dengan method *onChange* yang memberikan hasil pencarian secara langsung tanpa menklik tombol. Tombol *search* tidak diperlukan maka dihilangkan dari tampilan.

Gambar 4.17 Halaman Edit Unggahan

Pada saat pengguna ingin mengedit unggahan skripsi, pengguna diminta untuk mengisi form seperti saat mengunggah skripsi namun pada *backend* proses yang terjadi adalah mengubah data unggahan lama. Pengguna Karena proses yang berbeda maka lebih baik bila pengguna tidak harus mengisi semua data dengan data baru. Melainkan data lama tetap tersedia dalam *form* dan pengguna dapat mengubahnya. Implementasi halaman edit unggahan dapat dilihat pada gambar 4.17.

Saat insial rendering dilakukan, data skripsi milik pengguna ditampilkan sebagai *defaultValue* pada form. Berikut adalah potongan kode pada *frontend*:

```
checkSkripsi={()=>{
```

```

axios({
  method: 'get',
  url: `/user/skripsi/`,
  headers: {
    Authorization: this.props.token
  }
}).then(res=>{
  this.setState({
    skripsi: res.data,
    isLoading: true,
    title: res.data.title,
    year: res.data.published_year.toString(),
    abstrak: res.data.abstrak,
    abstract: res.data.abstract,
    category: res.data.category,
    keywords: res.data.keywords,
  })
}).catch((err) => {
  if (err.response) {
    console.log(err.response)
  }
})
}
...
<textarea      type="text"      id="title"      maxLength="255"
onBlur={this.handleInput}      className="form-control"
placeholder="Judul Skripsi" defaultValue={skripsi.title}/>
...
<input    type="number"    id="year"    onBlur={this.handleInput}
className="form-control" placeholder="Tahun Publikasi Skripsi"
defaultValue={skripsi.published_year}/>
...
<textarea id="abstrak" onBlur={this.handleInput} className="form-
control"      placeholder="Input      Abstrak"
defaultValue={skripsi.abstrak}/>
...
<textarea      id="abstract"      onBlur={this.handleInput}
className="form-control"      placeholder="Input      Abstrak"
defaultValue={skripsi.abstract}/>
...
<select  className="custom-select"  onChange={this.handleInput}
id="category"  defaultValue={skripsi.category}>
...
<input  type="text"  id="keywords"  onChange={this.handleInput}
className="form-control"      placeholder="Kata      Kunci"
defaultValue={skripsi.keywords}/>

```

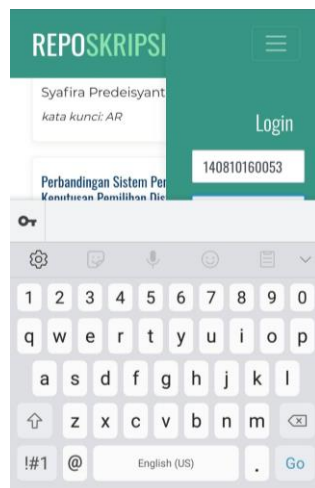
Saat pengguna *login* menggunakan *smartphone*, terdapat navigator pada bagian bawah untuk memudahkan pengguna berpindah menu. Navigator tersebut sebelumnya berwarna hijau secara *default* dan tidak menunjukkan *route* dimana

pengguna berada. Pada fase pemeliharaan ini, navigator dibuat berubah warna dan memberikan kesan aktif sesuai dengan *route* yang diakses oleh pengguna.

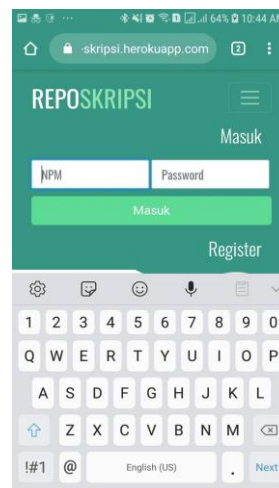


Gambar 4.18 Navigator Menu

Beberapa pengguna mengeluhkan tampilan *login* yang tertutup oleh keyboard *smartphone*. Oleh karena itu tampilan *login* diubah posisinya menjadi berada dibagian atas.



(a)



(b)

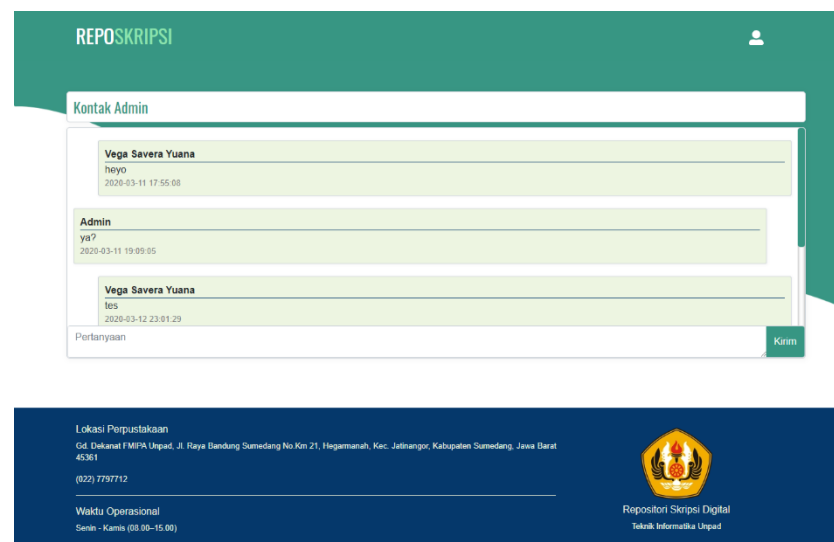
Gambar 4.19 Menu *Login* pada *Smartphone* (a) Lama (b) Baru

Implementasi yang dibahas sebelumnya adalah *feedback* minor yang tidak menambah *user story*. *Feedback* yang menambahkan *user story* ialah penambahan

fitur kontak admin dan lupa *password*. Kedua fitur tersebut diimplementasikan pada fase ini.

1. Kontak Admin

Kontak admin adalah fitur yang ditambahkan karena dirasa perlu adanya fitur komunikasi antara mahasiswa dan admin. Mahasiswa dapat mengirim pesan kepada admin melalui web dengan mengakses menu kontak admin. Implementasi fitur kontak admin dapat dilihat pada gambar 4.20.



Gambar 4.20 Kontak Admin

Pesan dikirimkan melalui method post untuk disimpan ke tabel *forums*. Id pesan diisi dengan 12 digit NPM dan 5 digit tambahan sebagai digit unik bagi tiap pesan yang dikirim. Pesan ditampilkan di web. Pesan yang ditampilkan adalah pesan yang memiliki id dengan 12 digit pertama yang sama dengan NPM pengguna. Halaman kontak admin diuji dengan skenario *black box testing* pada tabel 4.10. Berikut adalah potongan kode untuk mengirim dan menampilkan pesan:

```
//mengirim pesan
router.post('/insert-text', (req, res) =>{
  let bearer = req.headers.authorization
```

```

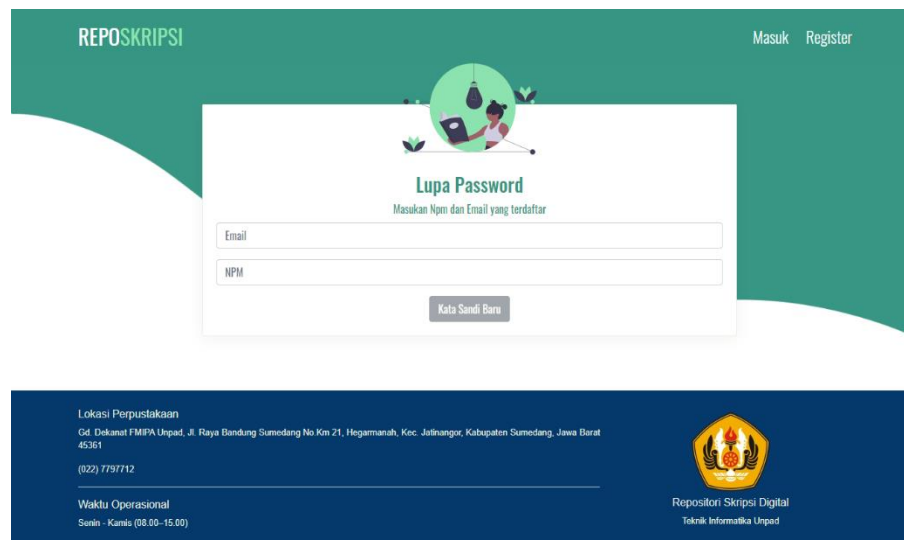
let token = bearer.split(' ')[1]
let payload = jwt.decode(token, secret).request
let {text} = req.body
let randomId = uuid().substring(0, 5)
let post = {
  id: payload.npm+randomId,
  npm: payload.npm,
  text: text,
  status: 0,
  sent_at: moment().format()
}
//menambahkan pesan ke tabel forums
let sql = 'INSERT INTO forums SET ?'
db.query(sql, post, (err, result)=>{
  if(err){
    console.log(err)
    return utils.template_response(res, 500, "Gagal mengirim" ,
null)
  }
  console.log('success!')
  return utils.template_response(res, 200, "Berhasil mengirim",
null)
})
})
//Menampilkan pesan
router.get('/forum', (req, res) =>{
  let bearer = req.headers.authorization
  let token = bearer.split(' ')[1]
  let payload = jwt.decode(token, secret).request
  let sql = `SELECT forums.npm, forums.text, forums.sent_at,
users.name FROM forums join users on users.npm =forums.npm WHERE
forums.id like '${payload.npm}%' order by forums.sent_at asc`
  db.query(sql, (err, result)=>{
    if (err) console.log(err)
    res.send(result)
  })
})
})

```

Tabel 4.10 *Black Box Testing* Kontak Admin

Skenario	Hasil yang diharapkan	Hasil
Mengirim pesan kosong	Data tidak akan dikirimkan.	Berhasil
Mengirim pesan berisi hanya spasi	Data tidak akan dikirimkan.	Berhasil
Mengirim pesan berisi text	Data dikirimkan dan ditampilkan di kolom pesan	Berhasil

2. Lupa Password



Gambar 4.21 Halaman Lupa Password

Fitur ini ditambahkan untuk memudahkan pengguna memulihkan akun ketika pengguna lupa *password* akunnya. Lupa *password* dapat diakses dibawah kolom *login*. Pengguna diminta untuk menginputkan *email* dan NPM yang terdaftar untuk melakukan permintaan *password* baru. Ketika klik permintaan kata sandi baru, pengguna dikirimkan *password* baru melalui *email*. Implementasi halaman lupa *password* dapat dilihat pada gambar 4.21. Fitur lupa *password* diuji dengan skenario *black box testing* pada tabel 4.11. Berikut adalah potongan kode fungsi lupa *password*:

```
router.put('/forgot-pass', (req, res) =>{
  let {npm, email} = req.body
  console.log(req.body)
  let findUser = `SELECT npm FROM users where role='user' AND
npm='${npm}' and email='${email}'`
  db.query(findUser, async(err, data)=>{
    try{
      console.log(data)
      if(data.length===0){
        return utils.template_response(res, 422, "Akun belum
terdaftar" , null)
      }
      let newPass = uuid().substring(0, 8)
```

```

    let enPass = await bcrypt.hash(newPass, 10)
    console.log(enPass)
    let sql = `update users set password='${enPass}' where
npm='${npm}' and email='${email}'`
    console.log(sql)
    db.query(sql, (err, result)=>{
    if (err) {
        console.log(err)
        return utils.template_response(res, 500, "Password baru gagal
dibuat", null)
    }
    console.log('Success')
    var helper = require('sendgrid').mail
    var from_email = new helper.Email('no-reply@repositori-
skripsi.com')
    var to_email = new helper.Email(email)
    var subject = 'Lupa Password'
    var emailText=`<html>
    <body>
        <p>Anda melakukan permintaan untuk mengubah password.
Berikut adalah password baru anda</p>
        <b style='text-align:center, color:#379683'>${newPass}</b>
        <p>Login menggunakan password diatas dan ganti password
anda</p>
    </body>
    </html>`
    var content = new helper.Content('text/html', emailText)
    var mail = new helper.Mail(from_email, subject, to_email,
content)
    var sg = require('sendgrid')(process.env.SENDGRID_API_KEY)
    var request = sg.emptyRequest({
        method: 'POST',
        path: '/v3/mail/send',
        body: mail.toJSON(),
    })
    sg.API(request, function(error, response) {
        console.log(response.statusCode)
        console.log(response.body)
        console.log(response.headers)
    })
    return utils.template_response(res, 200, 'Password berhasil
diganti' , null)
    })
}
catch{
    if (err) console.log(err.response)
    return utils.template_response(res, 500, "Password baru gagal
dibuat", null)
}
})
})

```

Skenario	Hasil yang diharapkan	Hasil
Tidak mengisi kolom NPM atau <i>email</i>	Tombol kata sandi baru tidak aktif.	Berhasil
Tidak mengisi NPM dengan 12 digit angka	Muncul pesan bahwa NPM tidak benar dan tombol kata sandi baru tidak aktif.	Berhasil
Menginputkan <i>email</i> atau NPM yang tidak terdaftar	Muncul pesan bahwa akun belum terdaftar dan untuk mengecek NPM dan <i>email</i> yang diinputkan setelah mengklik tombol kata sandi baru.	Berhasil
Menginputkan <i>email</i> dan NPM yang terdaftar	Muncul pesan untuk mengecek <i>email</i> . <i>Email</i> berisi <i>password</i> baru dikirimkan.	Berhasil
<i>Login</i> dengan <i>password</i> baru	<i>Login</i> berhasil dilakukan	Berhasil

Pada titik ini, penelitian memasuki fase akhir dan menghasilkan final produk yaitu web aplikasi yang telah diperbaharui. Final produk dari web repositori skripsi teknik Informatika Unpad terdiri dari fitur an fungsi sebagai berikut: register, *login*, unggah skripsi, verifikasi akun, tinjau skripsi, detail skripsi (pratinjau skripsi), pencarian dan penyaringan, *edit password*, cek status skripsi, fitur PWA yaitu tampilan pada saat *offline* dan fungsi *add to homescreen*, lupa *password*, dan kontak admin.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan berbagai hal yang telah disampaikan pada bab sebelumnya serta perancangan dan implementasi web repositori yang sudah dilakukan penulis, maka dapat diambil simpulan sebagai berikut:

1. Web repositori institusi ini mampu memudahkan mahasiswa Teknik Informatika Unpad untuk mengakses skripsi baik melalui *desktop* maupun *smartphone*.
2. Pengimplementasian metode *Extreme Programming* pada pengembangan web repositori skripsi Teknik Informatika Unpad dapat menghasilkan suatu produk dengan nilai *usability testing* sebesar 88.53 yang berarti web aplikasi berkualitas sangat baik. Hal ini dikarenakan XP berfokus pada permintaan pengguna, aplikasi terus mendapat *feedback* sehingga meningkatkan kualitas aplikasi secara berkala. Sifat kesederhanaan pada XP juga membuat pengembangan aplikasi dapat diwujudkan dalam waktu yang singkat karena berfokus pada fitur yang benar-benar dibutuhkan.
3. *Progressive Web Apps* yang diimplementasikan meningkatkan nilai pengalaman pengguna pada web karena web aplikasi dapat bertingkah layaknya aplikasi *native* pada perangkat yang digunakan tanpa perlu meng-*install* aplikasi dan tetap memiliki tampilan disaat *offline*. Pernyataan ini didukung dengan nilai *usability testing* sebesar 88.53.

5.2 Saran

Berdasarkan penelitian yang dilakukan, penulis memberikan saran yang dapat diimplementasikan pada penelitian selanjutnya, yaitu sebagai berikut:

1. Penelitian terhadap *extreme programming* selanjutnya diharapkan mengkaji lebih banyak sumber dan referensi sehingga memberikan hasil yang lebih baik lagi dari penelitian ini.
2. Penelitian selanjutnya diharap mengumpulkan lebih banyak responden dari penelitian ini sehingga *feedback* yang didapat terhadap penelitian lebih mendalam.
3. Web repositori skripsi dapat dikembangkan lagi dengan menambahkan lebih banyak materi digital yang dapat diakses seperti: *paper*, jurnal, dan publikasi lainnya.
4. Web repositori skripsi dapat mengimplementasikan fitur PWA lainnya agar dapat meningkatkan kualitas web aplikasi.

DAFTAR PUSTAKA

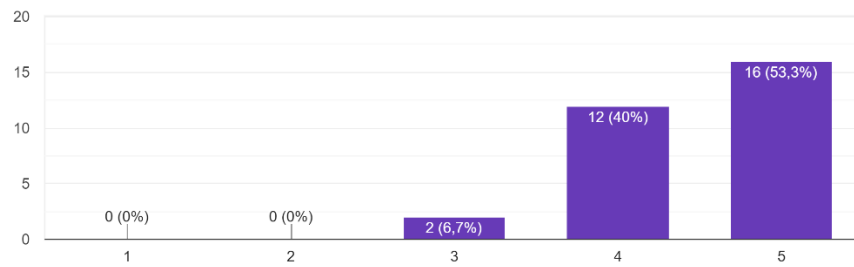
- A.S, R. dan Shalahuddin, M. (2018) *Rekayasa Perangkat Lunak*. Bandung: Informatika.
- Ater, T. (2017) *Building Progressive Web Apps: Bringing the Power of Native to the Browser*. O'Reilly Media, Inc.
- Ferguson, R. (2019) *Beginning JavaScript. The Ultimate Guide to Modern JavaScript Development*. 3rd edition. New Jersey: Apress.
- Freeman, A. (2019) *Pro React 16*. London: Apress.
- Hahn, E. M. (2016) *Express In Action. Writing, building, and testing Node.js applications*. New York: Manning Publications Co.
- Handiwidjojo, W. dan Ernawati², L. (2016) 'Pengukuran Tingkat Ketergunaan (Usability) Sistem Informasi Keuangan Studi Kasus: Duta Wacana Internal Transaction (Duwit)', *JUI SI*, 2(1).
- Hume, D. A. (2017) *Progressive Web Apps*. New York: Manning Publications.
- Jaya, T. S. (2018) 'Pengujian Aplikasi dengan Metode Blackbox Testing Boundary Value Analysis (Studi Kasus: Kantor Digital Politeknik Negeri Lampung)', *Jurnal Informatika: Jurnal Pengembangan IT (JPIT)*, 3(2).
- Karpagam, D. V. *et al.* (2017) 'Performance Enhancement of Webpage Using Progressive Web App Features', *International Journal of Innovative Research in Advanced Engineering (IJIRAE)*, 4(3).
- Krishna, T. S. R. *et al.* (2011) 'Survey on Extreme Programming in Software Engineering', in *International Journal of Computer Trends and Technology*.
- Maryuliana, *dkk.* (2016) 'Sistem Informasi Angket Pengukuran Skala Kebutuhan Materi Pembelajaran Tambahan Sebagai Pendukung Pengambilan Keputusan Di Sekolah Menengah Atas Menggunakan Skala Likert', *Jurnal Transistor Elektro dan Informatika (TRANSISTOR EI)*, 1(2).
- Mehta, C. *et al.* (2018) *MySQL 8 Administrator's Guide*. Birmingham Mumbai: Packt.
- Nadia, R. *dkk.* (2018) 'Rancang Bangun Aplikasi CallTenant dengan Penyimpanan

- Basis Data untuk Form Dinamis Menggunakan Framework Laravel', *JURNAL TEKNIK*, 7(1).
- Prabowo, S. A. *dkk* (2013) 'Rancang Bangun Aplikasi Web Inforasi Eksekutif pada Pemerintahan Kabupaten XYZ', *Jurnal Teknik POMITS*, 2, pp. A476–A480.
- Repanovici, A. (2009) 'Marketing Research about Attitudes, Difficulties and Interest of Academic Community about Institutional Repository', *Proceedings of the 3rd International Conference in Management, Marketing and Finances*, MMF'09, pp. 88–95.
- Ropianto, M. (2016) 'Pemahaman Penggunaan Unified Modelling Language', *JT-IBSI*, 01(01).
- Santoni, M. (2018) *Progressive Web Apps browser support & compatibility*. Available at: <https://www.goodbarber.com/blog/progressive-web-apps-browser-support-compatibility-a883/> (Accessed: 20 January 2019).
- Singh, H. and Bhatt, M. (2016) *Learning Web Development with React and Bootstrap*. Birmingham: Packt Publishing Ltd.
- Spillner, A. *et al.* (2014) *Software Testing Foundations*. 4th edn. Santa Barbara, CA: Rocky Nook Inc.
- Sutedjo, M. (2014) 'Pengelolaan Repositori Perguruan Tinggi dan Pengembangan Repositori Karya Seni', *Makalah Seminar Nasional "Digital Local Content: Strategi Membangun Repository Karya Seni"*, *GKU FSR ISI Yogyakarta*.

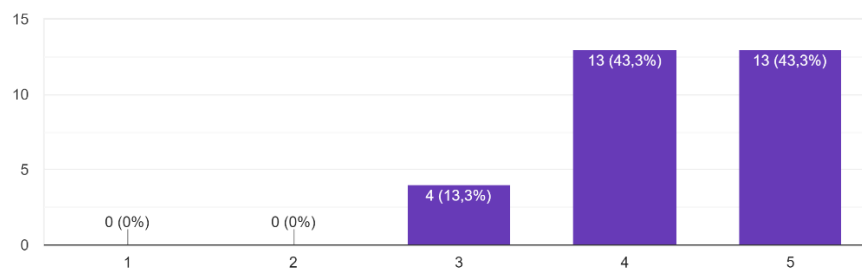
LAMPIRAN

Lampiran 1 Ringkasan Hasil *Usability Testing*

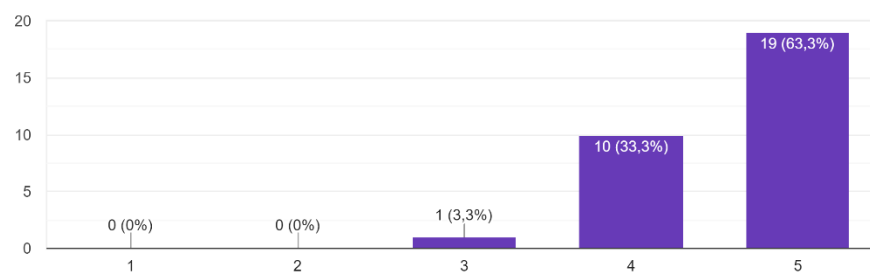
Bagaimana penilaian tampilan halaman utama
30 tanggapan



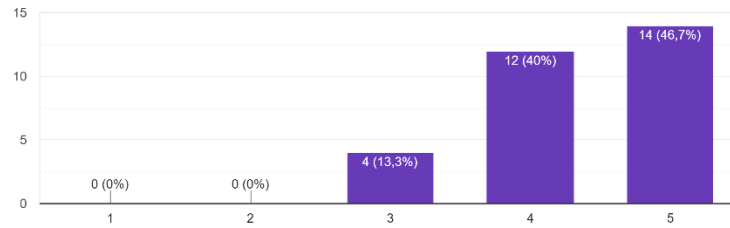
Bagaimana penilaian fungsi pencarian dan filter
30 tanggapan



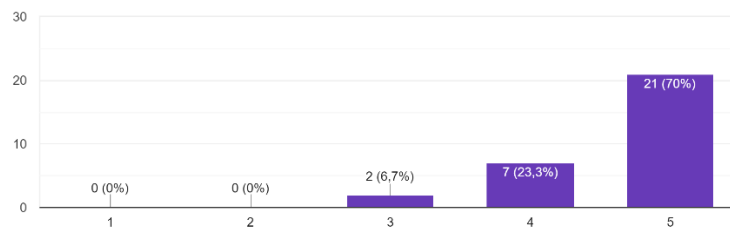
Bagaimana penilaian fungsi registrasi
30 tanggapan



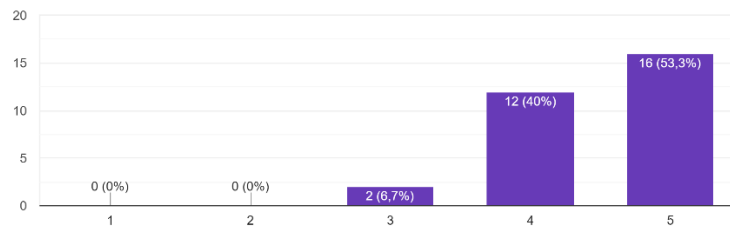
Bagaimana penilaian tampilan registrasi
30 tanggapan



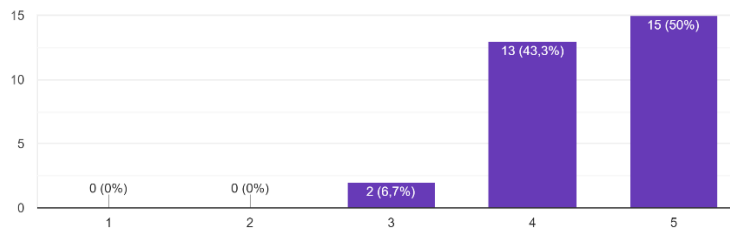
Bagaimana penilaian fungsi login
30 tanggapan



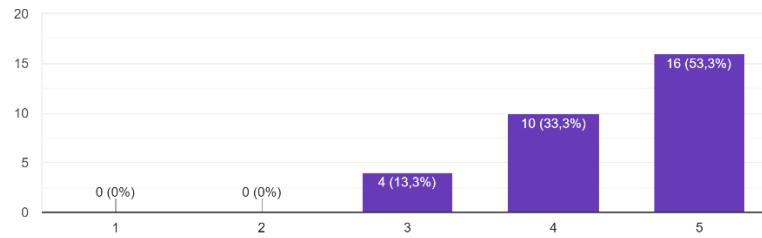
Bagaimana penilaian tampilan login
30 tanggapan



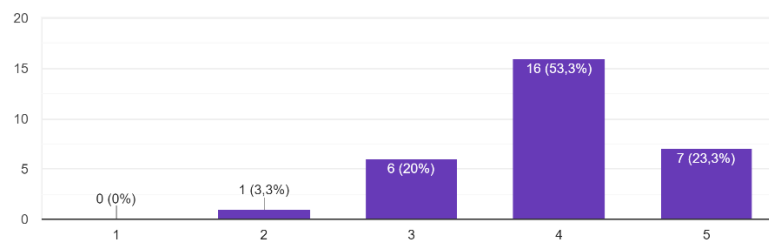
Bagaimana penilaian tampilan halaman detail skripsi (pratinjau skripsi)
30 tanggapan



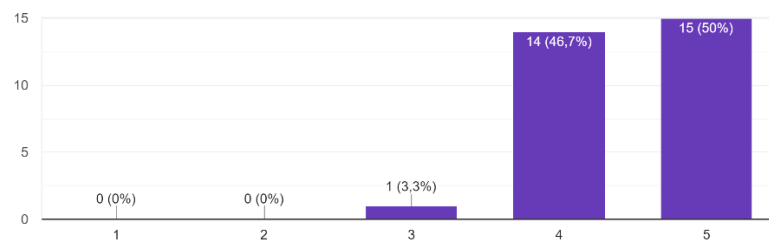
Bagaimana penilaian fungsi upload skripsi
30 tanggapan



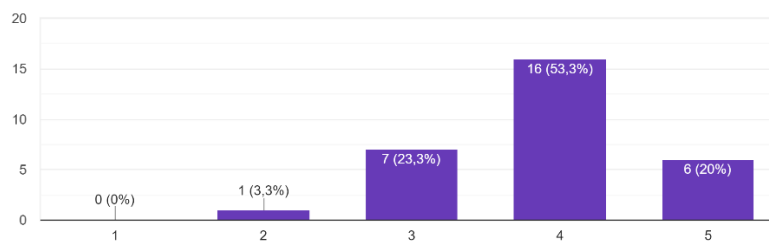
Bagaimana penilai fungsi edit unggahan
30 tanggapan



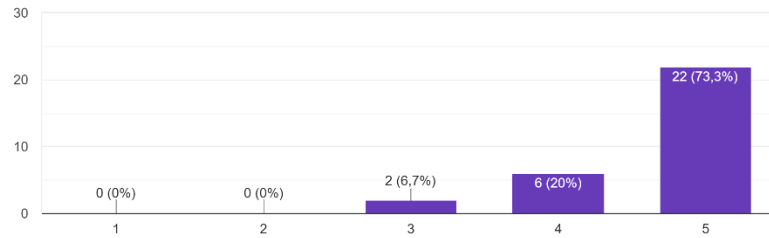
Bagaimana penilaian tampilan upload skripsi
30 tanggapan



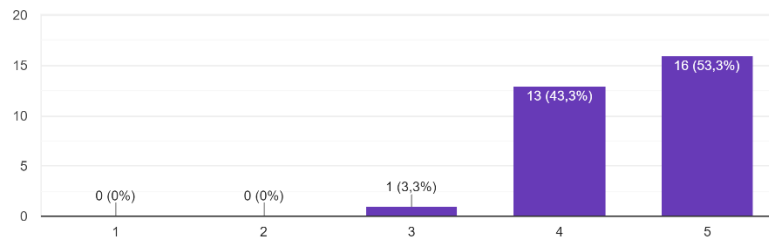
Bagaimana penilaian tampilan profil mahasiswa
30 tanggapan



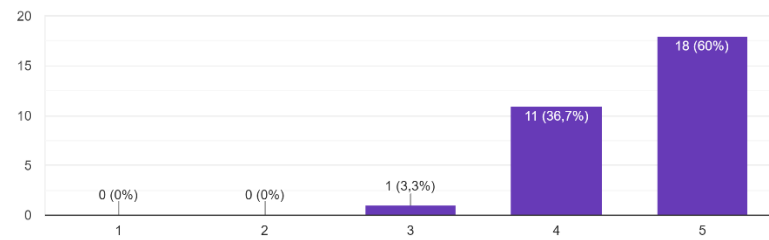
Bagaimana penilaian fungsi edit password
30 tanggapan



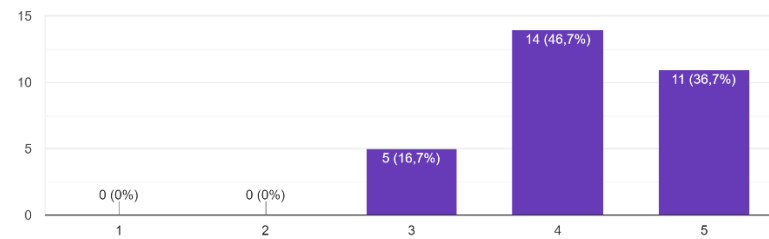
Bagaimana penilaian tampilan edit password
30 tanggapan



Bagaimana penilaian fungsi pada halaman status skripsi
30 tanggapan

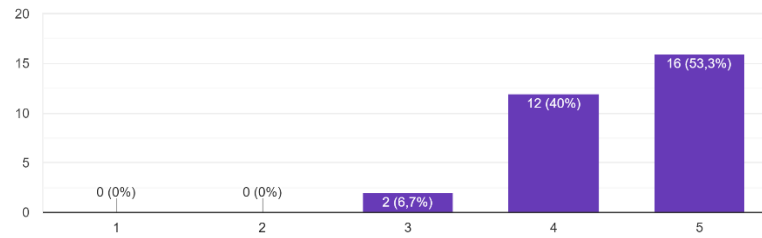


Bagaimana penilaian tampilan halaman status skripsi
30 tanggapan



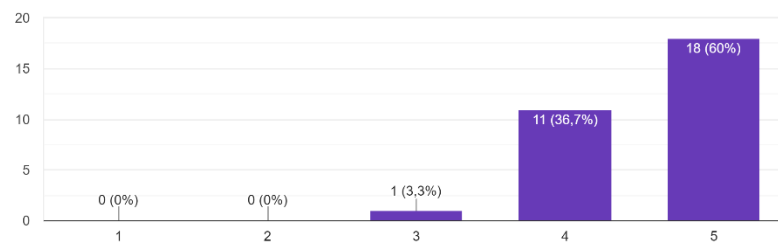
Bagaimana penilaian terhadap fungsi offline?

30 tanggapan



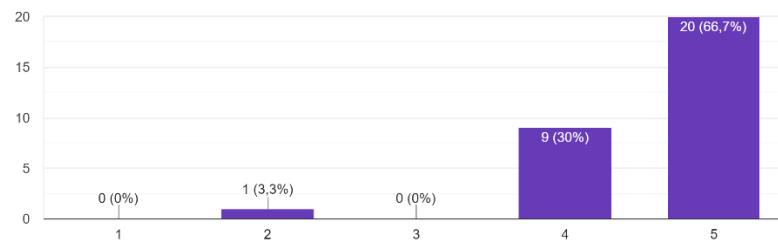
Bagaimana penilaian terhadap tampilan saat offline?

30 tanggapan



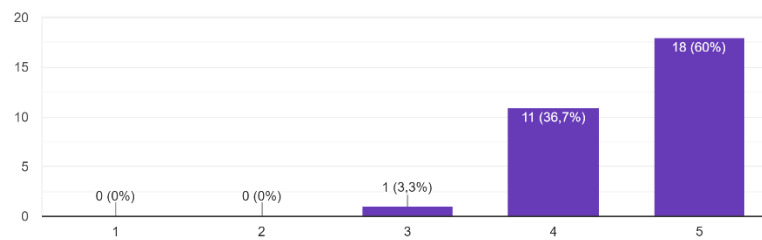
Bagaimana penilaian terhadap fungsi add to homescreen?

30 tanggapan



Apakah fitur add to homescreen membuat tampilan nyaman seperti native app? Berikan skala penilaian

30 tanggapan



Adakah feedback terhadap fitur-fitur web maupun web secara keseluruhan ? (terutama fitur yang diberi nilai < 3)

30 tanggapan

1. Untuk fitur pencarian dan filter, tombol search dirasa kurang berguna dikarenakan tanpa memencet tombol search pun, hasil pencarian sudah ditampilkan. Kata pa adit waktu semweb, kalo mau kayak gitu, mending tombolnya ilangin aja. Atau fungsinya diganti jadi harus mencet search dulu
2. Ketika register, ada dua pemberitahuan 'sedang diproses', bagusnya sih cukup 1 saja.
3. Status skripsi yang belum ditinjau, lebih baik diubah warnanya menjadi merah saja, daripada ukuran fontnya diperbesar tetapi warnanya sama, malah terlihat membuat tampilan menjadi tidak proporsional. Mending yang dibikin lebih gede fontnya mah yang jadi headernya, yang diatas nama itu.
4. Pada profile, lebih baik dikasih penjelasan bahwa itu tuh nama, pake yang kayak ' nama : ' gitu aja, biar lebih rapi dan enak dilihat. Kalo langsung datanya doang keliatannya kurang rapih dan asal simpen aja.
5. Pas upload skripsi, tahunnya kending pake dropdown aja karena pilihannya juga dikit. Dan biasanya kalo tahun tuh pake dropdown

edit ungahan seharusnya gak ngisi semua formnya dari awal

Tampilan profil terlihat terlalu seadanya. Yuk bisa yuk veg

Lampiran 2 Kode Web Aplikasi

Register

```
import React, { Component } from 'react'
import bg2 from '../icons/bg2.webp'
import axios from 'axios'
import { scrollTop } from '../helpers/autoScroll'
import { ProgressBar, Modal } from 'react-bootstrap'
import { Redirect } from 'react-router-dom'
import { connect } from 'react-redux'

export class Register extends Component {
  initialState = {
    showLoading: false,
    npm: '',
    email: '',
    pass: '',
    passCheck: '',
    message: '',
    status: '',
    displayForm1: 'block',
    displayForm2: 'none',
    displayForm3: 'none',
    file: null,
    progress: 34
  }
  state = this.initialState
  next = (e) => {
    e.preventDefault()
    this.setState({
      showLoading: true,
    })
    let { npm, pass, email } = this.state
    let name = this.refs.name.value
    let data = {
      name: name,
      email: email,
      npm: npm,
      password: pass
    }
    axios({
      method: 'POST',
      url: '/check-form',
      data: data
    }).then(res => {
      this.setState({
        message: '',
        displayForm1: 'none',
        displayForm2: 'block',
        progress: this.state.progress + 33,
        showLoading: false
      })
      scrollTop()
    }).catch(err => {
      this.setState({
        showLoading: false
      })
    })
  }
}
```

```

    })
    if (err.response) {
      this.setState({
        message: err.response.data.message,
        status: err.response.data.status,
      })
    }
    else{
      this.setState({
        message: 'Network error, Cek Koneksi Anda',
        status: 500,
      })
    }
  })
}
submitKTM = e => {
  e.preventDefault()
  this.setState({
    showLoading:true,
  })
  let { npm, pass, file,email } = this.state
  let name = this.refs.name.value
  const formData = new FormData()
  formData.append('ktm', file)
  formData.append('npm', npm)
  formData.append('email', email)
  formData.append('name', name)
  formData.append('password', pass)
  console.log(file)
  axios({
    method: 'POST',
    url: '/register',
    data: formData,
    headers: {
      'Content-Type': 'multipart/form-data'
    }
  }).then(res => {
    this.setState(this.initialState)
    this.setState({
      displayForm1: 'none',
      displayForm2: 'none',
      displayForm3: 'block',
      message: res.data.message,
      status: res.data.status,
      progress: 100
    })
    this.refs.registerForm.reset()
    scrollToTop()
  }).catch(err => {
    this.setState({
      showLoading:false
    })
    if (err.response) {
      this.setState({
        message: err.response.data.message,
        status: err.response.data.status,

```

```

    })
  }
  else{
    this.setState({
      message: 'Network error, Cek Koneksi anda',
      status: 500,
    })
  }
})
}
handleInput = e => {
  this.setState({
    [e.target.id] : e.target.value,
  })
}
handleRetype = (e) =>{
  let {pass} = this.state
  if (e.target.value !== pass ){
    this.setState({
      passCheck: false
    })
  } else {
    this.setState({
      passCheck: true
    })
  }
}
handleFile = e => {
  if (e.target.files[0]) {
    this.setState({
      file: e.target.files[0]
    })
  } else {
    this.setState({
      file: ''
    })
  }
}
back = e => {
  e.preventDefault()
  this.setState(this.initialState)
}
render() {
  let {npm, email, pass, passCheck, message, status,
  showLoading} =this.state
  if (this.props.token){
    return <Redirect to={'/'}/>
  }
  return (
    <>
      <img src={bg2} alt='Logo' className='bg2' />
      <div className='row no-margin'>
        <div className='col-xl-9 col-lg-12 register-box'>
          <h3>Register</h3>
          <form ref='registerForm' autoComplete='off'>
            <div>

```



```

        <ProgressBar now={this.state.progress} />
      </div>
      <fieldset style={{ display: this.state.displayForm1 }}>
        <div className='form-group'>
          <label>Nama</label>
          <input type='text' ref='name' className='form-control'
placeholder='Nama' />
        </div>
        <div className='form-group'>
          <label>Email</label>
          <input type='text' id='email' onBlur={this.handleInput}
className='form-control' placeholder='Email' />
        </div>
        {email.length>0 && !email.includes('@')}
        <div className='alert alert-warning' role='alert'>
          Mohon inputkan Email yang valid
        </div>
        :<></> }
        <div className='form-group'>
          <label>NPM</label>
          <input type='number' id='npm' onBlur={this.handleInput}
className='form-control' placeholder='NPM' />
        </div>
        {npm.length !== 12 && npm.length > 0 ?
        <div className='alert alert-warning' role='alert'>
          <strong>NPM salah! </strong>memerlukan 12 digit
        </div>
        : <></> }
        <div className='form-group'>
          <label>Password</label>
          <input type='password' onBlur={this.handleInput}
id='pass' className='form-control' placeholder='Password' />
        </div>
        <div className='form-group'>
          <label>Konfirmasi Password</label>
          <input type='password' onChange={this.handleRetype}
className='form-control' placeholder='Password' />
        </div>
        {passCheck === true || !pass ? ( <></> ) : (
        <div className='alert alert-warning' role='alert'>
          Password tidak cocok
        </div>
        )}
        <button type='next' className='btn btn-primary' onClick={e
=> this.next(e)} disabled={!npm || npm.length!==12 || !email ||
!email.includes('@') || !pass || !passCheck}>
          Lanjut
        </button>
        {message === '' ? ( <></> ) : (
        <div className='alert alert-danger' role='alert'>
          <strong>{this.state.message}</strong>
        </div>
        )}
      </fieldset>
      <fieldset style={{ display: this.state.displayForm2 }}>
        <div className='form-group'>

```

```

        <label>Foto KTM (Maks 5Mb)</label>
        <input type='file' id='ktm' onChange={this.handleFile}
className='form-control-file' accept='.png, .jpg, .jpeg' />
      </div>
      <button type='submit' className='btn btn-primary'
onClick={e => this.submitKTM(e)}>
        Submit
      </button>
      {message === '' ? (<></>) : (
        <div className='alert alert-danger' role='alert'>
          <strong>{this.state.message}</strong>
        </div>
      )}
    </fieldset>
    <fieldset style={{ display: this.state.displayForm3}}>
      {status === 200 ? (
        <div className="text-center register-text"
style={{marginBottom: '20px'}}>
          <p><b>{this.state.message}</b></p>
          <label>Silahkan cek email anda untuk mendapatkan link
verifikasi</label>
          <label>Link akan aktif selama 30 menit kedepan</label>
        </div>
      ) : (<></>)}
      <button className='btn btn-primary' onClick={e =>
this.back(e)}>
        Kembali
      </button>
    </fieldset>
  </form>
</div>
<div className='col-xl-3'></div>
<Modal show={showLoading} centered>
  <Modal.Body className='modal-box'>
    Sedang diproses...
  </Modal.Body>
</Modal>
</div>
</>
)
}
}
const mapStateToProps = state => {
  return{
    token: state.auth.token
  }
}
export default connect(mapStateToProps, null)(Register)

//backend
const express = require('express')
const router = express.Router()
const bcrypt = require('bcrypt')
const uuid = require('uuid/v4')
const utils = require('../utils/templates')
const asyncHandler = require('express-async-handler')

```

```

const moment = require('moment')

//connect DB
const db = require('../db/db')
require('../db/connection')
//Multer : Handle Uploaded Files
const multer = require('multer')
// Set The Storage Engine
const storage = multer.diskStorage({
  destination: 'files/ktm/',
  filename: function(req, file, cb){
    cb(null, Date.now() + file.originalname)
  }
})
//Check Image type
const fileFilter = (req, file, cb) => {
  if (file.mimetype === 'image/jpeg' || file.mimetype ===
'image/png' ) {
    cb(null, true)
  } else {
    let err={
      message:'File harus jpeg, jpg, or png'
    }
    cb(err, false)
  }
}
//Init Upload
const upload = multer({
  storage: storage,
  limits:{fileSize: 1024 * 1024 * 5},
  fileFilter:fileFilter
}).single('ktm')

//Register
router.post('/register', (req, res) =>{
  upload(req, res, asyncHandler(async(err) => {
    let {name, npm, password, email } = req.body
    console.log(req.file)
    console.log(req.body)
    console.log(moment().add(30, 'minutes').format())
    if(err){
      return utils.template_response(res, 500, err.message , null)
    }
    if (!req.file){
      return utils.template_response(res, 500, 'File tidak boleh
kosong' , null)
    }
    let encryptPassword = await bcrypt.hash(password, 10)
    let expired=moment().add(30, 'minutes').format()
    let data = {
      name: name,
      email:email,
      npm: npm,
      password: encryptPassword,
      ktm_url:req.file.path,
      created_at:moment().format(),

```

```

    token: uuid(),
    token_expired:expired
  }
  console.log(data)
  let sql = 'INSERT INTO temp_users SET ?'
  db.query(sql, data, (err, result)=>{
    if (err){
      console.log('Failed',err)
      return utils.template_response(res, 400, "Gagal register",
null)
    }
    console.log('Success')
    var helper = require('sendgrid').mail;
    var from_email = new helper.Email('no-reply@repositori-
skripsi.com');
    var to_email = new helper.Email(email);
    var subject = 'Verifikasi Email!';
    var emailText=`<html>
      <body>
        <p>Halo ${name},</p>
        <p>Anda telah melakukan registrasi akun pada web aplikasi
repositori-skripsi</p>
        <p>Mohon verifikasi email anda dengan menklik link berikut.
link akan aktif selama 30 sejak email dikirim</p>
        <a href=${'https://repositori-skripsi.herokuapp.com/email-
verification/'+data.token}>Verifikasi Email!</a>
      </body>
    </html>`
    var content = new helper.Content('text/html', emailText);
    var mail = new helper.Mail(from_email, subject, to_email,
content);
    var sg = require('sendgrid')(process.env.SENDGRID_API_KEY);
    var request = sg.emptyRequest({
      method: 'POST',
      path: '/v3/mail/send',
      body: mail.toJSON(),
    });
    sg.API(request, function(error, response) {
      console.log(response.statusCode);
      console.log(response.body);
      console.log(response.headers);
    });
    return utils.template_response(res, 200, "Email verifikasi
telah dikirim", null)
  })
}))
})

router.post('/check-form', (req, res) =>{
  let { name, npm, password, email } = req.body
  //Check Fields
  if (!name || !npm || !password || !email) {
    return utils.template_response(res, 400, "Semua field harus
diisi" , null)
  }
  //Check min NPM

```

```

    if(npm.length<12 || npm.length>=15 ) {
      return utils.template_response(res, 422, "NPM salah.
Memerlukan 12 digit" , null)
    }
    //check email
    if(!email.includes('@') || !email.includes('.') ) {
      return utils.template_response(res, 422, "Email tidak valid" ,
null)
    }
    //Check if npm is already registered
    let findUser = `SELECT npm FROM users where role='user' AND
npm='${npm}' or email='${email}'`
    db.query(findUser, (err, data)=>{
      if (err) console.log(err.response)
      console.log(data)
      if(data.length>0){
        return utils.template_response(res, 422, "NPM atau Email
sudah pernah didaftarkan" , null)
      }
      console.log('Data is valid')
      return utils.template_response(res, 200, "Data valid", null)
    })
  })

router.get('/verify-email/', (req, res) =>{
  let { id } = req.query
  let data_temp=[]
  let now = moment().format()
  //cek token
  let findToken = `SELECT * FROM temp_users where token='${id}'
&& token_expired>'${now}'`
  db.query(findToken, (err, data)=>{
    if (err){
      console.log(err)
      return
    }
    if(data.length===0){
      console.log('kosong')
      return utils.template_response(res, 422, "Token tidak valid"
, null)
    }
    data_temp = data[0]
    console.log(data_temp)
    console.log('Token valid')
    //cek User
    let findUser = `SELECT * FROM users where
npm='${data_temp.npm}' or email='${data_temp.email}'`
    db.query(findUser, (err, user)=>{
      if (err) {
        console.log(err)
        return
      }
      if(user.length>0){
        return utils.template_response(res, 200, "Akun sudah pernah
didaftarkan" , null)
      }
    })
  })
})

```

```

//insert ke tabel users
let insertSql = `INSERT INTO users (name, npm, ktm_url,
password, email, created_at)
SELECT name, npm, ktm_url, password, email, created_at
FROM temp_users WHERE token='${data_temp.token}'`
db.query(insertSql, (err, result)=>{
  if (err) {
    console.log('err', err)
    return
  }
  //delete dari table temp
  let delSql = `delete from temp_users where
token='${data_temp.token}'`
  db.query(delSql, (err, del)=>{
    if (err) {
      console.log(err)
      return
    }
    console.log('sukses')
    return utils.template_response(res, 200, "Register
Berhasil" , null)
  })
})
})
})
})
module.exports = router

```

Login

```

import React, { PureComponent } from 'react'
import UserMenu from '../components/UserMenu'
import AdminMenu from '../components/AdminMenu'
import { Link, Redirect } from 'react-router-dom'
import { setToken, delToken } from '../reducers/authReducer'
import { connect } from 'react-redux'
import { Modal } from 'react-bootstrap'
import { FaRegCheckCircle } from 'react-icons/fa'
import '../styles/nav.css'
import axios from 'axios'
import { scrollToTop } from '../helpers/autoScroll'
import MediaQuery from 'react-responsive'

export class Nav extends PureComponent {
  state = {
    npm: '',
    pass: '',
    message: '',
    status: null,
    showLogin: false,
    showLoading: false,
    justLoggedIn: false, //ketika pertama kali login, agar refresh
    tidak redirect
  }
  handleInput = (e) =>{
    this.setState({
      status: '',

```

```

    [e.target.id] : e.target.value
  })
}
submitLogin = e => {
  e.preventDefault()
  this.setState({
    showLoading:true
  })
  axios({
    method: 'post',
    url: '/login',
    data: {
      npm: this.state.npm,
      password: this.state.pass
    }
  }).then(res => {
    let loginInfo = res.data.data
    console.log(loginInfo)
    if (loginInfo.isLogged){ //response didapat
      this.props.login(loginInfo) //set state global
      this.setState({ //show modal
        showLoading:false,
        status:res.data.status,
        justLoggedIn:true,
        showLogin:true,
      })
      scrolllToTop()
      setTimeout(() =>
        this.setState({ //hide modal
          showLogin:false
        }), 1000)
    }
    else{
      this.setState({
        showLoading:false,
        status:500
      })
    }
  }).catch((err) => {
    if(err.response){
      this.setState({
        message:err.response.data.message,
        status:err.response.data.status,
        showLoading:false,
      })
    } else{
      this.setState({
        status: 500,
        showLoading:false,
      })
    }
  })
}
logout = () =>{
  this.props.logout()
}

```

```

handleClose = () => {
  this.setState({
    showLogin:false
  })
}
render() {
  let { message, status, justLoggedIn } = this.state
  let { token, role } = this.props
  return (
    <>
      <nav className='navbar navbar-expand-md sticky-top navbar-dark'>
        <div data-toggle="collapse" data-target=".navbar-collapse.show">
          <Link to="/" className='navbar-brand'>
            <p>
              REPO<span>SKRIPSI</span>
            </p>
          </Link>
        </div>
        {!token ?
          <>
            { /*Toggler*/ }
            <button className='navbar-toggler' type='button' data-toggle='collapse' data-target='#toggle1'>
              <span className='navbar-toggler-icon'></span>
            </button>
            { /*Collapse Items*/ }
            <div className='collapse navbar-collapse' id='toggle1'>
              <ul className='navbar-nav'>
                <li className='nav-item dropdown'>
                  <button className='btn btn-nav btn-transition dropdown' data-toggle='dropdown'>Masuk</button>
                  <ul className='dropdown-menu login-form'>
                    <form className='form'>
                      <div className="padding-15">
                        <div className="row">
                          <div className="col-6 col-md-5 no-padding">
                            <input type='text' id='npm' className='form-control' placeholder='NPM' onChange={this.handleInput} required/>
                          </div>
                          <div className="col-6 col-md-5 no-padding">
                            <input type='password' id='pass' className='form-control' placeholder='Password' onChange={this.handleInput} required/>
                          </div>
                          <div className="col-12 col-md-2 no-padding">
                            <button type='submit' className='btn btn-primary' onClick={e => this.submitLogin(e)}>
                              Masuk
                            </button>
                          </div>
                        </div>
                      </div>
                    </form>
                  </ul>
                </li>
              </ul>
            </div>
          </>
        }
      </nav>
    </>
  )
}

```



```

        <Link to="/forgot" className='nav-forgot'>Lupa
        Password</Link>
      </div>
    </div>
    {status===400?
    <div className='alert alert-warning login-alert'
    role='alert'>
      <strong>{message}</strong>
    </div>
    : status===500 ?
    <div className='alert alert-danger login-alert'
    role='alert'>
      <strong>Terjadi Kesalahan!</strong>Periksa koneksi
      anda dan coba lagi
    </div>
    : <></> }
    </form>
  </ul>
</li>
<li className='nav-item'>
  <Link to='/register' className='btn btn-nav btn-
  transition'>
    Register
  </Link>
</li>
</ul>
</div>
</>
:role === 'admin' ?
<>
  {/* justLoggedIn agar redirect ketika login tapi tidak
  ketika refresh */}
  {justLoggedIn? <Redirect to={'/admin'} /> :<></>}
  <AdminMenu logout={this.logout}></AdminMenu>
</>
:
<>
  {justLoggedIn? <Redirect to='/' /> : <></>}
  <MediaQuery query='(min-device-width:768px)'>
    <UserMenu logout={this.logout}></UserMenu>
  </MediaQuery>
  <MediaQuery query='(max-device-width:767px)'>
    <ul className='navbar-nav'>
      <li className='nav-item right'>
        <Link to='/' className='btn btn-nav btn-transition'
        onClick={()=>this.logout()}>
          Log out
        </Link>
      </li>
    </ul>
  </MediaQuery>
</>
}
<Modal show={this.state.showLogin}
onHide={this.handleClose} centered>
  <Modal.Body className='modal-box'>

```

```

        <div className='icon-check'><FaRegCheckCircle/></div>
        Log In Berhasil
      </Modal.Body>
    </Modal>
    <Modal show={this.state.showLoading} centered>
      <Modal.Body className='modal-box'>
        Sedang diproses ...
      </Modal.Body>
    </Modal>
  </nav>
</>
)
}
}
const mapDispatchToProps = dispatch => {
  return {
    login: (loginInfo) => dispatch(setToken(loginInfo)),
    logout: () => dispatch(delToken())
  }
}
const mapStateToProps = state => {
  return{
    token: state.auth.token,
    role: state.auth.role
  }
}
export default connect(mapStateToProps, mapDispatchToProps)(Nav)

```

Unggah

```

import React, { Component } from 'react'
import { connect } from 'react-redux'
import { Redirect, Link } from 'react-router-dom'
import axios from 'axios'
import { Spinner, Modal } from 'react-bootstrap'

export class Upload extends Component {
  initialState={
    skripsi:{},
    isLoading:false,
    offline: false,
    showLoading:false,
    title:'',
    titleAlert:'initial',
    year:'',
    yearAlert:'initial',
    abstract:'',
    abstractAlert:'initial',
    abstrak:'',
    abstrakAlert:'initial',
    category:'',
    keywords:'',
    file:null,
    message: '',
    status:'',
  }
  state=this.initialState

```

```

submit = (e) =>{
  e.preventDefault()
  this.setState({
    showLoading:true
  })
  let {title, year, abstrak, abstract, category, keywords} =
this.state
  let {file} = this.state
  const formData = new FormData()
  formData.append('file', file)
  formData.append('title', title)
  formData.append('year', year)
  formData.append('abstrak', abstrak)
  formData.append('abstract', abstract)
  formData.append('category', category)
  formData.append('keywords', keywords)
  axios({
    method: 'POST',
    url: `/user/upload/`,
    data: formData,
    headers:{
      'Content-Type':'multipart/form-data',
      'Authorization': this.props.token
    }
  }).then((res) =>{
    this.refs.uploadForm.reset()
    this.setState({
      message:res.data.message,
      status:res.data.status,
      showLoading:false
    })
  }).catch((err) => {
    this.setState({
      showLoading:false
    })
    if( err.response){
      this.setState({
        message:err.response.data.message,
        status:err.response.data.status,
      })
    }
  })
}
handleInput = (e) =>{
  if(e.target.id==='title'){
    e.target.value=e.target.value.replace(/\n/g, ' ')
  }
  this.setState({
    [e.target.id] : e.target.value,
  })
  if (e.target.id==='title'){
    this.setState({
      titleAlert: e.target.value,
    })
  }
  else if (e.target.id==='year'){

```

```

        this.setState({
          yearAlert: e.target.value,
        })
      }
      else if (e.target.id==='abstract'){
        this.setState({
          abstractAlert: e.target.value,
        })
      }
      else if (e.target.id==='abstrak'){
        this.setState({
          abstrakAlert: e.target.value,
        })
      }
    }
  }
  handleFile=(e)=>{
    this.setState({
      file:e.target.files[0],
    })
  }
  checkSkripsi=()=>{
    axios({
      method: 'get',
      url: `/user/skripsi/`,
      headers: {
        Authorization: this.props.token
      }
    }).then(res=>{
      this.setState({
        skripsi: res.data,
        isLoading: true
      })
    }).catch((err) => {
      if(err.response){
        console.log(err.response)
      }
    })
  }
  componentDidMount(){
    if (navigator.onLine){
      this.checkSkripsi()
      this.setState({
        offline:false
      })
    }
    else{
      this.setState({
        offline:true,
      })
    }
  }
  render() {
    let { message, status, isLoading, offline, skripsi, file,
      title, year, abstrak, abstract, titleAlert, yearAlert,
      abstrakAlert, abstractAlert, keywords} = this.state
    if (!this.props.token || this.props.role==='admin'){

```

```

    return <Redirect to={'/' } />
  }
  return (
    <>
    <div className='row no-margin'>
      <div className='upload-box'>
        <h3>Unggah</h3>
        {offline? <p className='text-center'> Anda sedang offline.
Cek koneksi anda dan refresh </p>
          : !isLoading? <div className='spin-box middle'><Spinner
animation='border' variant='secondary' /></div>
          : skripsi ? <><hr/><div className='msg-upload'><h5>Anda
sudah mengunggah skripsi</h5><p>Cek status skripsi di menu
profil</p></div></>
        :<>
          <form ref='uploadForm'>
            {status === 200?
              <>
                <div className='alert alert-success' role='alert'>
                  <strong>{this.state.message}</strong>
                </div>
                <Link to='/'><button className='btn btn-
primary'>Selesai</button></Link>
              </> :
              <>
                <div className='form-group'>
                  <label>Judul *</label>
                  <textarea type='text' id='title' maxLength='255'
onBlur={this.handleInput} className='form-control'
placeholder='Judul Skripsi' />
                  {titleAlert==='initial'? <></> : !titleAlert ?
                    <div className='alert alert-danger' role='alert'>
                      <strong>Judul tidak boleh kosong</strong>
                    </div> : <></> }
                </div>
                <div className='form-group'>
                  <label>Tahun *</label>
                  <input type='number' id='year' onBlur={this.handleInput}
className='form-control' placeholder='Tahun Publikasi Skripsi' />
                  {yearAlert==='initial'? <></> : year.length!==4 ||
year<2000 || year>2100 ?
                    <div className='alert alert-danger' role='alert'>
                      <strong>Tahun harus diisi dengan benar</strong>
                    </div> : <></> }
                </div>
                <div className='form-group'>
                  <label>Abstrak *</label>
                  <textarea id='abstrak' onBlur={this.handleInput}
className='form-control' placeholder='Input Abstrak' />
                  {abstrakAlert==='initial'? <></> : !abstrakAlert ?
                    <div className='alert alert-danger' role='alert'>
                      <strong>Abstrak tidak boleh kosong</strong>
                    </div> : <></> }
                </div>
                <div className='form-group'>
                  <label>Abstract *</label>

```

```

        <textarea id='abstract' onBlur={this.handleInput}
className='form-control' placeholder='Input Abstract' />
        {abstractAlert=== 'initial'? <></> : !abstractAlert ?
        <div className='alert alert-danger' role='alert'>
            <strong>Abstrak tidak boleh kosong</strong>
        </div> : <></> }
    </div>
    <div className='form-group'>
        <label>File * (Pdf Maks 20mb)</label>
        <input type='file' ref='file' onChange={this.handleFile}
className='form-control-file' id='file' accept='.pdf' />
        {!file ? <></> : file.type==='application/pdf' ? <></> :
        <div className='alert alert-danger' role='alert'>
            <strong>File must be PDF</strong>
        </div>
        }
    </div>
    <div className='form-group'>
        <label>Bidang Minat Skripsi</label>
        <select className='custom-select'
onChange={this.handleInput} id='category'>
            <option value='1'>Sistem Cerdas dan Sistem Grafika
(SCSG)</option>
            <option value='2'>Sistem Informasi dan Rekayasa
Perangkat Lunak (SIRPL)</option>
            <option value='3'>Jaringan Komputer dan Komunikasi Data
(JKKD)</option>
            <option value='4'>Ilmu Komputasi dan Metode Numerik
(IKMN)</option>
        </select>
    </div>
    <div className='form-group'>
        <label>Kata Kunci </label>
        <p style={{fontSize: '0.8rem'}}>Input 1-5 kata kunci yang
berkaitan dengan skripsi (pisahkan dengan koma)</p>
        <input type='text' id='keywords'
onChange={this.handleInput} className='form-control'
placeholder='Kata Kunci' />
        {keywords.length<255 ? <></> :
        <div className='alert alert-danger' role='alert'>
            <strong>Kata kunci terlalu banyak</strong>
        </div>
        }
    </div>
    <button type='submit' className='btn btn-primary'
onClick={ (e)=>this.submit(e) } disabled={!title || !abstract ||
!abstrak || year.length!==4 || year<2000 || year>2100 || !file
|| keywords.length>=255}>Submit</button>
    {message === '' ? <></> :
    <div className='alert alert-danger' role='alert'>
        <strong>{this.state.message}</strong>
    </div>
    }
</>
}
</form>

```

```

        <Modal show={this.state.showLoading} centered>
          <Modal.Body className='modal-box'>
            Sedang diproses ...
          </Modal.Body>
        </Modal>
      </>
    }
  </div>
</div>
</>
)
}
}
const mapStateToProps = state => {
  return{
    token : state.auth.token,
    role: state.auth.role
  }
}
export default connect(mapStateToProps, null)(Upload)

```

Detail Skripsi

```

import React, { Component } from 'react'
import { Spinner } from 'react-bootstrap'
import axios from 'axios'
import { connect } from 'react-redux'
import { FaChevronLeft, FaChevronRight } from 'react-icons/fa'
import { Document, pdfjs, Page } from 'react-pdf'
pdfjs.GlobalWorkerOptions.workerSrc =
`//cdnjs.cloudflare.com/ajax/libs/pdf.js/${pdfjs.version}/pdf.worker.js`

export class SkripsiDetail extends Component {
  state={
    skripsi:[],
    isLoading:false,
    offline:false,
    pageNumber:1,
    numPages:null
  }
  getData =()=>{
    let id = this.props.match.params.id
    if(this.props.token){
      axios({
        method: 'get',
        url: `/skripsi/detail/${id}`,
        params:{
          id : id
        },
        headers: {
          Authorization: this.props.token
        }
      }).then(res=>{
        this.setState({
          skripsi: res.data[0],
          isLoading: true

```

```

    })
  }).catch(err=>{
    if(err.response){
      console.log(err.response)
    }
  })
}
else{
  axios({
    method: 'get',
    url: `/skripsi/info/`,
    params:{
      id : id
    }
  }).then(res=>{
    this.setState({
      skripsi: res.data[0],
      isLoading: true
    })
  }).catch(err=>{
    if(err.response){
      console.log(err.response)
    }
  })
}
}
onDocumentLoadSuccess = ({ numPages }) => {
  this.setState({ numPages });
}
next = () => {
  let {pageNumber, numPages} = this.state
  if( pageNumber<=numPages){
    this.setState({pageNumber:this.state.pageNumber+1})
  }
}
before = () => {
  let {pageNumber} = this.state
  if( pageNumber>1){
    this.setState({pageNumber:this.state.pageNumber-1})
  }
}
componentDidMount(){
  if (navigator.onLine){
    this.getData()
    this.setState({
      offline:false
    })
  }
  else{
    this.setState({
      offline:true,
    })
  }
}
render() {

```



```

    let { isLoading, skripsi, offline, pageNumber, numPages } =
this.state
    return (
      <div className="main-box">
        { offline ?
          <div className="row">
            <div className="col-12">
              <div className="line"></div>
              <div className="offline-box">
                <p>Anda sedang offline. Cek koneksi anda dan refresh </p>
              </div>
            </div>
          </div>
          :
          <>
          <div className="row">
            <div className="col-12 col-md-4">
              <div className="small-box">
                <div className="line"></div>
                <div className="detail-box">
                  { !isLoading ? <Spinner animation="border"
variant="secondary" /> :
                  <>
                    { !skripsi ? <>No Data</> :
                    <>
                      <div className="detail-header">
                        <h4>IDENTITAS</h4>
                        <hr/>
                      </div>
                      <h5>Judul</h5>
                      <p>{skripsi.title}</p>
                      <h5>Penulis</h5>
                      <p>{skripsi.name}</p>
                      <h5>Tahun</h5>
                      <p>{skripsi.published_year}</p>
                      <h5>Kategori</h5>
                      <p>{skripsi.category===1 ? <>Sistem Cerdas dan Sistem
Grafika (SCSG)</> :
                        skripsi.category===2 ? <>Sistem Informasi dan Rekayasa
Perangkat Lunak (SIRPL)</> :
                        skripsi.category===3 ? <>Jaringan Komputer dan Komunikasi
Data (JKKD)</> :
                        skripsi.category===4 ? <>Ilmu Komputasi dan Metode Numerik
(IKMN)</> : <></> }</p>
                      <h5>Kata Kunci</h5>
                      <p>{skripsi.keywords ? skripsi.keywords : <>-</>}</p>
                    </>
                  </div>
                </div>
              </div>
            </div>
            <div className="col-12 col-md-8 abstract">
              <div className="small-box">

```

```

    <div className="line" style={{ backgroundColor: '#8ee4af
'}}></div>
    <div className="detail-box">
      {!isLoading ? <Spinner animation="border"
variant="secondary" /> :
      <>
        {!skripsi ? <>No Data</> :
        <>
          <div className="detail-header">
            <h4>ABSTRAK</h4>
            <hr/>
          </div>
          <p>{skripsi.abstrak}</p>
          <hr/>
          <p>{skripsi.abstract}</p>
          </>
        </>
      </div>
    </div>
    </div>
    </div>
    {!this.props.token? <></> :
    <div className="row file">
      <div className="col-12">
        <div className="small-box">
          <div className="line"
style={{backgroundColor: '#5cdb95'}}></div>
          <div className="detail-box">
            <h5>FILE</h5>
            <hr/>
            {!isLoading ? <Spinner animation="border"
variant="secondary" /> :<>
            {!skripsi ? <>No Data</> :
            <>
              <Document file={'https://repositori-
skripsi.herokuapp.com/'+skripsi.file_url}
onLoadSuccess={this.onDocumentLoadSuccess}>
                <Page pageNumber={pageNumber} />
              </Document>
              <div className="btn-pdf-box">
                <button className="btn btn-primary btn-pdf"
disabled={pageNumber===1? true: false}
onClick={()=>this.before()}><FaChevronLeft/></button>
                <p className="no-margin num-page">Page {pageNumber} of
{numPages}</p>
                <button className="btn btn-primary btn-pdf"
disabled={pageNumber===18? true: false}
onClick={()=>this.next()}><FaChevronRight/></button>
              </div>
            </>
          </div>
        </div>
      </div>
    </div>
  </div>

```

```

        </div>
      </div>
    }
  </>
}
</div>
)
}
}
const mapStateToProps = state => {
  return{
    token : state.auth.token,
  }
}
export default connect(mapStateToProps, null)(SkripsiDetail)

```

Profil dan Edit *Password*

```

import React, { Component } from 'react'
import { Spinner, Modal } from 'react-bootstrap'
import { connect } from 'react-redux'
import axios from 'axios'
import moment from 'moment'

export class ProfileInfo extends Component {
  state={
    user:{},
    isLoading:false,
    showLoading:false,
    offline:false,
    newPass:'',
    oldPass:'',
    message:'',
    status:null,
    passCheck:''
  }
  getProfile= ()=>{
    axios({
      method: 'get',
      url: `/user/profile/`,
      headers: {
        Authorization: this.props.token
      }
    }).then(res=>{
      this.setState({
        user: res.data,
        isLoading: true
      })
    }).catch(err=>{
      console.log(err.response)
    })
  }
  componentDidMount(){
    if (navigator.onLine){
      this.getProfile()
      this.setState({
        offline:false
      })
    }
  }
}

```

```

    })
  }
  else{
    this.setState({
      offline:true,
    })
  }
}
submit=(e)=>{
  e.preventDefault()
  this.setState({
    showLoading:true
  })
  let { newPass, oldPass } = this.state
  axios({
    method: 'put',
    url: `/user/edit-pass`,
    headers:{
      Authorization: this.props.token
    },
    data: {
      newPass:newPass,
      oldPass:oldPass
    }
  }).then(res => {
    this.refs.editForm.reset();
    this.setState({
      newPass:'',
      oldPass:'',
      message:res.data.message,
      status:res.data.status,
      showLoading:false
    })
  }).catch((err) => {
    this.setState({
      showLoading:false
    })
    if(err.response){
      this.setState({
        message:err.response.data.message,
        status:err.response.data.status,
      })
    }
  })
}
handleInput = (e) =>{
  this.setState({
    [e.target.id]: e.target.value,
    Status:null
  })
  if(e.target.id==='newPass' && this.refs.confirmPass.value){
    if(e.target.value!==this.refs.confirmPass.value){
      this.setState({passCheck: false})
    }
    else{
      this.setState({passCheck: true})
    }
  }
}

```

```

    }
  }
}
handleRetype = (e) =>{
  if (e.target.value !== this.state.newPass){
    this.setState({passCheck: false})
  }
  else{
    this.setState({passCheck: true})
  }
}
clear = (e) =>{
  this.refs.editForm.reset()
  this.setState({
    newPass:'',
    oldPass:'',
    message:'',
    status:null,
    passCheck:''
  })
}
render() {
  let { user, isLoading, oldPass, newPass, passCheck, status,
message, offline} = this.state
  return (
    <div>
      {offline? <p>Anda sedang offline. Cek koneksi anda dan
refresh </p>
      : !isLoading ? <div className="spin-box"><Spinner
animation="border" variant="secondary"/></div>
      : <>
        <div className="row">
          <div className="col-md-5 img-div">
            <div className="img-box"><img src={user.ktm_url}
alt='ktm'className='ktm' /></div>
          </div>
          <div className="col-md-7">
            <h5><b>Nama</b></h5>
            <p className='column'>{user.name}</p>
            <h5><b>NPM</b></h5>
            <p className='column'>{user.npm}</p>
            <h5><b>Waktu diaktifkan</b></h5>
            <p
className='column'>{moment(user.processed_at).format("YYYY-MM-D
H:mm:ss")}</p>
            <button type="button" className="btn btn-primary" data-
toggle="modal" data-target="#editPass">Edit Password</button>
          </div>
        </div>

        {/* Edit Password Modal */}
        <div className="modal fade" id="editPass" tabIndex="-1"
role="dialog" aria-hidden="true">
          <div className="modal-dialog modal-dialog-centered"
role="document">
            <div className="modal-content">

```

```

        <div className="modal-header">
          <h5 className="modal-title" id="exampleModalLabel">Edit
Password</h5>
          <button type="button" className="close" data-
dismiss="modal" aria-label="Close">
            <span aria-hidden="true">&times;</span>
          </button>
        </div>
        <div className="modal-body">
          <form ref='editForm'>
            <div className="form-group">
              <label>Password Lama</label>
              <input type="password" id="oldPass" className="form-
control" placeholder="Password" onBlur={this.handleInput} />
            </div>
            <div className="form-group">
              <label>Password Baru</label>
              <input type="password" id="newPass" className="form-
control" placeholder="Password" onChange={this.handleInput}/>
            </div>
            <div className="form-group">
              <label>Konfirmasi Password Baru</label>
              <input type="password" ref='confirmPass'
className="form-control" placeholder="Konfirmasi Password"
onChange={this.handleRetype}/>
            </div>
            {passCheck === false ?
            <div className="alert alert-warning" role="alert">
              Password tidak cocok
            </div> : <></>}
            {status===400?
            <div className="alert alert-danger" role="alert">
              <strong>{message}</strong>
            </div>
            :status===200?
            <div className="alert alert-success" role="alert">
              <strong>{message}</strong>
            </div>
            :<></> }
            <button type="button" className="btn btn-danger mr-2"
data-dismiss="modal" onClick={this.clear}>{ status===200?
<>Tutup</> : <>Batal</>}</button>
            {status===200? <></> :
            <button type="button" className="btn btn-primary"
onClick={this.submit} disabled={!passCheck || !newPass ||
!oldPass || !this.refs.confirmPass.value}>Simpan
Perubahan</button>}
          </form>
        </div>
      </div>
    </div>
    <Modal show={this.state.showLoading} centered>
      <Modal.Body className='modal-box'>
        Sedang diproses ...
      </Modal.Body>
    </Modal>
  </div>

```

```

        </Modal>
      </>
    }
  </div>
)
}
}
const mapStateToProps = state => {
  return{
    token : state.auth.token
  }
}
export default connect(mapStateToProps, null)(ProfileInfo)

```

Status Skripsi

```

import React, { Component } from 'react'
import { connect } from 'react-redux'
import axios from 'axios'
import {Spinner} from 'react-bootstrap'
import { FaCheck, FaFilePdf, FaTimes } from 'react-icons/fa'
import { Link } from 'react-router-dom'
import moment from 'moment'

export class SkripsiStatus extends Component {
  state={
    skripsi:{},
    isLoading:false,
    Offline:false
  }
  getSkripsi=()=>{
    axios({
      method: 'get',
      url: `/user/skripsi/`,
      headers: {
        Authorization: this.props.token
      }
    }).then(res=>{
      this.setState({
        skripsi: res.data,
        isLoading: true
      })
    }).catch(err=>{
      if(err.response){
        console.log(err.response)
      }
    })
  }
  componentDidMount(){
    if (navigator.onLine){
      this.getSkripsi()
      this.setState({
        offline:false
      })
    }
    else{
      this.setState({

```

```

    offline:true,
  })
}
}
render() {
  let { isLoading, offline, skripsi } = this.state
  return (
    <div className='status-skripsi'>
      <b>Skripsi</b>
      <hr/>
      {offline? <p>Anda sedang offline. Cek koneksi anda dan
refresh </p>
      :!isLoading? <div className="spin-box"><Spinner
animation="border" variant="secondary"/></div>
      :!skripsi ? <div><p className='status-not>Anda Belum
Mengunggah Skripsi</p>
      <Link to='/upload'><button className='btn btn-
primary'>Unggah</button></Link>
      </div>
      : <>
      <table class="table">
        <tbody>
          <tr>
            <th scope="row" className="td-status"><h5>Judul</h5></th>
            <td><p>{skripsi.title}</p></td>
          </tr>
          <tr>
            <th scope="row"><h5>Abstrak</h5></th>
            <td><p className='abs'>{skripsi.abstrak}</p></td>
          </tr>
          <tr>
            <th scope="row"><h5>Abstract</h5></th>
            <td><p className='abs'>{skripsi.abstract}</p></td>
          </tr>
          <tr>
            <th scope="row"><h5>Tahun</h5></th>
            <td><p>{skripsi.published_year}</p></td>
          </tr>
          <tr>
            <th scope="row"><h5>File</h5></th>
            <td>
              <a href={'https://repositori-
skripsi.herokuapp.com/'+skripsi.file_url} alt="skripsi"
target='_blank' rel='noreferrer noopener'><FaFilePdf/></a>
            </td>
          </tr>
          <tr>
            <th scope="row"><h5>Bidang Minat</h5></th>
            <td><p>{skripsi.category===1 ? <>Sistem Cerdas dan Sistem
Grafika (SCSG)</> :
              skripsi.category===2 ? <>Sistem Informasi dan Rekayasa
Perangkat Lunak (SIRPL)</> :
              skripsi.category===3 ? <>Jaringan Komputer dan Komunikasi
Data (JKKD)</> :
              skripsi.category===4 ? <>Ilmu Komputasi dan Metode
Numerik (IKMN)</>

```



```

      : <></> }</p></td>
    </tr>
    <tr>
      <th scope="row"><h5>Kata Kunci</h5></th>
      <td> <p>{skripsi.keywords? <>{skripsi.keywords}</> : <>-
</> }</p></td>
    </tr>
    <tr>
      <th scope="row"><h5>Waktu Unggah</h5></th>
      <td> <p>{moment(skripsi.uploaded_at).format("YYYY-MM-D
H:mm:ss")}</p></td>
    </tr>
    <tr>
      <th scope="row"><h5>Status</h5></th>
      <td>
        {skripsi.is_approved===1 ?<p className='status status-
green'><FaCheck/> Dipublikasikan</p> :
        skripsi.is_approved===0 ? <p className='status status-
red'><FaTimes/> Ditolak</p> : <p className='status status-
muted'>Belum Ditinjau</p>}
      </td>
    </tr>
    {skripsi.is_approved===2 ? <></> :
    <tr>
      <th scope="row">{skripsi.is_approved===1 ? <h5>Waktu
Dipublikasikan</h5> : <h5>Waktu Diproses</h5> }</th>
      <td><p>{moment(skripsi.processed_at).format("YYYY-MM-D
H:mm:ss")}</p></td>
    </tr>
  }
</tbody>
</table>
{skripsi.is_approved===1? <></> :
  <Link to='/reupload'><button className="btn btn-
primary">Edit Unggahan</button></Link>}
</>
}
</div>
)
}
}
const mapStateToProps = state => {
  return{
    token: state.auth.token
  }
}
export default connect(mapStateToProps, null)(SkripsiStatus)

```

Kontak Admin

```

import React, { Component } from 'react'
import { Redirect } from 'react-router-dom'
import { Spinner } from 'react-bootstrap'
import { connect } from 'react-redux'
import axios from 'axios'
import Forum from '../components/Forum'

```

```

class UserForum extends Component {
  state={
    text:'',
    dataLoaded:false,
    isLoading:false,
    chats:[],
    offline:false,
    message:''
  }
  handleText = e =>{
    this.setState({
      [e.target.id] : e.target.value.trim()
    })
  }
  submit = e => {
    let {text} =this.state
    e.preventDefault()
    if(text){
      this.setState({
        isLoading:true
      })
      axios({
        method: 'post',
        url: '/user/insert-text',
        headers: {
          Authorization:this.props.token
        },
        data: {
          text: text,
        }
      }).then(res => {
        this.setState({
          isLoading:false,
          text:''
        })
        this.getForum()
        this.refs.messages.reset()
      }).catch(err=>{
        console.log(err.response)
        if(err.response){
          this.setState({
            message:err.response.data.message
          })
        }
        this.setState({
          isLoading:false,
        })
        setTimeout(() =>
          this.setState({
            message:''
          }), 5000)
        })
      }
    }
    getForum= ()=>{
      axios({

```

```

        method: 'get',
        url: '/user/forum',
        headers: {
            Authorization: this.props.token
        }
    }).then(res=>{
        this.setState({
            dataLoaded:true,
            chats:res.data
        })
    }).catch((err) => {
        if(err.response){
            console.log(err.response.statusText)
        }
        this.setState({
            dataLoaded:true
        })
    })
}
componentDidMount(){
    if (navigator.onLine){
        this.getForum()
        this.setState({
            offline:false
        })
    }
    else{
        this.setState({
            offline:true,
        })
    }
}
render() {
    let {isLoading, dataLoaded, chats, offline, message } =
this.state
    if (!this.props.token || this.props.role==='admin'){
        return <Redirect to={'/'}/>
    }
    return (
        <div className="main-box">
            <h4 className='forum-title'>Kontak Admin</h4>
            <div className="forum-box">
                <Forum dataLoaded={dataLoaded} chats={chats}
message={message }offline={offline}/>
                <form ref='messages' autoComplete='off'>
                    <div className="input-group forum">
                        <textarea type="text" className="form-control"
style={{height:'50px'}} id='text' onBlur={e =>
this.handleText(e)} placeholder="Pertanyaan" aria-
label="pertanyaan" aria-describedby="basic-addon2"/>
                        <div className="input-group-append">
                            {isLoading? <button className="btn btn-primary"
type="button" disabled={true}><Spinner animation="border"
className="spin-green"/></button>:
                            <button className="btn btn-primary" type="button"
onClick={e => this.submit(e)}>Kirim</button>

```

```

        }
      </div>
    </div>
  </form>
</div>
</div>
)
}
}
const mapStateToProps = state => {
  return{
    token : state.auth.token,
    role: state.auth.role
  }
}
export default connect(mapStateToProps, null)(UserForum)

import React, { Component } from 'react'
import moment from 'moment'
import { Spinner } from 'react-bootstrap'

export default class Forum extends Component {
  render() {
    let {dataLoaded, chats, offline, message } =this.props
    return (
      <div className='forum-content'>
        {offline? <div className="text-middle"><p>Anda sedang offline. Cek koneksi anda dan refresh </p> </div>
        :!dataLoaded? <div className="text-middle"><Spinner animation="border" variant="secondary"/></div>
        : <>
          {!chats? <></> : chats.length===0 ? <div className="text-middle"><p>Belum ada pesan yang dikirim</p></div> :
          <>
            {!message? <></> :
              <div className="alert alert-warning" style={{margin: '10px 10px 0px 0px'}} role="alert">
                <strong>{message}</strong>
              </div>
            }
            {chats.map((chat,i)=>
              <div className={chat.npm==='admin'? 'forum-text forum-text-admin' : 'forum-text forum-text-user' } key={i}>
                <div className="head">
                  <b>{chat.npm==='admin'? 'Admin' : chat.name}</b>
                </div>
                <p>{chat.text}</p>
                <p style={{fontSize: 'smaller', color:'grey'}}>{moment(chat.sent_at).format("YYYY-MM-DD H:mm:ss")}</p>
              </div>
            )}
          </>
        }
      </>
    )
  }
}

```

```

    </div>
  )
}
}

```

Lupa Password

```

import React, { Component } from 'react'
import { Spinner } from 'react-bootstrap'
import { connect } from 'react-redux'
import { Redirect } from 'react-router-dom'
import Bg3 from '../components/Bg3'
import axios from 'axios'

export class Forgot extends Component {
  state={
    email:'',
    npm:'',
    checknpm:'',
    displaySection1:'block',
    displaySection2:'none',
    sending:false,
    status:'',
    message:''
  }
  handleInput = e => {
    this.setState({
      [e.target.id] : e.target.value,
    })
  }
  checkNpm = e =>{
    this.setState({
      checknpm : e.target.value,
    })
  }
  sendEmail = e =>{
    e.preventDefault()
    this.setState({
      sending:true
    })
    axios({
      method: 'put',
      url: '/forgot-pass',
      data: {
        npm: this.state.npm,
        email: this.state.email
      }
    }).then(res => {
      this.refs.forgotForm.reset()
      this.setState({
        sending:false,
        status:res.data.status,
        message:res.data.message,
        displaySection1:'none',
        displaySection2:'block'
      })
    }).catch((err) => {

```

```

    console.log(err.response)
    if(err.response){
      this.setState({
        status:err.response.data.status,
        message:err.response.data.message,
        sending:false,
      })
    }
    else{
      this.setState({
        message: 'Network error, Cek Koneksi Anda',
        status: 500,
        sending:false,
      })
    }
  })
}
render() {
  let {status, message, displaySection1, displaySection2,
  sending, email, npm, checknpm} = this.state
  if (this.props.token){
    return <Redirect to={'/'}/>
  }
  return (
    <>
    <Bg3/>
    <div className="main-box">
      <div className='info-box forgot-box'>
        <h3>Lupa Password</h3>
        <form ref='forgotForm' autoComplete='off'>
          <fieldset style={{ display: displaySection1 }}>
            <label className='forgot-text'>Masukan Npm dan Email yang
terdaftar</label>
            <div className='form-group'>
              <input type='email' id='email' onBlur={this.handleInput}
className='form-control' placeholder='Email' />
            </div>
            {email.length>0 && !email.includes('@')}
            <div className='alert alert-warning' role='alert'>
              <strong>Mohon inputkan Email yang valid</strong>
            </div>:<></> }
            <div className='form-group'>
              <input type='number' id='npm' onBlur={this.checkNpm}
onChange={this.handleInput} className='form-control'
placeholder='NPM' />
            </div>
            {checknpm.length !== 12 && checknpm.length > 0 ?
            <div className='alert alert-warning' role='alert'>
              <strong>NPM salah! </strong>memerlukan 12 digit
            </div> : <></> }
            {!message?<></>:
              <div className='alert alert-warning' role='alert'>
                {message}
                {status===422? <p>Cek kembali email dan npm yang anda
masukan</p> : <></>}
              </div>

```

```

    }
    {sending?
      <button type='submit' className='btn btn-primary'
disabled={true}>
        <div className="spin-box"><Spinner animation="border"
className="spin-forgot"/></div>
      </button>
      : <button type='submit' className='btn btn-primary'
onClick={e => this.sendEmail(e)} disabled={!npm ||
npm.length!==12 || !email || !email.includes('@')}>
        Kata Sandi Baru
      </button>
    }
  </fieldset>
  <fieldset style={{ display: displaySection2}}>
    <p><b>{message}</b></p>
    <div>Cek email anda untuk mendapatkan password baru</div>
  </fieldset>
</form>
</div>
</div>
</>
)
}
}
const mapStateToProps = state => {
  return{
    token: state.auth.token
  }
}
export default connect(mapStateToProps, null)(Forgot)

```

PWA

```

//Lifecycle = register -> install -> activate
https://bit.ly/CRA-PWA

//ip localhost ([::1] - 127.0.0.0/8 )
const isLocalhost = Boolean(
  window.location.hostname === 'localhost' ||
  window.location.hostname === '[::1]' ||
  window.location.hostname.match(
    /^127(?:\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)?) {3}$/
  )
)

export function register(config) {
  //Cek apakah service worker support pada browser
  if (process.env.NODE_ENV === 'production' && 'serviceWorker' in
navigator) {
    const publicUrl = new URL(process.env.PUBLIC_URL,
window.location.href)
    //Service worker tidak jalan jika public file diluar url
    if (publicUrl.origin !== window.location.origin) {
      return
    }
  }
}

```

```

//Ketika halaman telah di load
window.addEventListener('load', () => {
  const swUrl = `${process.env.PUBLIC_URL}/service-worker.js`
  if (isLocalhost) {
    checkValidServiceWorker(swUrl, config)
    navigator.serviceWorker.ready.then(() => {
      console.log('Web app served cache-first')
    })
  } else {
    //Register SW
    registerValidSW(swUrl, config)
  }
})
}
}

function registerValidSW(swUrl, config) {
  //-----Register SW
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
      registration.onupdatefound = () => {
        const installingWorker = registration.installing
        if (installingWorker == null) {
          return
        }
        installingWorker.onstatechange = () => {
          //-----install SW
          if (installingWorker.state === 'installed') {
            //Cek apakah sw aktif
            if (navigator.serviceWorker.controller) {
              // Update precached content di fetch tapi SW lama masih
              // di jalankan sampai tab di tutup
              console.log('Content baru telah di fetch dan siap
              digunakan saat tab telah ditutup')
              if (config && config.onUpdate) {
                config.onUpdate(registration)
              }
            } else {
              console.log('Content telah di cached untuk mode offline.')
              if (config && config.onSuccess) {
                config.onSuccess(registration)
              }
            }
          }
        }
      }
    })
    .catch(error => {
      console.error('Error saat register service worker:', error)
    })
}

function checkValidServiceWorker(swUrl, config) {
  // Cek apakah SW sudah ada
  fetch(swUrl, {

```



```

    headers: { 'Service-Worker': 'script' }
  })
  .then(response => {
    const contentType = response.headers.get('content-type')
    if (
      response.status === 404 ||
      (contentType !== null && contentType.indexOf('javascript') ===
-1)
    ) {
      // SW tidak di temukan
      navigator.serviceWorker.ready.then(registration => {
        registration.unregister().then(() => {
          window.location.reload()
        })
      })
    } else {
      // SW ditemukan. Proses dilanjutkan
      registerValidSW(swUrl, config)
    }
  })
  .catch(() => {
    console.log(
      'Tidak ada koneksi Internet. Web App dalam offline mode.')
  })
}

export function unregister() {
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker.ready.then(registration => {
      registration.unregister()
    })
  }
}

```

RIWAYAT HIDUP

DATA PRIBADI

Nama Lengkap : Vega Savera Yuana

NPM : 140810160053

Tempat,Tanggal Lahir: Banda Aceh, 9 Februari 1998

Jenis Kelamin : Perempuan

Agama : Islam

No Telepon : +6285262456235

Email : vegayuana@gmail.com

RIWAYAT PENDIDIKAN

2002 s.d. 2004 : TK YKA Banda Aceh

2004 s.d. 2010 : SD Kartika Banda Aceh

2010 s.d. 2013 : SMP Negeri 1 Banda Aceh

2013 s.d. 2016 : SMA Negeri 10 Fajar Harapan

2016 s.d. sekarang : Universitas Padjadjaran, Jurusan Teknik Informatika

RIWAYAT ORGANISASI

Instansi	Jabatan	Tahun
Badan Eksekutif Himatif FMIPA Unpad	Staf Departemen Pengembangan Organisasi	2017-2018
Panguyuban Aceh (Rakan Aneuk Nanggoe Unpad)	Sekretaris Umum	2018

RIWAYAT KEPANITIAAN

Instansi	Nama Acara	Jabatan	Tahun
Himatif FMIPA Unpad	Informatics Fun Day	Ketua Pelaksana	2018
Himatif FMIPA Unpad	Seminar AMADI	MC Seminar	2018
Himatif FMIPA Unpad	Informatics Festival	Kepala divisi Humas	2018
Himatif FMIPA Unpad	Technopreneurship	Staf Sponsorship and Ticketing	2018
Himatif FMIPA Unpad	Character Building Season	Pembimbing Kelompok	2017
Himatif FMIPA Unpad	Informatics Festival	Staf divisi Humas	2017
Himatif FMIPA Unpad	Technopreneurship	Staf divisi Acara	2017
Himatif FMIPA Unpad	Informatics Sport Art and Games (Instagram)	Staf divisi Humas	2017
Himatif FMIPA Unpad	Informatics Fun Day	Staf divisi Humas	2016

RIWAYAT PRESTASI

Nama Acara	Penyelenggara	Tahun	Keterangan
Huawei in University ICT Competition	Huawei	2019	Juara 2
Seeds for The Future	Huawei	2019	Delegasi Indonesia
Competition of Informatics	Himatif FMIPA Unpad	2019	Juara 1 Android Development
Nut-App in Nutrition Fair IPB	Himagizi IPB	2018	Juara 2 Android Development
Asisten Lab	Teknik Informatika Unpad	2018-2019	Asisten Lab