

**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**

## **ABSTRAK**

## **ABSTRACT**

## DAFTAR ISI

KATA PENGANTAR .....	i
ABSTRAK.....	ii
ABSTRACT .....	iii
DAFTAR ISI.....	iv
DAFTAR TABEL .....	vii
DAFTAR GAMBAR.....	viii
<b>DAFTAR LAMPIRAN .....</b>	<b>x</b>
<b>BAB I.....</b>	<b>1</b>
<b>1.1 Latar Belakang.....</b>	<b>1</b>
<b>1.2 Identifikasi Masalah .....</b>	<b>4</b>
<b>1.3 Batasan Masalah .....</b>	<b>4</b>
<b>1.4 Maksud dan Tujuan Penelitian .....</b>	<b>4</b>
<b>1.5 Manfaat Penelitian .....</b>	<b>5</b>
<b>1.6 Metodologi Penelitian .....</b>	<b>5</b>
<b>1.7 Sistematika Penulisan .....</b>	<b>6</b>
<b>BAB II.....</b>	<b>8</b>
<b>2.1 Repositori Institusi .....</b>	<b>8</b>
<b>2.2 <i>Extreme Programming</i>.....</b>	<b>8</b>
<b>2.3 <i>Progressive Web Apps</i> .....</b>	<b>10</b>
<b>2.4 ReactJS.....</b>	<b>11</b>
<b>2.5 Express .....</b>	<b>12</b>
<b>2.6 Javascript .....</b>	<b>13</b>
<b>2.7 Bootstrap .....</b>	<b>14</b>
<b>2.8 MySQL.....</b>	<b>15</b>
<b>2.9 <i>Unified Modeling Language (UML)</i> .....</b>	<b>15</b>
<b>2.9.1 <i>Use Case Diagram</i>.....</b>	<b>16</b>
<b>2.9.2 <i>Activity Diagram</i> .....</b>	<b>17</b>
<b>2.9.3 <i>Deployment Diagram</i>.....</b>	<b>18</b>
<b>2.10 <i>Entity Relationship Diagram (ERD)</i>.....</b>	<b>19</b>
<b>BAB III.....</b>	<b>Error! Bookmark not defined.</b>
<b>3.1 Fase Eksplorasi .....</b>	<b>23</b>

3.1.1	Kebutuhan Pengguna .....	24
3.1.2	Kebutuhan Data .....	24
3.1.3	Kebutuhan Perangkat Lunak .....	24
3.1.4	Kebutuhan Perangkat Keras .....	25
3.1.5	Kebutuhan Sistem .....	26
3.1.6	<i>User Story</i> .....	27
3.2	Fase Perencanaan .....	29
3.3	Fase Iterasi .....	30
3.3.1	Analisis sistem .....	30
3.3.2	Analisis Arsitektur .....	32
3.3.3	Analisis Basis data .....	33
3.3.4	Desain Antarmuka .....	36
3.3.5	<i>Testing</i> .....	44
3.4	Fase Produksi .....	44
3.5	Fase Pemeliharaan dan Fase Akhir .....	47
<b>BAB IV</b> .....		48
4.1	Impelementasi Program .....	48
4.1.1	Halaman Utama .....	48
4.1.2	Halaman Register .....	51
4.1.3	Menu Login .....	56
4.1.4	Halaman Unggah Skripsi .....	60
4.1.5	Halaman Menu Admin .....	64
4.1.6	Halaman Verifikasi Akun .....	65
4.1.7	Halaman Tinjau Skripsi .....	69
4.1.8	Halaman Detail Skripsi .....	72
4.1.9	Fitur Pencarian dan Penyaringan .....	73
4.1.10	Halaman Profil dan Fitur <i>Edit Password</i> .....	76
4.1.11	Halaman Status Skripsi .....	79
4.1.12	Implementasi PWA .....	83
4.2	Fase Produksi .....	87
4.2.1	Rilisan Kecil .....	87
4.2.2	Hasil Pengujian .....	88
4.3	Fase Pemeliharaan .....	89

4.3.1	Feedback .....	89
4.3.2	Implementasi program .....	90
4.3.3	Rilisan yang diperbaharui .....	<b>Error! Bookmark not defined.</b>
4.4	Fase Akhir .....	90
<b>BAB V</b> .....		91
5.1	Kesimpulan .....	91
5.2	Saran .....	91
<b>DAFTAR PUSTAKA</b> .....		92

## 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 3.1 Kebutuhan Pengguna: Mahasiswa .....	27
Tabel 3.2 Kebutuhan Pengguna: Admin .....	27
Tabel 3.3 <i>User Story</i> .....	28
Tabel 3.4 Prioritas Fitur .....	30
Tabel 3.5 Atribut Tabel Skripsi .....	34
Tabel 3.6 Atribut Tabel Users .....	35
Tabel 3.7 Skenario <i>Testing</i> .....	46
Tabel 4.1 <i>Blackbox Testing</i> Pada Fitur Registrasi.....	55
Tabel 4.2 <i>Blackbox Testing</i> Pada Fitur <i>Login</i> .....	59
Tabel 4.3 <i>Blackbox Testing</i> Pada Fitur Unggah <i>Skripsi</i> .....	64
Tabel 4.4 <i>Blackbox Testing</i> pada Fitur Verifikasi Akun.....	69
Tabel 4.5 <i>Blackbox Testing</i> pada Fitur Tinjau Skripsi.....	72
Tabel 4.6 <i>Blackbox Testing</i> pada Fitur Pencarian dan Penyaringan.....	76
Tabel 4.7 <i>Blackbox Testing</i> pada Fitur <i>Edit Password</i> .....	79
Tabel 4.8 <i>Blackbox Testing</i> Fitur PWA .....	87
Tabel 4.9 Hasil Pengujian Rilis Kecil .....	88



## 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).....	12
Gambar 3.1 Kompatibilitas <i>Browser</i> Chrome dan Mozilla pada Perangkat (Santoni, 2018).....	26
Gambar 3.2 <i>Use Case Diagram</i> .....	31
Gambar 3.3 <i>Activity Diagram</i> .....	32
Gambar 3.4 <i>Deployment Diagram</i> .....	33
Gambar 3.5 <i>Entity Relationship Diagram</i> .....	33
Gambar 3.6 Tabel Model Data Fisik .....	34
Gambar 3.7 Desain Halaman Utama .....	36
Gambar 3.8 Desain Halaman Registrasi Bagian 1.....	37
Gambar 3.9 Desain Halaman Registrasi Bagian 2.....	37
Gambar 3.10 Desain <i>Login Section</i> .....	38
Gambar 3.11 Desain Halaman Detail Skripsi .....	38
Gambar 3.12 Desain Halaman Unggah Skripsi.....	39
Gambar 3.13 Desain Halaman Profil Mahasiswa.....	40
Gambar 3.14 Desain Halaman Status Skripsi .....	40
Gambar 3.15 Desain Halaman Status Skripsi .....	41
Gambar 3.16 Desain Halaman Menu Admin .....	41
Gambar 3.17 Desain Halaman Verifikasi Akun Mahasiswa.....	42
Gambar 3.18 Desain Halaman Tinjau Skripsi.....	43

Gambar 4.1 Halaman Utama .....	49
Gambar 4.2 Halaman Registrasi bagian 1 .....	51
Gambar 4.3 Halaman Registrasi Bagian 2 .....	52
Gambar 4.4 Menu <i>Login</i> .....	56
Gambar 4.5 Halaman Unggah Skripsi .....	60
Gambar 4.6 Halaman Menu Admin.....	65
Gambar 4.7 Halaman Verifikasi Akun .....	66
Gambar 4.8 Halaman Tinjau Skripsi .....	69
Gambar 4.9 Halaman Detail Skripsi .....	72
Gambar 4.10 Halaman Profil.....	76
Gambar 4.11 Bagian <i>Edit Password</i> .....	77
Gambar 4.12 Halaman Status Skripsi .....	<b>Error! Bookmark not defined.</b>
Gambar 4.13 Halaman Unggah Ulang .....	83
Gambar 4.14 Implementasi Fitur <i>Offline</i> .....	85
Gambar 4.15 Implementasi fitur <i>add to homescreen</i> .....	86
Gambar 4.16 Web dalam bentuk <i>Native App</i> .....	87

## **DAFTAR LAMPIRAN**

# **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 digital yang dibuat oleh institusi tersebut (Repanovici, 2009). Hal ini 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 buruk. Bila web memakan waktu yang lama untuk *me-load* 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 — tetapi 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.

Dari latar belakang tersebut 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 tersebut, masalah yang akan dipecahkan dalam penelitian ini adalah bagaimana perancangan repositori institusi menggunakan ReactJS dan mengimplementasikan konsep PWA dengan metode pengembangan perangkat lunak XP.

## 1.3 Batasan Masalah

Dari permasalahan yang disampaikan di atas, akan dilakukan pembatasan masalah pada penelitian sebagai berikut :

1. Web aplikasi dibangun dengan metode pengembangan perangkat lunak XP.
2. Web aplikasi yang dibangun akan mengimplementasikan PWA.
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. Pengguna yang memiliki akun dapat mengunggah dan mengunduh skripsi pada repositori yang dibangun.

## 1.4 Maksud dan Tujuan Penelitian

Maksud dari penelitian ini adalah membuat perancangan repositori skripsi Teknik Informatika Unpad menggunakan ReactJS dan 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 sehingga aplikasi bersifat cepat, *reliable* walau dalam jaringan yang buruk, dan bertingkah seperti *native application* pada perangkat yang digunakan.

### **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 bersifat cepat, *reliable* walau dalam jaringan yang buruk, dan bertingkah seperti *native application* pada perangkat yang digunakan 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 digital 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***

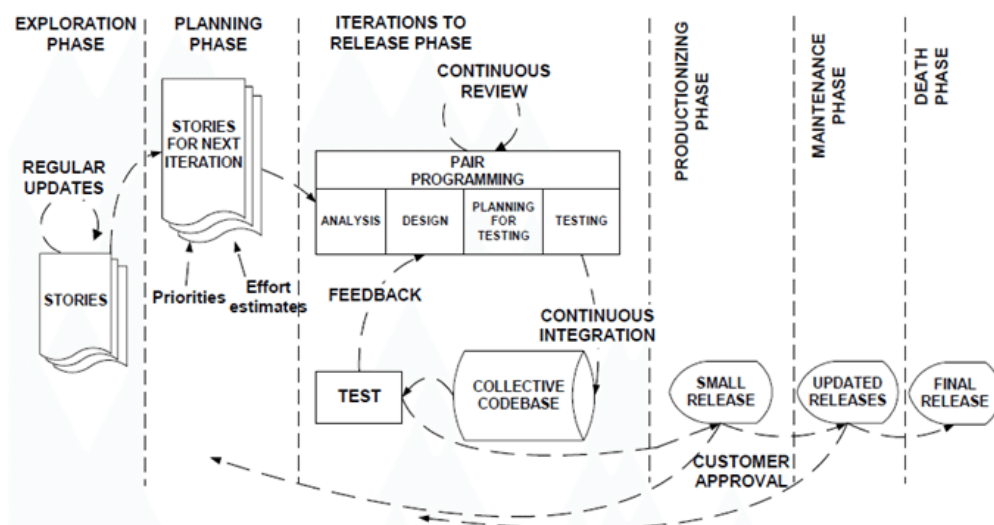
*Extreme programming (XP)* adalah metode pengembangan perangkat lunak yang diciptakan oleh Kent Beck. XP merupakan salah satu metode pengembangan Agile. Metode pengembangan ini cocok digunakan ketika tim dihadapkan dengan *requirement* yang tidak jelas maupun jika terjadi perubahan *requirement* yang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

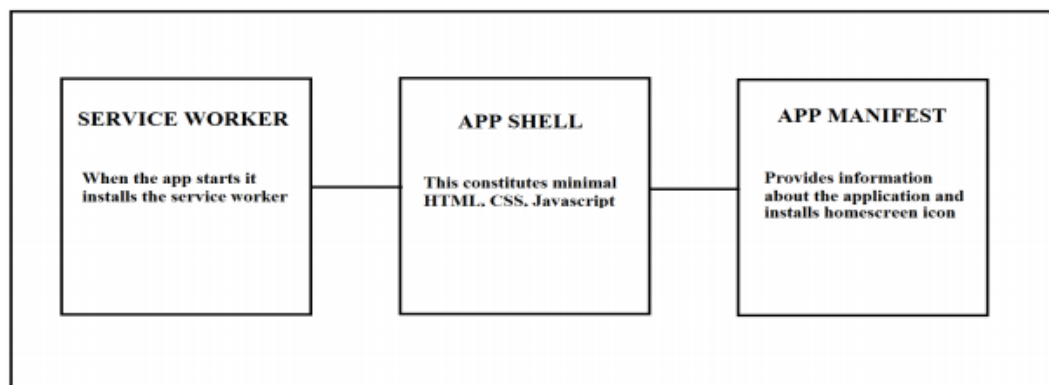
XP memiliki nilai-nilai yaitu kesederhanaan (*simplicity*), komunikasi (*communication*), umpan balik (*feedback*), dan keberanian (*courage*). Metode pengembangan ini juga memiliki manfaat yaitu: fokus terhadap pelanggan, ditekankan pada kerja tim, komunikasi, dan tanggung jawab terhadap kualitas, pengukuran 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

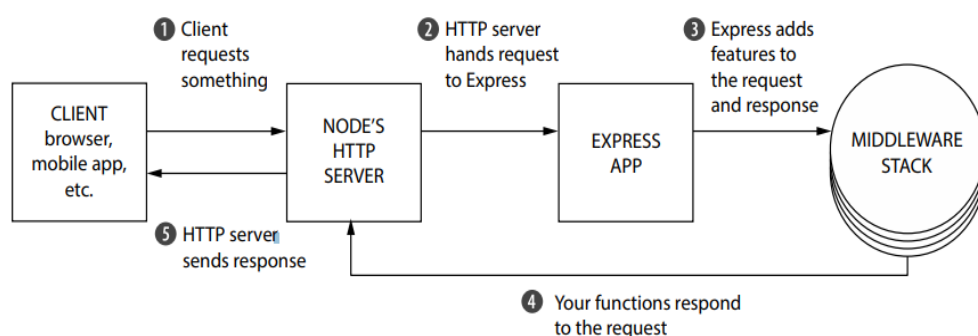
ReactJS adalah kerangka kerja (*framework*) *open-source* yang fleksibel dan kuat untuk mengembangkan aplikasi sisi klien. ReactJS membutuhkan isyarat dari pengembangan sisi-server dan menerapkannya pada elemen HTML, hal itu menciptakan pondasi yang memudahkan *developer* membangun aplikasi web (Freeman,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




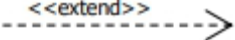
*Unified Modeling Language* (UML) adalah bahasa visual untuk pemodelan sistem dengan menggunakan diagram dan teks-teks pendukung. UML muncul

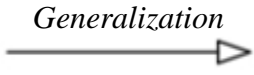
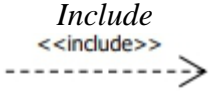
karena adanya kebutuhan standardisasi untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML terbaru adalah UML 2.3 yang terdiri dari 13 macam diagram dan 3 kategori yaitu: *structure diagrams*, *behavior diagrams*, dan *intraction diagrams* (A.S dan Shalahuddin, 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*


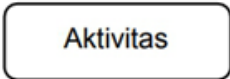
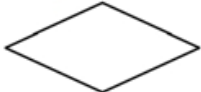

No	Simbol	Keterangan
1		Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor
2		Orang atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat. Biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
3		Komunikasi antara 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


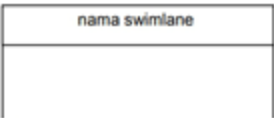
No	Simbol	Keterangan
		meski tanpa <i>use case</i> tambahan itu. Arah panah mengarah pada <i>use case</i> yang ditambahkan.
5		Hubungan generalisasi dan spesialisasi (umum-khusus) antar dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
6		Relasi <i>use case</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*

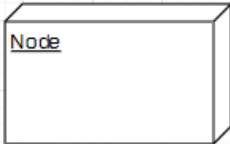



No	Simbol	Keterangan
1	Status Awal 	Menandakan tindakan awal atau titik awal aktivitas untuk setiap diagram aktivitas.
2	Aktivitas 	Menunjukkan aktivitas yang dilakukan sistem
3	Percabangan / <i>decision</i> 	Asosiasi percabangan adalah keadaan dimana terdapat pilihan aktivitas lebih dari satu.
4	Penggabungan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

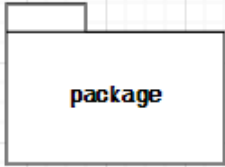
No	Simbol	Keterangan
5	<i>Status akhir</i> 	Menunjukkan bagian akhir dari aktivitas.
6	<i>Swimlane</i> 	<i>Swimlane</i> memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

### 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>Node</i> 	Node mengacu pada perangkat keras atau perangkat lunak yang tidak dibuat sendiri.
2	<i>Link</i> 	Relasi antar <i>node</i>
3	Kebergantungan / <i>dependency</i> 	Ketergantungan antar <i>node</i> , arah panah mengarah pada node yang dipakai
4	<i>Package</i> 	<i>Package</i> merupakan sebuah bungkus dari satu atau lebih node.

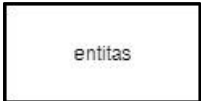
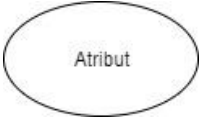



No	Simbol	Keterangan
		

### 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 Ernawati<sup>2</sup>, 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 (Likert, 1932). 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 kita hitung dengan menentukan interval dan kriteria nilai terlebih dahulu dengan menggunakan Persamaan 2.1:

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

Persamaan 5.1 Interval

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

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

Persamaan 5.2 Total Skor

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

$$Nilai\ hasil\ (\%) = \frac{Total\ skor}{5 \times Jumlah\ responden} \times 100$$

## **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 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. Maka web aplikasi akan 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 memiliki kemampuan 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, foto ktm, dan *password*.  
Dan data admin terdiri dari nama, *username*, dan *password*
2. Data skripsi yang terdiri dari judul, penulis, tahun dipublikasi, abstrak, berkas 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 akan 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 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 Mozila pada Perangkat  
(Santoni, 2018)

### 3.1.5 Kebutuhan Sistem

Pada tahapan ini dilakukan survei kepada target pengguna 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 Pengguna: Mahasiswa

Pengguna : Mahasiswa	
Fitur	Deskripsi
Registrasi	Pengguna mendaftarkan diri dengan meng- <i>input</i> nama, NPM dan <i>password</i>
Login	Pengguna Login dengan meng- <i>input</i> NPM dan <i>password</i>
Cari Skripsi	Pengguna dapat mencari skripsi berdasarkan judul, penulis, tahun, atau filter berdasarkan topik
Unduh Skripsi	Pengguna yang telah <i>login</i> dapat mengunduh skripsi mahasiswa lain
Unggah Skripsi	Pengguna dapat mengunggah skripsi
Edit <i>Password</i>	Mengubah kata sandi
Status Skripsi	Mengecek status skripsi telah disetujui admin atau tidak
Tampilan <i>offline</i>	Web tetap memiliki tampilan pada saat koneksi <i>offline</i>
Add <i>to homescreen</i>	Web dapat ditambahkan ke <i>homescreen</i> dan digunakan seperti <i>native app</i>

Tabel 3.2 Kebutuhan Pengguna: Admin

Pengguna : Admin	
Fitur	Deskripsi
Login	Pengguna Login dengan meng- <i>input</i> <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.

### 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* (Choudhari & Suman, 2012).

Tabel 3.3 *User Story*

Judul	Deskripsi	Acceptance Criteria
Register	Sebagai mahasiswa, saya ingin mendaftarkan diri agar memiliki akses terhadap aplikasi web	-Terdapat menu registrasi untuk mahasiswa memperoleh akun -Mahasiswa mengisi nama lengkap, NPM, dan password
Login	Sebagai pengguna, saya ingin mengakses fitur yang tersedia pada aplikasi web	-Mahasiswa mengisi NPM dan password pada kolom login -Admin mengisi username dan password yang terdaftar -Sistem melakukan validasi terhadap data yang di- <i>input</i> -kan -Pengguna mendapatkan akses fitur-fitur
Mencari Skripsi	Sebagai mahasiswa, saya ingin mencari skripsi berdasarkan judul, penulis, tahun dan dapat mendownload skripsi tersebut	-Mahasiswa mengisi keyword ke dalam kolom pencarian untuk mencari skripsi -Mahasiswa menyaring skripsi berdasarkan tahun -Terdapat halaman identitas skripsi dan file-file yang dapat diunduh -Mahasiswa yang telah login dapat mengunduh file skripsi
Unggah Skripsi	Sebagai mahasiswa, saya ingin mengunggah skripsi milik saya ke web	-Terdapat form unggah skripsi. Mahasiswa mengisi identitas skripsi, mata kuliah terkait dengan skripsi, dan mengunggah file skripsi.
Tinjau Unggahan	Sebagai admin, saya ingin meninjau skripsi yang diunggah oleh mahasiswa	-Terdapat list skripsi yang siap untuk ditinjau -Admin dapat menyetujui atau menolak unggahan.
Verifikasi Akun Mahasiswa	Sebagai admin, saya ingin mengecek akun-akun yang terdaftar	-Admin dapat melihat data akun yang terdaftar. -Admin dapat menghapus akun atau memberi aktivasi terhadap akun yang sesuai dengan persyaratan
<i>Edit Password</i>	Sebagai mahasiswa, saya ingin mengubah kata sandi	-Mahasiswa dapat mengubah kata sandi
Status Skripsi	Sebagai mahasiswa, saya ingin mengetahui apakah skripsi yang saya unggah telah dipublikasikan oleh admin	-Mahasiswa dapat melihat status skripsi

Judul	Deskripsi	Acceptance Criteria
PWA ( <i>offline</i> )	Sebagai pengguna saya ingin web aplikasi tetap mempertahankan tampilan dan memberikan informasi saat koneksi <i>offline</i>	-Pengguna dapat melihat tampilan web pada saat <i>offline</i> dan 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>

### 3.2 Fase Perencanaan

Pada fase kedua, penulis memberikan prioritas kepada fitur-fitur yang telah dibentuk pada *user story* ditahapan eksplorasi. Prioritas tinggi akan diberikan pada fitur dengan nilai dan resiko paling tinggi. Fitur dengan prioritas tinggi akan 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. Login membutuhkan token dan proses membedakan jenis user. Maka register dan login akan diimplementasikan terlebih dahulu. Lalu diikuti dengan 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 akun yang akan digunakan dan data skripsi yang akan ditampilkan pada halaman utama, maka kedua fitur ini diimplentasikan setelah register, login, dan unggah skripsi. Kemudian fitur menampilkan skripsi dan unduh



skripsi, fitur pencarian, *edit password*, dan info tinjauan skripsi diimplementasikan setelahnya.

Tabel 3.4 Prioritas Fitur

No	Prioritas Fitur
1	Halaman utama / <i>Landing page</i>
2	Registrasi
3	Login (Autentikasi)
4	Unggah berkas
5	Verifikasi akun (Admin)
6	Tinjau skripsi (Admin)
7	Menampilkan dan mengunduh 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> )

### 3.3 Fase Iterasi

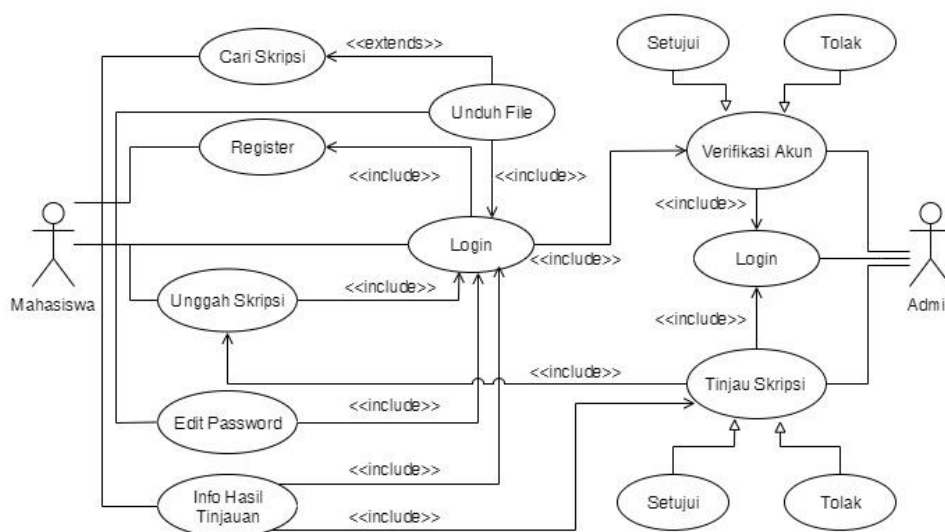
Pada fase iterasi dilakukan analisis, desain dan *testing*. Analisa pada penelitian ini dibagi menjadi tiga bagian yaitu, Analisis analisis sistem, analisis arsitektur, analisis basis data. Desain meliputi desain antarmuka. Analisis sistem dan arsitektur menggunakan diagram Unified Modelling Language (UML) sedangkan analisis basis data menggunakan Entity Relationship Diagram (ERD). Pada tahap ini pula kode diimplementasikan, diuji fungsionalitasnya, dan ditinjau secara berkala.

#### 3.3.1 Analisis sistem

##### 1. *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, unduh skripsi, unggah skripsi, *edit password*, dan melihat info tinjauan skripsi. Pencarian skripsi dapat dilakukan tanpa *login*, sementara fungsi-fungsi lainnya harus melakukan *login* terlebih dahulu. *Login* dapat dilakukan apabila mahasiswa telah mendaftarkan akun dan akun telah diverifikasi oleh admin. Admin dapat melakukan verifikasi akun dan tinjau skripsi apabila admin telah *login*.

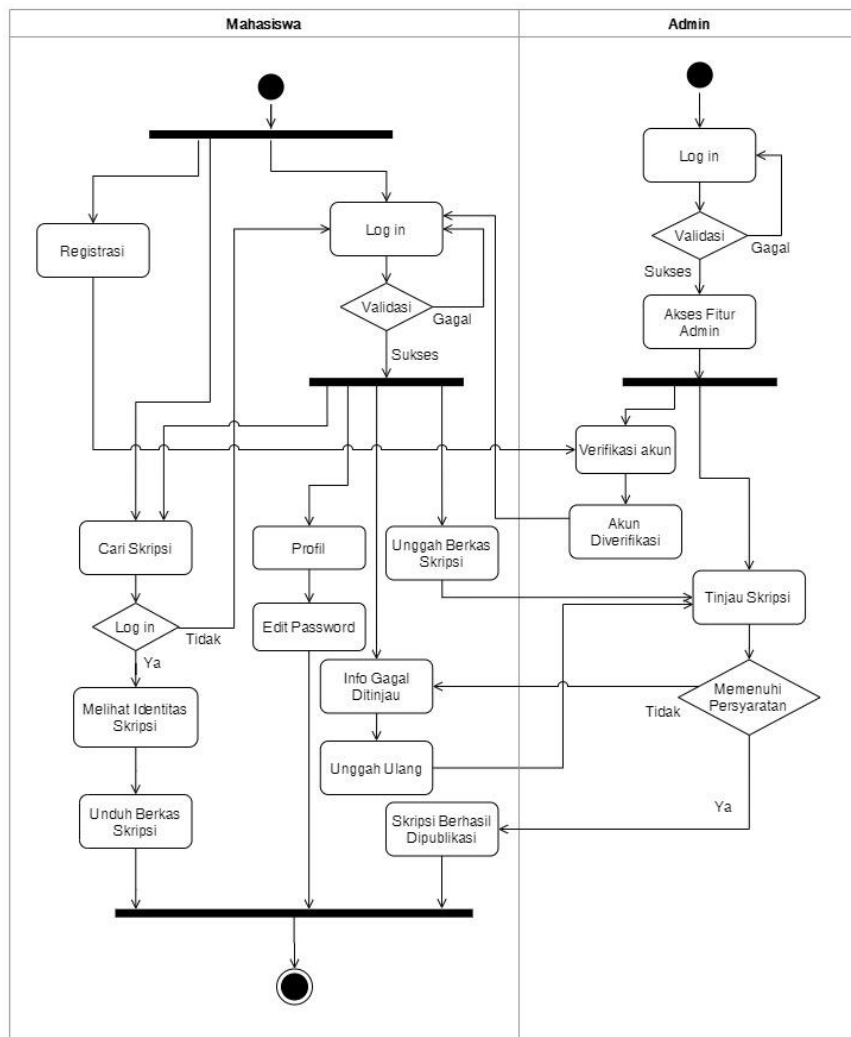


Gambar 3.2 Use Case Diagram

## 2. Activity diagram

Diagram aktivitas menggambarkan aliran kerja dari suatu sistem. Pada gambar 3.3, terdapat 2 *swimlane* dengan label mahasiswa dan admin. Swimlane memisahkan aktivitas berdasarkan pengguna. Mulai dari status awal, mahasiswa dapat melakukan 3 aktivitas yaitu registrasi, *log in*, mencari skripsi. Akun yang didaftarkan akan diverifikasi oleh admin terlebih dahulu. Setelah diaktifkan, mahasiswa dapat *login*. Setelah *log in* mahasiswa dapat mengunggah skripsi. Skripsi yang diunggah akan ditinjau oleh admin sebelum dipublikasikan. Mahasiswa akan menerima pemberitahuan dari hasil tinjauan admin. Mahasiswa

juga dapat mengunduh skripsi mahasiswa lain apabila telah *log in*. Aliran kerja admin ialah, *log in* kemudian dapat memilih melakukan verifikasi akun atau tinjau skripsi.

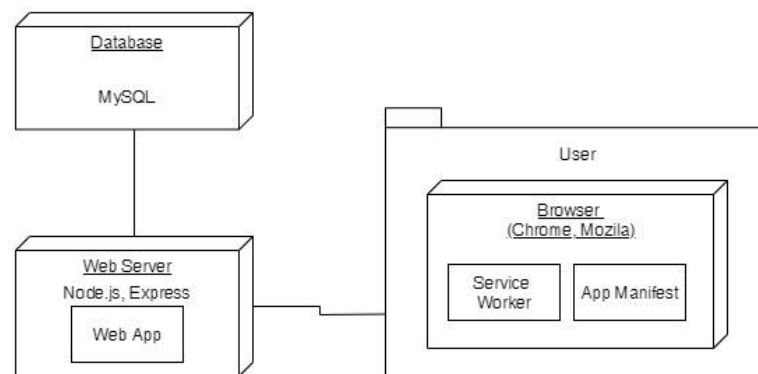


Gambar 3.3 Activity Diagram

### 3.3.2 Analisis Arsitektur Menggunakan *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

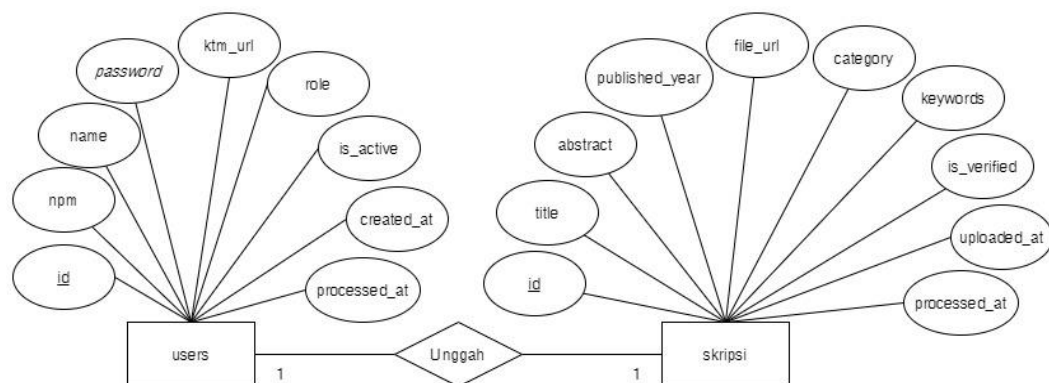
Chrome atau Mozilla. Aplikasi berjalan pada server web dan menggunakan *database* MySQL.



Gambar 3.4 *Deployment Diagram*

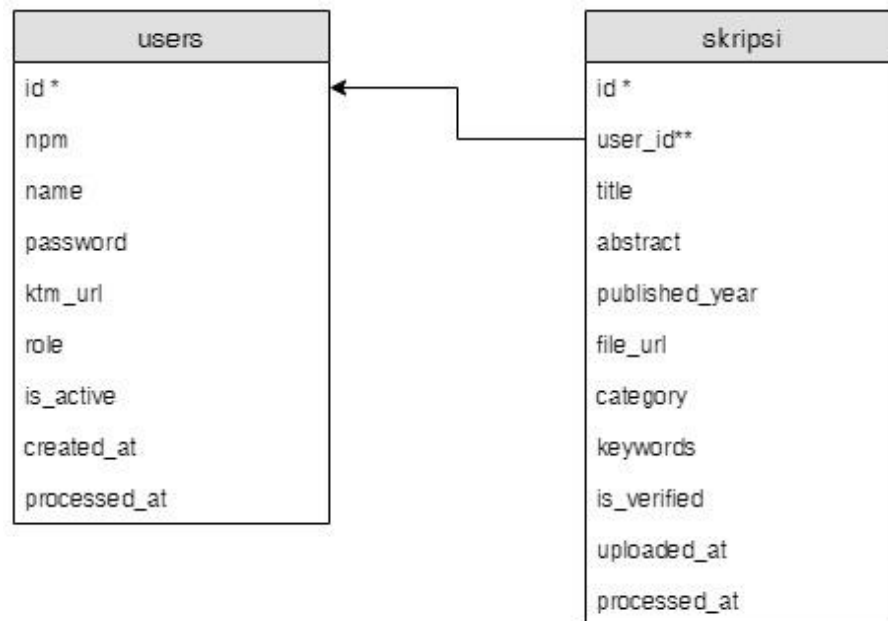
### 3.3.3 Analisis Basis Data Menggunakan ERD (*Entity Relationship Diagram*)

Pada gambar 3.5 terdapat 2 entitas yaitu pengguna dan skripsi yang berarti terdapat 2 tabel pada basis data. Kedua entitas tersebut memiliki relasi *one to one*.



Gambar 3.5 *Entity Relationship Diagram*

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



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 judul, abstrak, *url file* skripsi, kategori, kata kunci, tahun publikasi, waktu diunggah dan ditinjau oleh admin, serta id pengguna dari *user* terkait.

Tabel 3.5 Atribut Tabel Skripsi

No	Atribut	Tipe	Ukuran	Keterangan
1	id	varchar	100	Primary key sebagai kode identitas skripsi
2	user_id	varchar	100	Foreign key dari tabel user
3	title	varchar	255	Judul skripsi
4	abstract	text	-	Abstrak skripsi
5	file_url	varchar	255	Url penyimpanan file skripsi
6	category	tinyint	1	Kategori skripsi berdasarkan bidang minat. Nilai 1 adalah <i>artificial intelligence</i> , 2 adalah sistem informasi, dan 3 adalah jaringan komputer

No	Atribut	Tipe	Ukuran	Keterangan
7	keywords	varchar	255	Kata kunci terkait dengan skripsi untuk memudahkan pencarian
8	published_year	year	-	Tahun skripsi dipublikasikan
9	is_approved	tinyint	4	Terdiri dari 0 (ditolak admin), 1 (disetujui oleh admin), dan 2 belum di proses. Default awal adalah 2
10	uploaded_at	timestamp	-	Waktu skripsi diunggah
11	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 nama, npm, *url file* KTM, *password*, jenis *user*, status aktivasi akun, waktu mendaftar serta diverifikasi oleh admin.

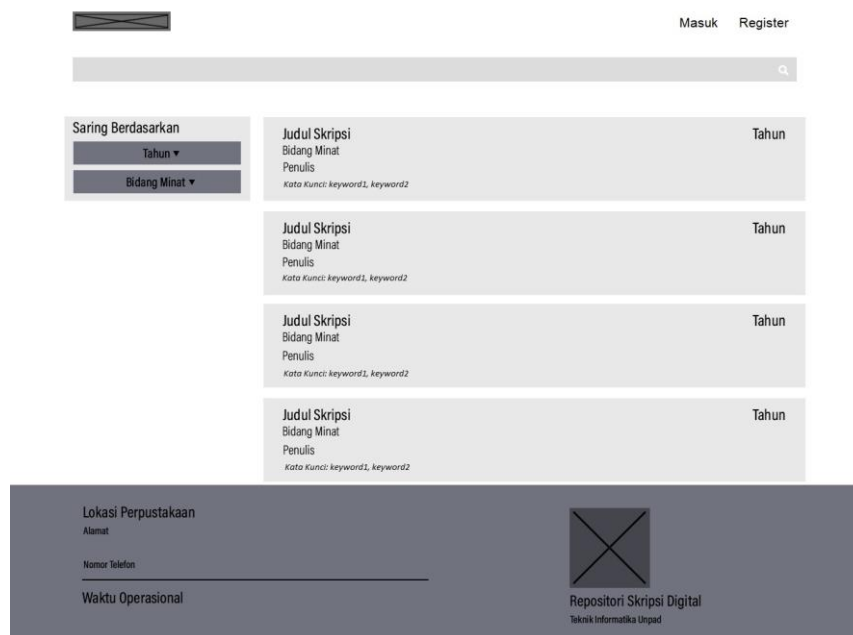
Tabel 3.6 Atribut Tabel Users

No	Atribut	Tipe	Ukuran	Keterangan
1	id	varchar	100	<i>Primary key</i> sebagai kode unik <i>record</i> pengguna
2	name	varchar	255	Nama lengkap pengguna
3	npm	varchar	15	Npm pengguna / <i>username</i> bagi admin. Bersifat unik
4	ktm_url	varchar	255	Url penyimpanan 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 diverifikasi oleh admin

### 3.3.4 Desain Antarmuka

#### 4. Halaman Utama

Rancangan desain antarmuka untuk halaman utama dapat dilihat pada Gambar 3.7. Pada halaman utama, mahasiswa akan disajikan beberapa skripsi yang sudah tersimpan di *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 *user* akan dibawa ke halaman detail dari skripsi tersebut. Apabila mahasiswa belum melakukan *login* maka terdapat button *Login* dan *Signup* pada pojok kanan atas dari halaman *web*.



Gambar 3.7 Desain Halaman Utama

#### 5. Registrasi

Rancangan desain antarmuka untuk halaman registrasi dapat dilihat pada gambar 3.8 dan gambar 3.9. 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, *password*, dan *confirm password*. Apabila data sudah dilengkapi dan mahasiswa menekan tombol *next* untuk lanjut 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 registrasi berhasil. Data akan disimpan di *database* yang menandakan bahwa mahasiswa sudah terdaftar.

Masuk Register

**Register**

Name  
Name

NPM  
NPM

Password  
Password

Confirm Password  
Confirm Password

Next

Lokasi Perpustakaan  
Alamat

Nomor Telepon

Waktu Operasional

Repositori Skripsi Digital  
Teknik Informatika Unpad

Gambar 3.8 Desain Halaman Registrasi Bagian 1

Masuk Register

**Register**

Foto KTM  
Choose file No file chosen

Submit

Lokasi Perpustakaan  
Alamat

Nomor Telepon

Waktu Operasional

Repositori Skripsi Digital  
Teknik Informatika Unpad

Gambar 3.9 Desain Halaman Registrasi Bagian 2



## 6. Login (Autentikasi)

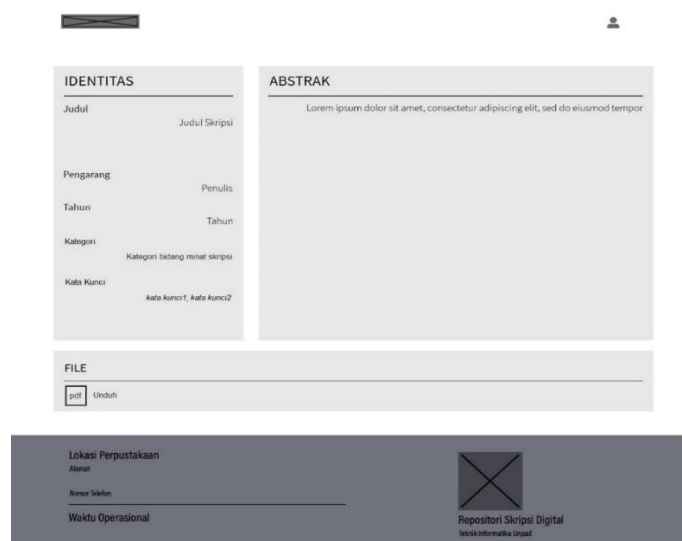
Rancangan desain antarmuka untuk halaman *login* dapat dilihat pada gambar 3.10. *Form* untuk melakukan login dapat diakses di semua halaman apabila pengguna belum melakukan *login* dengan cara mengklik tombol *login* pada *navbar*. Pengguna harus mengisi npm atau *username* bagi admin dan kata sandi untuk melakukan *login*.



Gambar 3.10 Desain *Login Section*

## 7. Detail Skripsi

Rancangan desain antarmuka untuk halaman detail skripsi dapat dilihat pada gambar 3.11. Apabila pengguna telah *login*, maka pengguna dapat mengakses detail skripsi dan mengunduh *file* skripsi pada halaman ini. Info skripsi dibagi menjadi tiga bagian, yaitu identitas, abstrak, dan *file*.



Gambar 3.11 Desain Halaman Detail Skripsi

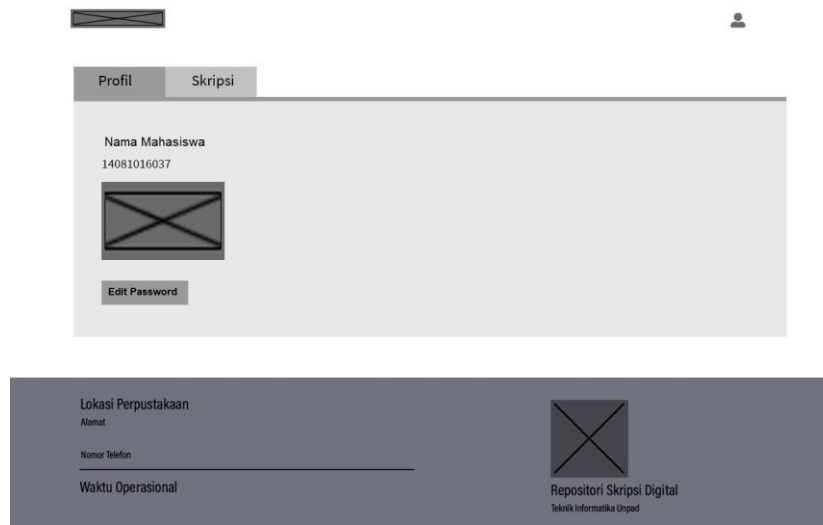
## 8. Unggah Skripsi

Gambar 3.12 Desain Halaman Unggah Skripsi

Rancangan desain antarmuka untuk halaman unggah skripsi dapat dilihat pada gambar 3.12. Mahasiswa yang telah menyelesaikan skripsi dapat mengunggah skripsinya. Data yang harus diisi ialah judul skripsi, tahun publikasi, abstrak, *file* skripsi, bidang minat yang terdiri dari *artificial intelligence*, sistem informasi dan jaringan komputer, serta kata kunci terkait dengan skripsi yang diunggah.

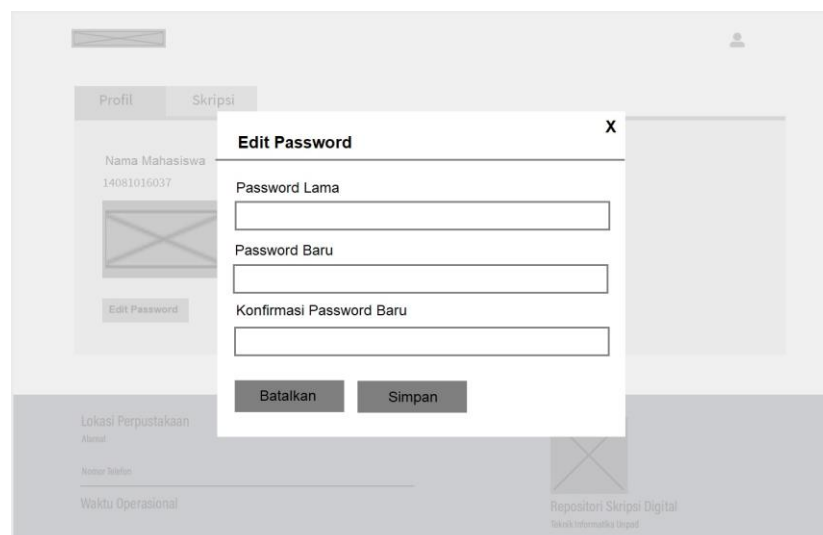
## 9. Profil

Rancangan desain antarmuka untuk halaman profil dapat dilihat pada gambar 3.13. Pada halaman ini mahasiswa dapat melihat nama, npm, foto ktm, dan mengubah *password*. Saat mahasiswa mengklik tombol *edit password* maka akan muncul modal seperti yang ditunjukkan pada gambar 3.14. Mahasiswa harus mengisi *password* lama, *password* dan mengisi konfirmasi *password*



The image shows a user profile page. At the top, there are two tabs: 'Profil' (selected) and 'Skripsi'. Below the tabs, the user's name 'Nama Mahasiswa' and ID '14081016037' are displayed. There is a placeholder for a profile picture and an 'Edit Password' button. Below this, there is a section for 'Lokasi Perpustakaan' with fields for 'Alamat', 'Nomor Telepon', and 'Waktu Operasional'. To the right of these fields is a logo for 'Repository Skripsi Digital' and 'Teknik Informatika Unged'.

Gambar 3.13 Desain Halaman Profil Mahasiswa



The image shows a modal window titled 'Edit Password' with a close button 'X'. It contains three input fields: 'Password Lama', 'Password Baru', and 'Konfirmasi Password Baru'. At the bottom, there are two buttons: 'Batal' and 'Simpan'. The background shows the same user profile page as in Gambar 3.13.

Gambar 3.14 Desain Halaman Status Skripsi

## 10. Status Skripsi

Rancangan desain antarmuka untuk halaman status skripsi dapat dilihat pada gambar 3.15. Data yang diisi pada saat unggah skripsi akan ditampilkan beserta waktu unggah, status skripsi, dan waktu publikasi oleh admin. Apabila skripsi di

tolak maka akan ada tombol unggah ulang. Mahasiswa harus mengulangi proses unggah.

Profil Skripsi

Skripsi

Judul Data Mining

Abstrak Lorem ipsum dolor sit amet, populo minimum detracto te duo, sale adversarium consequuntur id eam. Pro exant melius appellamur no. Ea sea semper splendide. Aliquid sapientem molestias at duo, summo albus philosophia id eum. No brute dissentiunt vim, habeo deserunt laboramus eum te.

Tahun 2017

File

Bidang Minat Sistem Informasi

Kata Kunci Keyword1, keyword2

Waktu unggah 2020-01-26 09:35:29

Status Dipublikasikan

Waktu Dipublikasikan 2020-01-26 09:35:29

Lokasi Perpustakaan

Alamat

Nomor Telepon

Waktu Operasional

Repositori Skripsi Digital  
Teknik Informatika Unpad

Gambar 3.15 Desain Halaman Status Skripsi

## 11. Menu Admin

Admin

Verifikasi Akun

Tinjau Skripsi

Lokasi Perpustakaan

Alamat

Nomor Telepon

Waktu Operasional

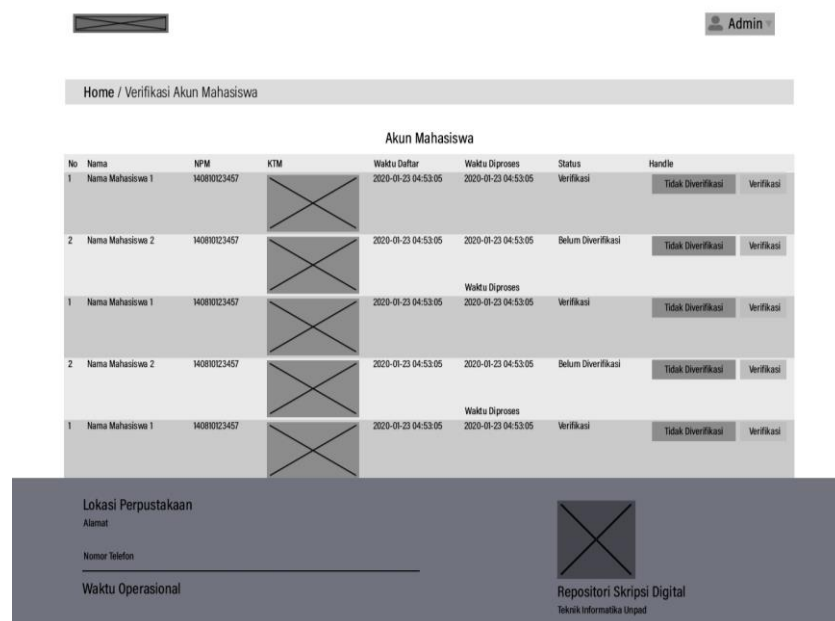
Repositori Skripsi Digital  
Teknik Informatika Unpad

Gambar 3.16 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 adapat dilihat pada gambar 3.16. Menu admin terdiri dari dua hal yaitu verifikasi akun dan tinjau skripsi.

## 12. Verifikasi akun

Rancangan desain antarmuka untuk halaman verifikasi akun dapat dilihat pada gambar 3.17. Halaman verifikasi akun hanya dapat diakses oleh admin. Pada tabel terdapat nama, npm, ktm mahasiswa, waktu daftar, waktu proses, status, dan tombol aksi. Admin dapat melihat data registrasi mahasiswa. Apabila data valid admin akan menverifikasi akun mahasiswa dan mengubah status akun dengan mengklik tombol verifikasi. Dan sebalik bila data tidak valid maka admin dapat mengklik tombol tidak terverifikasi dan menghapus data akun tersebut. Akun mahasiswa yang telah diverifikasi dapat melakukan *login*.



Home / Verifikasi Akun Mahasiswa

Akun Mahasiswa

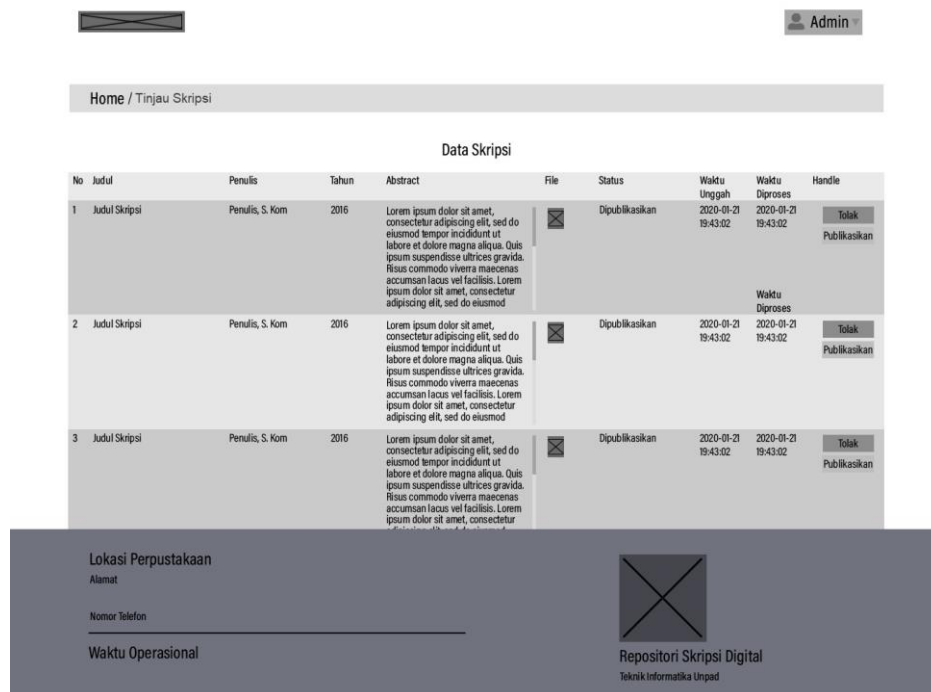
No	Nama	NPM	KTM	Waktu Daftar	Waktu Diproses	Status	Handle
1	Nama Mahasiswa 1	14081023457		2020-01-23 04:53:05	2020-01-23 04:53:05	Verifikasi	Tidak Diverifikasi Verifikasi
2	Nama Mahasiswa 2	14081023457		2020-01-23 04:53:05	2020-01-23 04:53:05	Belum Diverifikasi	Tidak Diverifikasi Verifikasi
1	Nama Mahasiswa 1	14081023457		2020-01-23 04:53:05	Waktu Diproses 2020-01-23 04:53:05	Verifikasi	Tidak Diverifikasi Verifikasi
2	Nama Mahasiswa 2	14081023457		2020-01-23 04:53:05	2020-01-23 04:53:05	Belum Diverifikasi	Tidak Diverifikasi Verifikasi
1	Nama Mahasiswa 1	14081023457		2020-01-23 04:53:05	Waktu Diproses 2020-01-23 04:53:05	Verifikasi	Tidak Diverifikasi Verifikasi

Lokasi Perpustakaan  
Alamat  
Nomor Telepon  
Waktu Operasional

Repositori Skripsi Digital  
Teknik Informatika Unged

Gambar 3.17 Desain Halaman Verifikasi Akun Mahasiswa

### 13. Tinjau Skripsi



Gambar 3.18 Desain Halaman Tinjau Skripsi

Rancangan desain antarmuka untuk halaman tinjau akun dapat dilihat pada gambar 3.18. Halaman verifikasi akun hanya dapat diakses oleh admin. Pada tabel terdapat data skripsi yang diunggah mahasiswa, yaitu judul, nama penulis, tahun, abstrak, file, status, waktu unggah, waktu proses dan tombol aksi. Admin dapat melakukan pengecekan pada data skripsi dan memutuskan apakah data skripsi valid dan siap untuk dipublikasikan.

Tombol publikasikan akan mengubah status skripsi dan menampilkan skripsi tersebut pada halaman utama. Begitu pula sebaliknya dengan tombol tolak. Status skripsi akan berubah dan mahasiswa akan mengetahui status skripsi. Apabila skripsi ditolak mahasiswa dapat mengunggah ulang skripsinya.

### 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 *user* tidak dilakukan pada tahap ini. Proses pengujian fungsional dilakukan setelah tiap fitur selesai dikerjakan.

Metode pengujian yang digunakan pada tahapan ini adalah metode *blackbox testing*, dimana pengujian dilakukan dengan cara memberi beberapa masukan dan memerhatikan output yang diberikan. Hal ini dilakukan untuk menguji berbagai skenario yang dapat dilakukan *user* 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 rilis kecil dihasilkan dan diujicobakan pada pengguna. Web repositori skripsi Teknik Informatika Unpad siap untuk dirilis apabila fitur-fitur pada yang dijelaskan pada bagian eksplorasi telah berhasil diimplementasikan dan berjalan dengan baik. Fitur-fitur tersebut adalah register, login, unggah skripsi, verifikasi akun, tinjau skripsi, detail skripsi, pencarian dan penyaringan, *edit password*, dan cek status skripsi.

Web repositori skripsi Teknik Informatika juga harus telah mengimplementasikan PWA, yaitu memiliki tampilan pada saat *offline* dan tombol *add to homescreen* untuk memenuhi kriteria rilis kecil. 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 3.7:

Tabel 3.7 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 3.8.



Tabel 3.8 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 <i>chrome</i> atau <i>mozilla</i>	Halaman utama / Landing page
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 login dengan akun yang telah didaftarkan	Login (Autentikasi)
5	Pengguna diminta untuk mengunduh <i>file</i> skripsi tertentu	Menampilkan dan mengunduh skripsi
6	Pengguna diminta untuk mengunggah skripsi	Unggah berkas
7	Pengguna diminta untuk mengecek status unggahannya	Status skripsi
8	Pengguna diminta untuk mengubah kata sandi	Edit password
9	Pengguna diminta untuk menambahkan web pada homescreen	PWA (tampilan saat <i>offline</i> dan <i>add to screen shortcut</i> )
10	Pengguna diminta untuk <i>offline</i> dan merefresh halaman	

Setelah responden selesai melakukan skenario pada tabel 3.8, responden diminta untuk mengisi form penilaian terhadap fitur-fitur dari segi fungsi dan UI menggunakan skala likert. Tabel 3.9 berisi pertanyaan pada form.

Tabel 3.9 *Form Usability Testing*

No	Pertanyaan	Fitur
1	Bagaimana penilaian tampilan halaman utama	Landing Page
2	Bagaimana penilaian fungsi yang dihadirkan pada halaman utama	
3	Bagaimana penilaian fungsi pencarian dan filter	Pencarian
4	Bagaimana penilaian fungsi registrasi	Register
5	Bagaimana penilaian tampilan registrasi	
6	Bagaimana penilaian fungsi login	Login
7	Bagaimana penilaian tampilan login	
8	Bagaimana penilaian tampilan halaman detail skripsi	Unduh skripsi
9	Bagaimana penilaian fungsi <i>download</i>	
10	Bagaimana penilaian fungsi <i>upload</i> skripsi	Unggah skripsi
11	Bagaimana penilaian tampilan <i>upload</i> skripsi	
12	Bagaimana penilaian tampilan profil mahasiswa	Edit <i>password</i>

13	Bagaimana penilaian fungsi edit <i>password</i>	
14	Bagaimana penilaian tampilan edit <i>password</i>	
15	Bagaimana penilaian fungsi pada halaman status skripsi	Cek status skripsi
16	Bagaimana penilaian tampilan halaman status skripsi	
17	Bagaimana penilaian terhadap fungsi <i>offline</i> ?	PWA
18	Bagaimana penilaian terhadap tampilan saat <i>offline</i> ?	
19	Bagaimana penilaian terhadap fungsi <i>add to homescreen</i> ?	
20	Apakah fitur <i>add to homescreen</i> membuat tampilan nyaman seperti <i>native app</i> ? Berikan skala penilaian	
21	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 *user* maka akan terjadi pengulangan yang menghasilkan rilisan yang diperbaharui.

Fase akhir dicapai apabila pengguna tidak memiliki *story* tambahan lagi. Pada fase ini, web repositori skripsi Teknik Informatika Unpad telah memenuhi tujuan dan batasan-batasan yang dijelaskan pada bab 1. Produk final akan dirilis sebagai hasil dari fase final.

## BAB IV

### HASIL DAN PEMBAHASAN

#### 4.1 Impelementasi Program

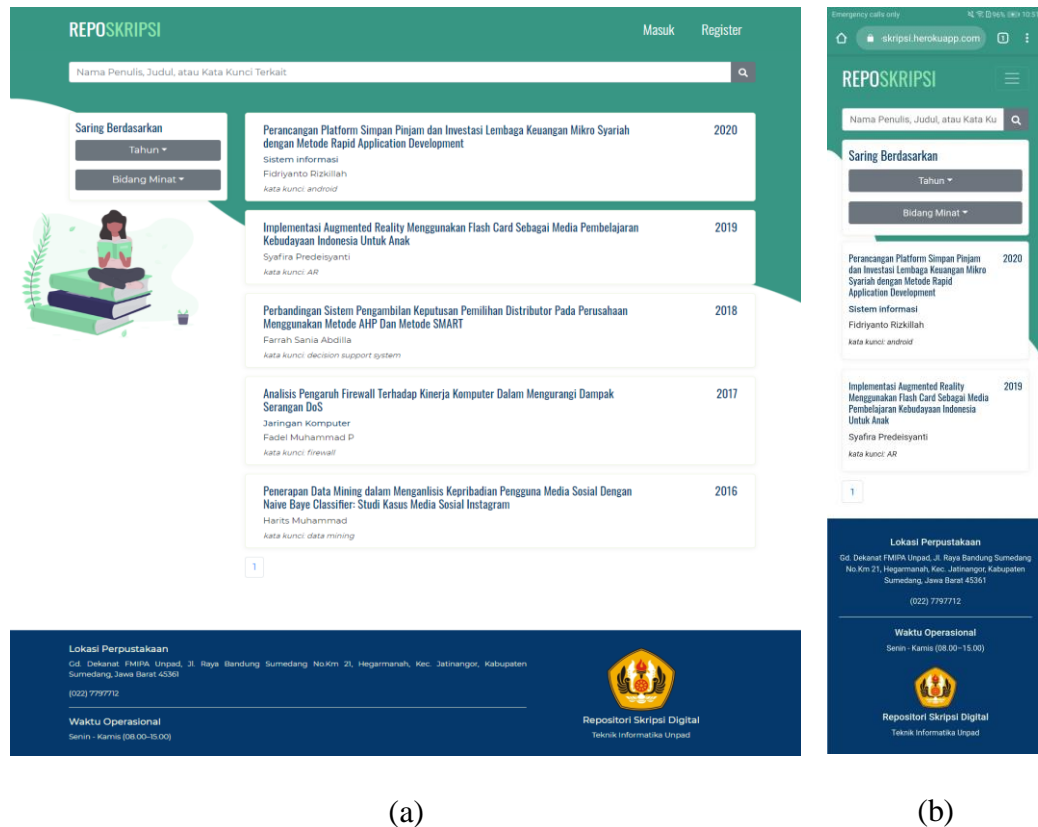
Dalam metode extreme programming, Implementasi program dilakukan pada fase iterasi. Implementasi program mengikuti urutan prioritas fitur sebagaimana yang dijelaskan pada tahap perencanaan. Pada tahapan ini, dilakukan implementasi *user interface (frontend)* yaitu melakukan pengkodean tampilan sesuai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

Setelah implementasi tampilan selesai, dilanjutkan dengan implementasi sistem API 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 user story*. Pada bagian ini akan 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*. Pengguna yang belum *login* hanya dapat mengakses halaman ini dan halaman register. 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.user_id, skripsi.title,
    skripsi.published_year, skripsi.category, skripsi.keywords,
    users.name FROM skripsi join users on users.id =
    skripsi.user_id where is_approved=${1}ORDER BY published_year
    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 akan dikirimkan sebagai *props* ke *component list* untuk di-*loop* dan ditampilkan ke dalam bentuk *card*. 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} isLoading={isLoading}> </ListCard>
```

#### 4.1.2 Halaman Register

Pengguna yang belum memiliki akun harus mendaftarkan diri melalui halaman ini. Halaman register dapat diakses melalui tombol *sign-up* dikanan atas *navbar* selama pengguna belum melakukan *login*. Halaman registrasi berisi *form* yang dipisahkan kedalam dua bagian. *Form* berisi data diri dengan tipe *string* akan diterima terlebih dahulu untuk dilakukan pengecekan data baik secara *frontend* maupun *backend*. Hasil implementasi halaman register dapat dilihat pada gambar 4.2 dan gambar 4.3

REPOSKRIPSI

Masuk Register

### Register

Name  
Name

NPM  
NPM

Password  
Password

Konfirmasi Password  
Password

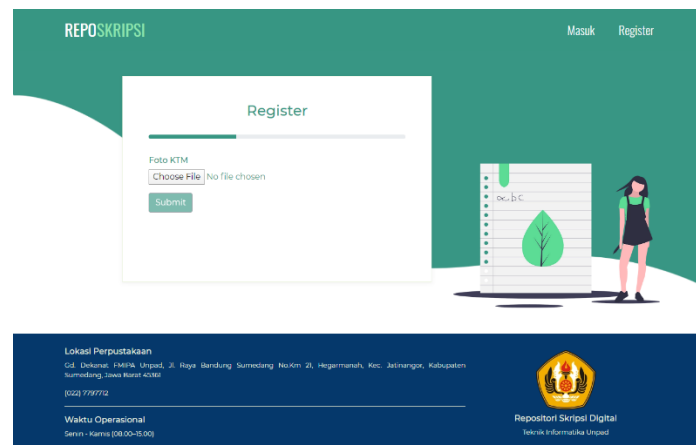
Submit

Lokasi Perpustakaan  
Gd. Diklat PAIKU Unpad, Jl. Raya Bandung-Sumedang No. Km 21, Hegarmanah, Kec. Jatinangor, Kabupaten Sumedang, Jawa Barat 40131  
0221 7797712

Waktu Operasional  
Senin - Kamis (08.00-15.00)

Repositori Skripsi Digital  
Jurnal Informatika Unpad

Gambar 4.2 Halaman Registrasi bagian 1



Gambar 4.3 Halaman Registrasi Bagian 2

Kolom *confirm password* harus diisi dengan benar. Kolom *confirm password* dengan data yang berbeda dengan kolom *password* akan men-trigger *disable* pada tombol *submit*. Menekan tombol *submit* akan memanggil fungsi *submit()*. Fungsi *submit()* akan mengirimkan isi *form* ke *backend* dengan *method post*. Berikut adalah potongan kode *submit* data bagian 1 pada *frontend*:

```
submit = (e) =>{ //Fungsi mengirim data dengan Axios
  e.preventDefault()
  this.setState({
    showLoading:true,
    message:'Sedang diproses ...'
  })
  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
    })
  }).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, Check your connection',
            status: 500,
          })
        }
      })
    }
  })
}

```

Pada *backend*, pengecekan data dilakukan. Pengecekan yang dilakukan adalah pengecekan *field* 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 mendapatkan data dari *database*:

```

router.post('/check-form', (req, res) =>{
  let { name, npm, password } = req.body
  console.log(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 apakah NPM sudah terdaftar sebelumny
  let findUser = `SELECT npm FROM users where role='user' AND npm='${npm}'`
  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 is valid')
  })
})

```



```

    return utils.template_response(res, 200, "Data valid", null)
  })
})

```

Apabila pengecekan sukses dilakukan, pengguna dapat melanjutkan ke bagian 2. Pengguna mengunggah *file* 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 *database*. Dengan demikian pengguna telah terdaftar. Tahap selanjutnya bagi pengguna ialah menunggu verifikasi akun oleh admin. Kriteria-kriteria pada *form* akan uji dengan *blackbox testing* seperti yang tertera pada tabel 4.1. Berikut adalah potongan kode pada register bagian 2 pada *backend*:

```

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)
    }
    password = await bcrypt.hash(password, 10)
    let data = {
      id: genId,
      name: name,
      npm: npm,
      password: password,
      ktm_url:req.file.path,
      created_at:moment().format()
    }
    let sql = 'INSERT INTO users SET ?' //query menambahkan user

```

```

    db.query(sql, data, (err, result)=>{
      if (err){
        console.log('Failed',err)
        return  utils.template_response(res, 400, "Gagal
register", null)
      }
      console.log('Success')
      return  utils.template_response(res, 200, "Register
Berhasil", null)
    })
  })
})

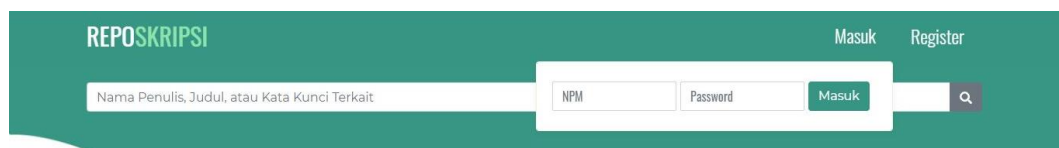
```

Tabel 4.1 *Blackbox Testing* Pada Fitur Registrasi

Skenario	Hasil yang diharapkan	Hasil
Tidak memberi masukan apapun pada <i>form</i> data diri	Pengguna tidak dapat <i>submit</i> karena tombol <i>submit</i> tidak aktif	Berhasil
Mengosongkan salah satu <i>field</i>	Muncul pesan untuk mengisi semua <i>field</i>	Berhasil
Meng- <i>input</i> -kan npm yang telah terdaftar	Muncul pesan bahwa NPM telah terdaftar setelah pengguna mengklik tombol <i>submit</i>	Berhasil
Meng- <i>input</i> -kan npm yang salah (bukan angka, !=12 digit)	Muncul pesan bahwa NPM harus 12 digit setelah pengguna selesai menulis NPM	Berhasil
Tidak mengisi kolom <i>confirm password</i> dengan benar	Pengguna tidak dapat <i>submit</i> karena tombol <i>submit</i> tidak aktif	Berhasil
Mengisi semua <i>field</i> data diri dengan benar	Pengguna dapat lanjut ke bagian kedua	Berhasil
Tidak mengunggah <i>file</i> 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 registrasi berhasil dilakukan dan data pengguna disimpan di <i>database</i>	Berhasil

### 4.1.3 Menu Login

Tombol *login* berada kanan atas halaman pada tampilan desktop dan berada dalam menu *hamburger* pada tampilan *mobile*. Menu *login* akan selalu ada pada *navbar* selama tidak ada token pengguna yang tersimpan pada *state global* Redux. 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": {
                "id": user.id,
                "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) {
                console.log('heo')
                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)
    }
})
})

```

```

submitLogin = e => {
    e.preventDefault()
    this.setState({
        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 => { //dilakukan saat login sukses
    let loginInfo = res.data.data
    console.log ('login info', loginInfo)
    this.setState({
      showLoading:false
    })
    if (loginInfo){
      this.setState({
        status:res.data.status,
        showModal:true,
      })
      if (loginInfo.isLogged) {
        this.props.login(loginInfo) //set state global
        this.setState({
          loginState:true,
        })
      }
      scrollToTop()
      setTimeout(() =>
        this.setState({
          showModal:false
        }), 1000)
    }
    else{
      this.setState({
        status:500
      })
    }
  }).catch((err) => { //dilakukan saat login gagal
    this.setState({
      showLoading:false
    })
    if(err.response){
      this.setState({
        message:err.response.data.message,
        status:err.response.data.status,
      })
    } else{
      this.setState({
        status: 500,
      })
    }
  })
}
...
const mapDispatchToProps = dispatch => { //Redux
  return {
    login: (loginInfo) => dispatch(setToken(loginInfo)),
    logout: () => dispatch(delToken())
  }
}

```

Info login disimpan pada *state global* agar dapat diakses oleh semua *component*. JWT nantinya akan dikirimkan melalui properti *authorization* pada *request* yang mengharuskan pengguna untuk *login*. *State global* disimpan menggunakan Redux. Berikut adalah potongan kode Redux untuk menyimpan info login pengguna:

```
export const setToken = (loginInfo) => {
  cookie.set('token', loginInfo.token, {path: '/', maxAge: 86400})
  cookie.set('role', loginInfo.role, {path: '/', maxAge: 86400})
  return dispatch => {
    dispatch({ type: "SET_TOKEN", payload: loginInfo })
  }
}

const authReducer = (state = initialState, action) => {
  switch (action.type) {
    case 'SET_TOKEN':
      return {
        ...state,
        role: action.payload.role,
        token: action.payload.token
      }
    ...
  }
}
```

Fitur *login* akan diuji menggunakan *blackbox testing* dengan skenario seperti yang tertera pada tabel 4.2.

Tabel 4.2 *Blackbox 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

Skenario	Hasil yang diharapkan	Hasil
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

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, 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.

REPOSKRIPSI

Unggah

Judul \*

Judul Skripsi

Tahun \*

Tahun Publikasi Skripsi

Abstrak \*

Input Abstrak

File \* (Maks 20mb)

Choose File No file chosen

Bidang Minat Skripsi

Bidang Minat

Kata Kunci

Input 1-3 kata kunci yang berkaitan dengan skripsi (pisahkan dengan koma)

Kata Kunci

Submit

Lokasi Perpustakaan  
 Cid. Tekanan PIRPA Unpad 31 Rajat Bandung Sumedang No Km 21, Hegamamah, Kec. Jatisirong, Kabupaten Sumedang Jawa Barat 40559  
 (0262) 777772

Waktu Operasional  
 Senin - Kamis 08.00 - 05.00

Repositori Skripsi Digital  
 Teknik Informatika Unpad

Gambar 4.5 Halaman Unggah Skripsi

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 dan 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
  })
  console.log(this.state)
  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('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) =>{ //dilakukan bila unggah sukses
        this.refs.uploadForm.reset()
        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,
          })
        }
      })
    }
  })
}

```

*Form* akan dicek sebelum dikirim ke *database*. Pengecekan yang dilakukan adalah pengecekan *field* data yang tidak boleh kosong, pengecekan ekstensi dan ukuran *file* yang diunggah. Id *user* 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-kolm data diisi
    let { title, year, abstract, category, keywords} = req.body
    console.log(req.body)
  })
})

```

```

    if (!title || !year || !abstract ) {
      return utils.template_response(res, 400, "Judul, tahun, dan abstrak harus diisi" , null)
    }
    if(!req.file){
      return utils.template_response(res, 400, "File skripsi harus diunggah" , null)
    }
    //mendapatkan user id 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 user_id='${payload.id}' 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(),
        user_id: payload.id,
        title: title,
        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 *blackbox testing* dengan skenario seperti yang tertera pada tabel 4.3.

Tabel 4.3 *Blackbox Testing* Pada Fitur Unggah Skripsi

Skenario	Hasil yang diharapkan	Hasil
Mengosongkan semua <i>field</i> data	Pengguna tidak dapat submit data karena tombol submit tidak aktif	Berhasil
Tidak mengisi salah satu <i>field</i> data yang wajib diisi	Pengguna tidak dapat submit data karena tombol submit tidak aktif	Berhasil
Tidak mengisi <i>field</i> 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

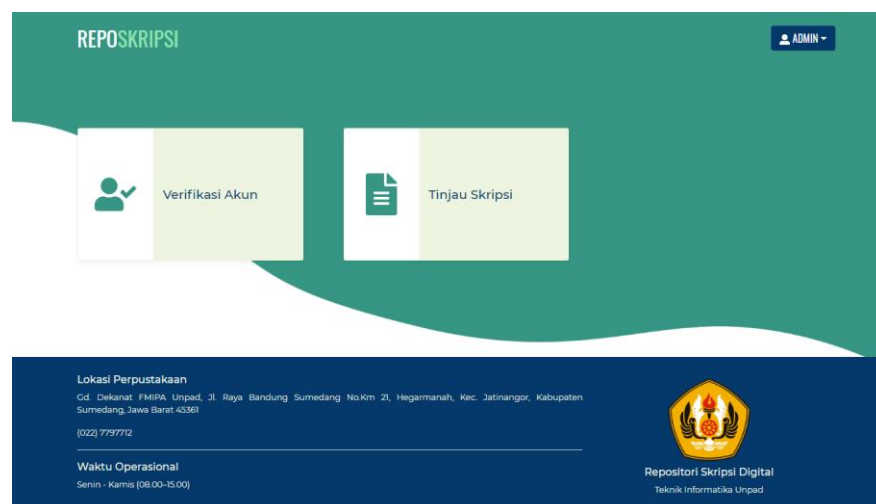
#### 4.1.5 Halaman Menu Admin

Admin yang telah melakukan *login* akan 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* akan membawa admin ke halaman utama web dan melihat skripsi yang telah dipublikasi. Implementasi halaman menu admin dapat dilihat pada gambar 4.6.

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* akan 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()
    })
  },
  ...
}
```

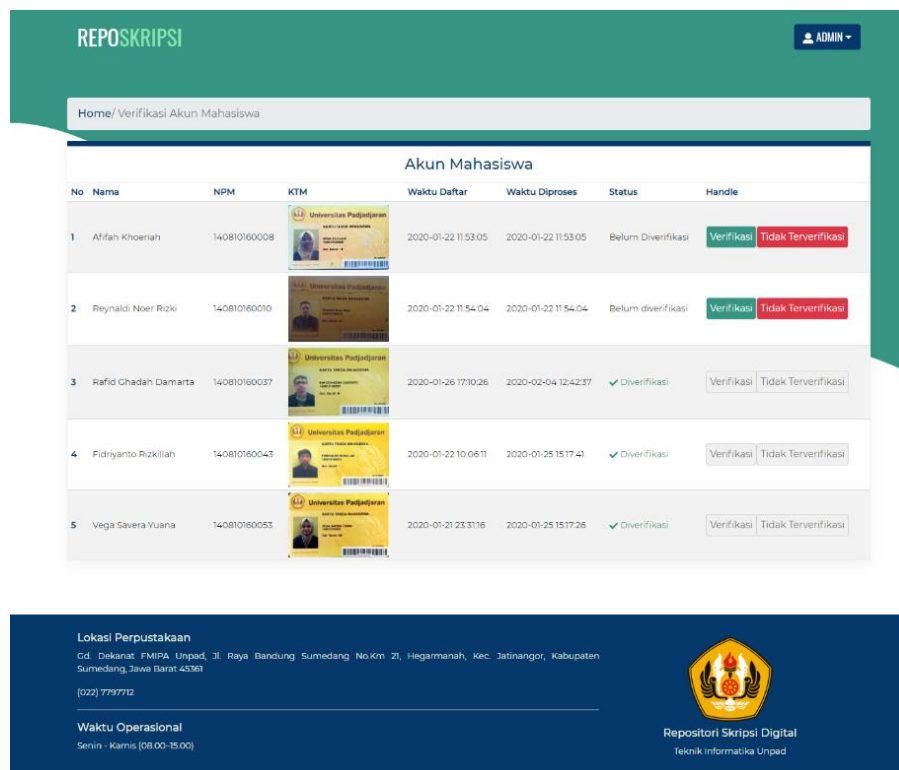


Gambar 4.6 Halaman Menu Admin

#### 4.1.6 Halaman Verifikasi Akun

Halaman verifikasi akun hanya dapat diakses oleh admin. Halaman ini berisi tabel data mahasiswa yang telah melakukan registrasi. Pada tabel terdapat nama, npm, ktm mahasiswa, waktu daftar, waktu proses, status, dan tombol aksi. Proses

yang terjadi pada halaman ini ialah *frontend* melakukan *request* data akun mahasiswa setelah inisial *rendering*. Kemudian data dari *table* user ditampilkan. Implementasi halaman verifikasi akun dapat dilihat pada gambar 4.7.



Gambar 4.7 Halaman Verifikasi Akun

Pada tabel terdapat 2 tombol aksi. Tombol aksi terdiri dari tombol verifikasi dan tidak terverifikasi. Tombol verifikasi 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 verifikasi pada *frontend* dan *backend*:

```
verified = (id) => {
  this.setState({ //muncul modal berisi pesan loading
    showLoading:true
  })
  axios({ //mengubah data dengan method put
    method: 'put',
    url: `/admin/verified/${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('/verified/:id', (req, res) =>{
  const id = req.params.id
  let time=moment().format()
  //query mengubah data pengguna
  let sql = `UPDATE users SET is_active=${true},
processed_at='${time}' where id='${id}'`
  db.query(sql, (err, result)=>{
    if (err) {
      console.log(err)
      return utils.template_response(res, 500, 'Gagal' , null)
    }
    return utils.template_response(res, 200, 'Berhasil' , null)
  })
})

```

Tombol tidak terverifikasi akan menghapus data pengguna dari *database*.

Akan muncul *pop-up* saat tombol tidak diverifikasi diklik untuk mencegah menghapus akun secara tidak sengaja. Berikut adalah potongan kode tombol tidak terverifikasi pada *frontend* dan *backend*:

```

unverified = (id) => {
  this.setState({ //memunculkan modal berisi pesan loading
    showLoading:true
  })
  axios({ //menghapus pengguna dengan method delete axios
    method: 'delete',
    url: `/admin/unverified/${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('/unverified/:id', (req, res) =>{
  const id = req.params.id
  let find = `SELECT ktm_url from users WHERE id='${id}' LIMIT 1`
  db.query(find, (err, result)=>{
    if (err) {
      console.log(err)
      return utils.template_response(res, 500, 'Gagal menghapus
foto KTM' , null)
    }
    else{
      //menghapus file ktm dari local folder
      let file_url=result[0].ktm_url
      fs.unlink(file_url, (err) => {
        if (err) console.log(err)
        console.log(file_url, 'was deleted')
      })
      //query menghapus user dari database
      let sql = `delete from users where id='${id}'`
      db.query(sql, (err, result)=>{
        if (err) {
          console.log(err)
          return utils.template_response(res, 500, 'Akun gagal
Dihapus' , null)
        }
        console.log('Success')
        return utils.template_response(res, 200, 'Berhasil' ,
null)
      })
    }
  })
})

```

Fitur verifikasi akun akan diuji menggunakan *blackbox testing* dengan skenario seperti yang tertera pada tabel 4.4.





Pada tabel terdapat 2 tombol aksi. Tombol aksi terdiri dari tombol tolak dan publikasi. Tombol publikasi akan mengubah nilai `is_approve` 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 akan 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)
  })
})
```

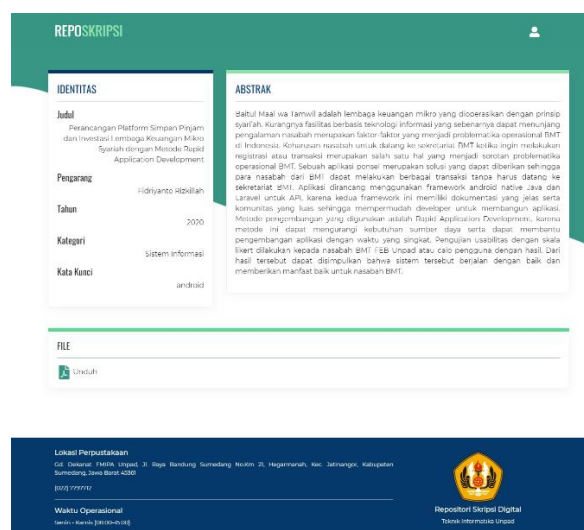
Fitur tinjau skripsi akan diuji menggunakan *blackbox testing* dengan skenario seperti yang tertera pada tabel 4.5.

Tabel 4.5 *Blackbox 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

Halaman detail skripsi dapat diakses saat mengklik salah satu skripsi pada halaman utama. Halaman ini hanya dapat diakses setelah pengguna *login*. Halaman detail skripsi 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. Hasil implementasi halaman detail skripsi dapat dilihat pada gambar 4.9. Berikut adalah potongan kode halaman detail skripsi pada *frontend* dan *backend*:



Gambar 4.9 Halaman Detail Skripsi

```

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

```

```

router.get('/detail/', (req, res) =>{
  let { id } = req.query
  //query join tabel skripsi dan user
  let sql = `SELECT skripsi.id, skripsi.user_id, skripsi.title,
skripsi.abstract, skripsi.file_url, skripsi.published_year,
skripsi.category, skripsi.keywords, users.name FROM skripsi join
users on users.id = skripsi.user_id where
skripsi.is_approved=${1} and skripsi.id='${id}' limit 1`
  db.query(sql, (err, result)=>{
    if (err) console.log(err)
    res.send(result)
  })
})

```

#### 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 akan diuji dengan skenario pada tabel 4.6. Berikut adalah kode fitur pencarian dan penyaringan.

```

onChange = (e) =>{
  let text = e.target.value.toLowerCase()
  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)=>{
  let year = e.target.id
  let {skripsi, cat} = this.state
  //All
  if(year==='Tahun'){
    if (cat){
      let filteredData = skripsi.filter(item => {
        // eslint-disable-next-line
        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 => {
        // eslint-disable-next-line
        return item.published_year == year && item.category==
cat
      })
    }
    else{
      filteredData = skripsi.filter(item => {
        // eslint-disable-next-line
        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
    //All
    if (cat==='all'){
      if (year){
        let filteredData = skripsi.filter(item => {
          // eslint-disable-next-line
          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 => {
          // eslint-disable-next-line
          return item.published_year == year && item.category==
cat
        })
      }
      else{
        filteredData = skripsi.filter(item => {
          // eslint-disable-next-line
          return item.category==cat
        })
      }
      this.setState({
        cat:cat,
        categorySelection:text,
        skripsiFiltered:filteredData,
        skripsiFilteredTemp:filteredData,

```

```

    })
  }
}

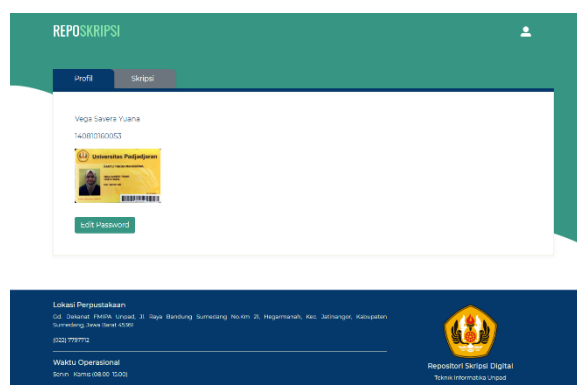
```

Tabel 4.6 *Blackbox 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

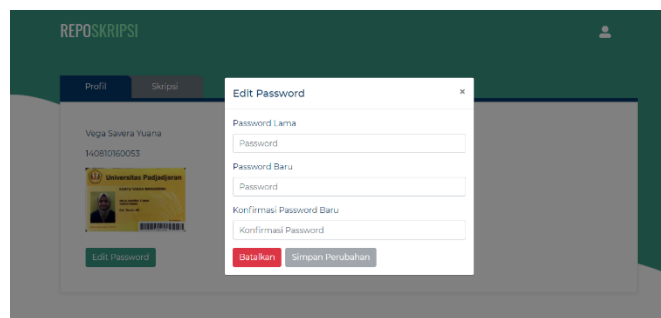
#### 4.1.10 Halaman Profil dan Fitur *Edit Password*

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 ktm. Implementasi halaman profil dapat dilihat pada gambar 4.10.



Gambar 4.10 Halaman Profil

Pada halaman ini mahasiswa dapat mengubah kata sandi. *Form* yang harus diisi ialah *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 *request update* kata sandi akun pengguna. 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
id='${payload.id}' limit 1`
    db.query(findOldPass, async(err, result)=>{
      try{
        if( await bcrypt.compare(oldPass, result[0].password)){
          console.log('Old password matches the database')
          if (oldPass===newPass){
            return utils.template_response(res, 400, "Password
baru harus berbeda dari password lama", null)
          }
          else{
            let password = await bcrypt.hash(newPass, 10)
            //query mengubah password ke database
            let sql = `UPDATE users SET password='${password}'
where id='${payload.id}'`
            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* akan diuji menggunakan *blackbox testing* dengan skenario yang dapat dilihat pada tabel 4.7.

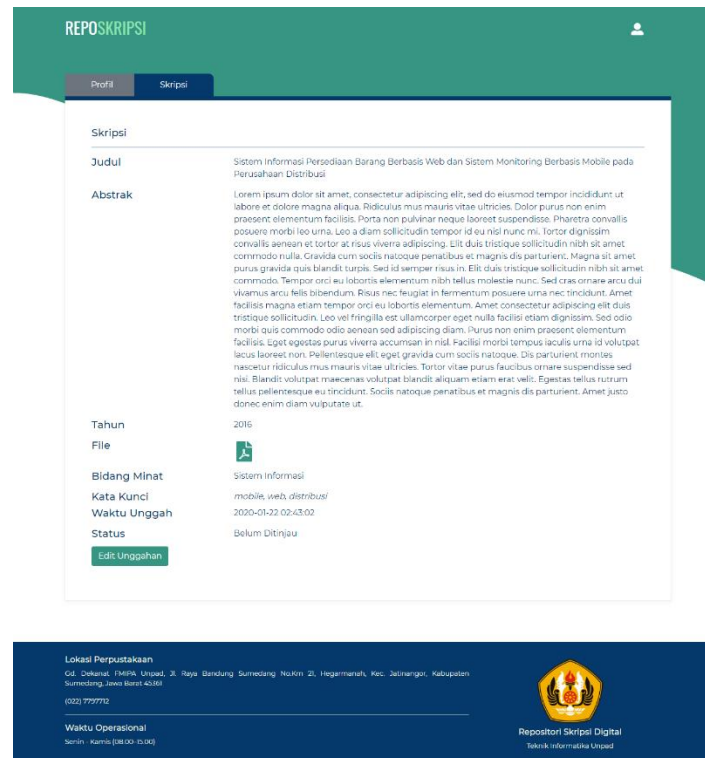
Tabel 4.7 *Blackbox Testing* pada Fitur *Edit Password*

Skenario	Hasil yang diharapkan	Hasil
Tidak mengisi semua <i>field</i>	Tombol simpan perubahan tidak aktif. Tombol batalkan aktif	Berhasil
<i>Password</i> lama 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 submit	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 selesai input dan tombol simpan perubahan tidak aktif	Berhasil
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

Halaman status skripsi berisi informasi skripsi yang telah di unggah. Bila mahasiswa belum mengunggah apapun maka halaman menampilkan pesan bahwa

pengguna belum mengunggah skripsi. Pada halaman ini mahasiswa dapat melihat judul, abstrak, tahun publikasi, *file*, bidang minat, kata kunci, waktu unggah, status, dan waktu proses. Hasil implementasi halaman status skripsi dapat dilihat pada gambar 4.12.



Gambar 4.12 Halaman Status Skripsi

Skripsi dengan status belum ditinjau tidak akan menampilkan waktu proses karena waktu proses adalah informasi waktu dimana admin meninjau skripsi. Skripsi dengan status belum ditinjau atau ditolak akan menampilkan tombol edit unggahan. 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: cookie.get('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 user_id='${payload.id}'
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. Implementasi halaman unggah ulang dapat dilihat pada gambar 4.13

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)
    }
  })
})

```

```

    }
    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
user_id='${payload.id}' LIMIT 1`
    db.query(checkSkripsi, (err, skripsi)=>{
        console.log('data', skripsi[0])
        if (err){
            console.log('err', err)
            return utils.template_response(res, 400, err.response,
null)
        }
        if(skripsi.length===0){
            return utils.template_response(res, 422, "Pengguna belum
mengunggah file" , null)
        }
        if(skripsi[0].is_approve===1){
            return utils.template_response(res, 422, "File sudah
dipublikasikan" , null)
        }
        let old_file=skripsi[0].file_url
        console.log('Old File', old_file)
        //hapus file lama dari local folder
        fs.unlink(old_file, (err) => {
            if (err) console.log(err);
            console.log(old_file, 'was deleted');
        })
        let id = skripsi[0].id
        let path_url = req.file.path
        console.log('new file', path_url)
        let post = {
            title: title,
            abstract: abstract,
            published_year: year,
            file_url: path_url,
            category: category,
            keywords: keywords
        }
        //query mengubah unggahan
        let sql = `UPDATE skripsi SET
uploaded_at='${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)
        })
    })
})

```

```

    })
  })

```

**REPOSKRIPSI**

**Unggah Ulang**

Judul \*

Tahun \*

Abstrak \*

File \* (Maks 20mb)

Bidang Minat Skripsi

Kata Kunci

Submit

**Lokasi Perpustakaan**  
Gd. Diklat IMIA Unpad, Jl. Raya Bandung-Sumedang No. Km. 21, Hegermanah, Kec. Jatinangor, Kabupaten Sumedang, Jawa Barat 40132  
(026) 7797712

**Waktu Operasional**  
Senin - Kamis (08.00-15.00)

**Repositori Skripsi Digital**  
Teknik Informatika Unpad

Gambar 4.13 Halaman Unggah Ulang

#### 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:

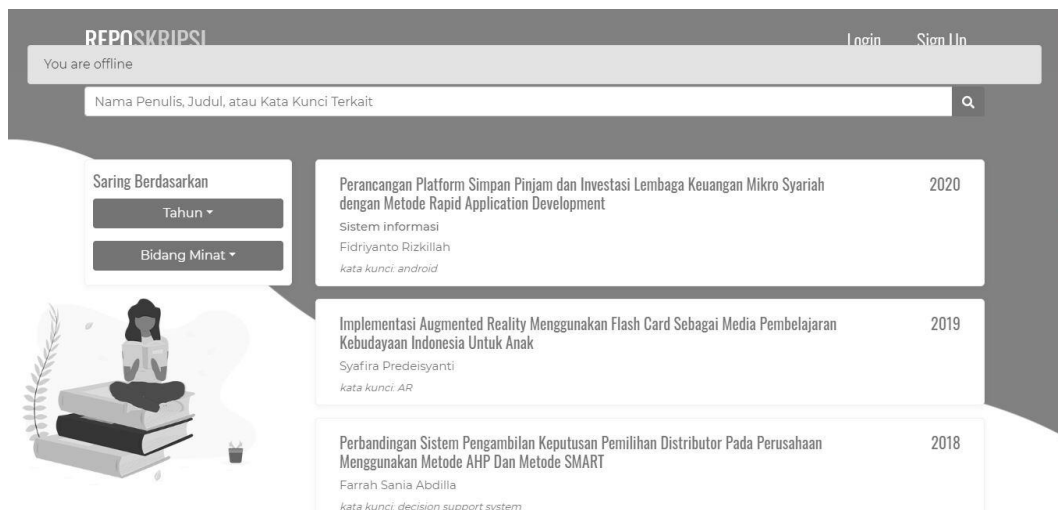
```
function checkOnline(){
  if (navigator.onLine) {
    document.body.classList.remove('offline')
  } else {
    document.body.classList.add('offline')
  }
}
checkOnline()
function handleNetworkChange(event) {
  checkOnline()
}
window.addEventListener('online', handleNetworkChange)
window.addEventListener('offline', 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 4.14. 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;
}
```



Gambar 4.14 Implementasi Fitur *Offline*

Untuk membuat fitur tambahan ke *homescreen*, manifes perlu ditambahkan kedalam program. Manifes 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"
    }
  ]
}
```

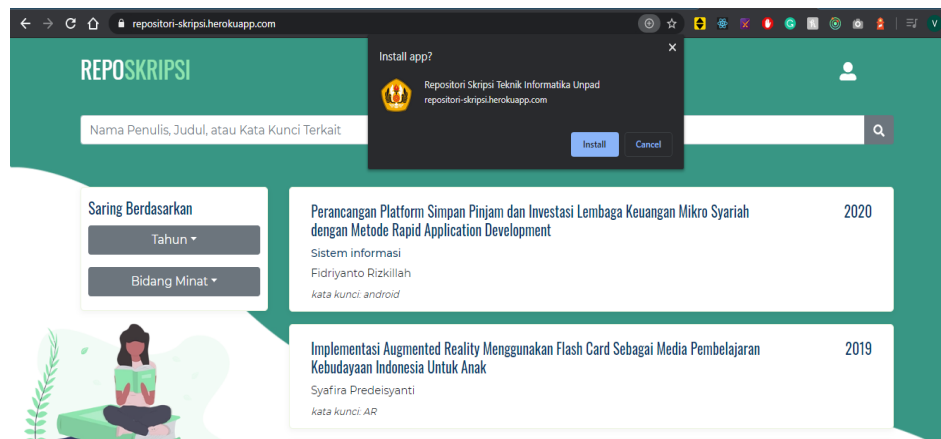


```

],
"start_url": "/",
"display": "standalone",
"theme_color": "#379683",
"background_color": "#379683"
}

```

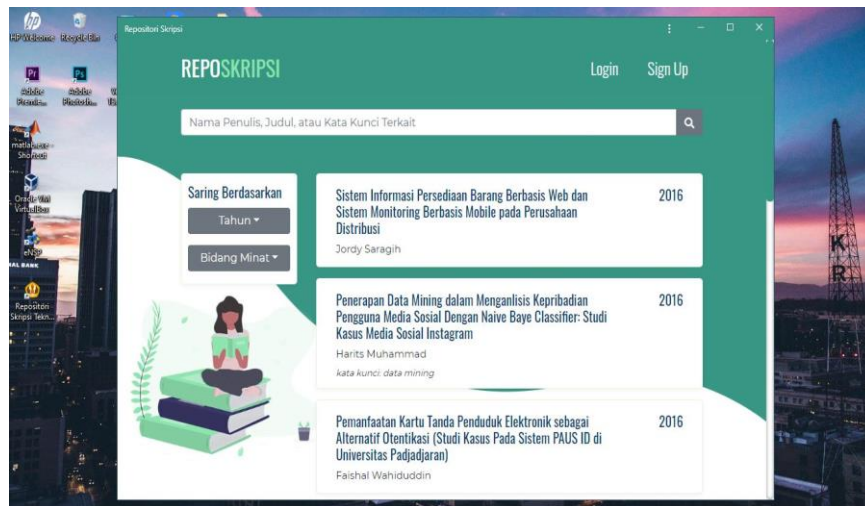
*Short\_name* dan *name* digunakan sebagai nama aplikasi, judul saat *launcher/splash screen*, dan *prompt install*. *Short\_name* digunakan apabila *name* terlalu panjang untuk ruang judul yang terbatas. *Icons* digunakan sebagai *icon app* pada *homescreen* dan gambar pada *splash screen*. *start\_url* 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 4.15.



Gambar 4.15 Implementasi fitur *add to homescreen*

Tombol tambahkan ke *homescreen* terdapat pada *address bar*. Saat di klik akan muncul pesan seperti gambar 4.15. Dengan mengklik *install*, *shortcut* akan ditambahkan pada *homescreen*. Web Repositori Skripsi Teknik Informatika Unpad

dapat diakses melalui *shortcut* tersebut. Web akan tampil dalam bentuk aplikasi seperti yang terlihat pada gambar 4.16. Fitur PWA akan diuji dengan skenario *blackbox testing* pada tabel 4.9.



Gambar 4.16 Web dalam bentuk *Native App*

Tabel 4.8 *Blackbox Testing* Fitur PWA

Skenario	Hasil yang diharapkan	Kesimpulan
Pengguna yang sebelumnya telah mengakses halaman, me- <i>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 penelitian ini, rilis kecil adalah web aplikasi dengan fitur register, login, unggah skripsi, verifikasi

akun, tinjau skripsi, detail skripsi, pencarian dan penyaringan, *edit password*, cek status skripsi, dan fitur PWA memiliki tampilan pada saat *offline* dan tombol *add to homescreen*.

Rilisan kecil diujikan kepada pengguna menggunakan *usability testing*. Web aplikasi di *deploy* menggunakan heroku, *platform cloud* sebagai layanan (PaaS), untuk memudahkan proses pengujian dan 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 Rilisan Kecil

Pertanyaan	Nilai 1	Nilai 2	Nilai 3	Nilai 4	Nilai 5	Nilai total	Nilai hasil
1	0	0	2	13	15	133	88.67
2	0	0	0	13	17	137	91.33
3	0	0	4	13	13	129	86
4	0	0	0	10	20	140	93.33
5	0	0	4	11	15	131	87.33
6	0	0	2	7	21	139	92.67
7	0	0	2	12	16	134	89.33
8	0	0	1	13	16	135	90
9	0	1	1	11	17	134	89.33
10	0	1	3	9	17	132	88
11	0	0	1	14	15	134	89.33
12	0	1	7	16	6	117	78
13	0	0	2	6	22	140	93.33
14	0	0	1	13	16	135	90
15	0	0	1	11	18	137	91.33

Pertanyaan	Nilai 1	Nilai 2	Nilai 3	Nilai 4	Nilai 5	Nilai total	Nilai hasil
16	0	0	5	14	11	126	84
17	0	0	2	11	17	135	90
18	0	0	1	11	18	137	91.33
19	0	1	0	10	19	137	91.33
20	0	0	1	12	17	136	90.67
						Total	89.267

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

Pertanyaan 12 memiliki nilai likert 78. Pertanyaan tersebut terkait tentang 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 pemeliharaan

### 4.3 Fase Pemeliharaan

#### 4.3.1 Feedback

Dari hasil pengujian kepada responden didapat feedback-feedback sebagai berikut:

1. Memperbaiki tampilan dan penulisan pada halaman profil
2. Pada saat unggah ulang, data unggahan awal tetap ada pada form
3. Menambahkan *preview* pdf pada halaman detail skripsi
4. Menghilangkan atau memindahkan tombol *search* pada kolom untuk menghindari ambiguitas

5. Menambahkan *indicator* pada menu navigasi di-*smartphone*

#### **4.3.2 Implementasi program**

Pada fase pemeliharaan kode yang diimplementasikan terkait poin-poin diatas adalah sebagai berikut

#### **4.4 Fase Akhir**

**Faseakhir berisi rilisan yang telah diupdate**

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

#### **5.2 Saran**

## DAFTAR PUSTAKA

- A.S, R. and 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 edn. 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*. Manning Publications Co.
- Handiwidjojo, W. and Ernawati<sup>2</sup>, L. (2016) 'Pengukuran Tingkat Ketergunaan (Usability) Sistem Informasi Keuangan Studi Kasus: Duta Wacana Internal Transaction (Duwit)', *JUI SI*, Vol 02(No 01).
- Hume, D. A. (2017) *Progressive Web Apps*. 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)*, Vol. 03,(No, 02).
- 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*.
- Likert, R. (1932) *A technique for the measurement of attitudes*.
- Mehta, C. *et al.* (2018) *MySQL 8 Administrator's Guide*. Birmingham Mumbai: Packt.
- Nadia, R., Ginardi, R. V. H. and Munif, A. (2018) 'Rancang Bangun Aplikasi CallTenant dengan Penyimpanan Basis Data untuk Form Dinamis Menggunakan Framework Laravel', *JURNAL TEKNIK*, 7(1).
- Prabowo, S. A., Sholiq and Muqtadiroh (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., Linz, T. and Schaefer, H. (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**