TUGAS AKHIR – KI141502

Platform *e-Learning* untuk Pembelajaran Pemrograman Web Menggunakan Konsep *Progressive Web Apps*

LAURENSIUS ADI
NRP 5113100141

Dosen Pembimbing I
Rizky Januar Akbar, S.Kom, M.Eng.

Dosen Pembimbing II
Wijayanti Nurul Khotimah, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

Platform e-Learning untuk Pembelajaran Pemrograman Web Menggunakan Konsep Progressive Web Apps

LAURENSIUS ADI
NRP 5113100141

Dosen Pembimbing I
Rizky Januar Akbar, S.Kom, M.Eng.

Dosen Pembimbing II
Wijayanti Nurul Khotimah, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

e-Learning Platform for Web Programming Learning Using Progressive Web Apps Concept

LAURENSIUS ADI
NRP 5113100141

Supervisor I
Rizky Januar Akbar, S.Kom, M.Eng.

Supervisor II
Wijayanti Nurul Khotimah, S.Kom., M.Sc.

INFORMATICS DEPARTMENT
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PLATFORM E-LEARNING UNTUK PEMBELAJARAN PEMROGRAMAN WEB MENGGUNAKAN KONSEP PROGRESSIVE WEB APPS

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Algoritma dan Pemrograman
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
LAURENSIUS ADI
NRP: 5113100141

Disetujui oleh Pembimbing Tugas Akhir:

1. Rizky Januar Akbar, S.Kom., M.Eng. .....
   NIP: 19870103 201404 1 001 (Pembimbing 1)

2. Wijayanti Nurul Khotimah, S.Kom., M.Sc. .....
   NIP: 19860312 201212 2 004 (Pembimbing 2)

SURABAYA
JUNI, 2017

[Halaman ini sengaja dikosongkan]

PLATFORM E-LEARNING UNTUK PEMBELAJARAN PEMROGRAMAN WEB MENGGUNAKAN KONSEP PROGRESSIVE WEB APPS

Nama Mahasiswa	**: LAURENSIUS ADI**
NRP	**: 5113100141**
Departemen	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Rizky Januar Akbar, S.Kom., M.Eng.**
Dosen Pembimbing 2	**: Wijayanti Nurul Khotimah, S.Kom., M.Sc.**

Abstrak

Platform *e-Learning* bisa menjadi alat bantu proses belajar yang efektif, karena peserta didik bisa belajar dengan sendiri dan dari tempat yang tidak terbatas, selama masih ada koneksi internet. Yang sering menjadi kendala adalah ketersediaan platform *e-Learning* yang stabil dalam koneksi internet yang minim atau kondisi *offline*.

Progressive Web Apps (PWA) adalah konsep pengembangan aplikasi berbasis web yang mencakup penerapan teknologi terbaru dari *browser* seperti *service workers* dan *app manifest*. Konsep PWA dapat memberikan pengalaman terbaik dalam menggunakan suatu aplikasi web walaupun dalam koneksi internet yang minim atau *offline* sekalipun dengan menggunakan *service worker*.

Service worker sebagai pengatur *request* dan *response* pada sebuah aplikasi web dapat dirancang sesuai kebutuhan. Pada tugas akhir ini, pada *service worker* digunakan strategi *caching network first, cache fallback* dengan tambahan *cache only* pada berkas-berkas statis. Strategi ini dipilih untuk menghindari eror *token* pada kerangka kerja Laravel dan tetap mendapat respons yang cepat ketika sebuah halaman web dibuka.

Modul penilaian soal pada *e-Learning* ini juga harus bisa dijalankan secara *offline* maka dibutuhkan modul penilaian yang bisa berjalan pada *browser* yaitu MochaJS. MochaJS sebagai kerangka kerja pengujian *front-end* ditambah dengan *library assertion test* jquery expect mampu memenuhi kebutuhan ini.

Penerapan konsep PWA khususnya *service worker* meningkatkan performa platform *e-Learning* terutama waktu memuat halaman menjadi lebih cepat dan dapat berjalan secara *offline*.

Kata kunci: e-Learning, Progressive Web Apps, offline, MochaJS

ELEARNING PLATFORM FOR PROGRAMMING LANGUAGE LEARNING

Student's Name	**: LAURENSIUS ADI**
Student's ID	**: 5113100141**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Rizky Januar Akbar, S.Kom., M.Eng.**
Second Advisor	**: Wijayanti Nurul Khotimah, S.Kom., M.Sc.**

Abstract

E-learning platform could be a powerful addition to learning process because students can study on their own from anywhere as long as internet connection is available. The lack of e-Learning platform that can fulfill the need for the platform to work on flaky connection or offline network is the main problem.

Progressive Web Apps is a new concept of web application development which includes newest technology of web browsers such as service workers and app manifest. PWA concept can give the best user experience on any web app on flaky internet connection or even offline by using service worker.

Service worker as request and response handler on a web app can be crafted as needed. In this final project, network first, cache fallback caching strategy is used on the service worker with additional cache only strategy on static files. This kind of strategy is chosen to avoid any token errors with Laravel framework and keeping fast response each time a webpage is opened.

Judging module in this e-Learning also needed to run on offline network, so MochaJS is chosen. MochaJS as a front-end testing framework, alongside jquery.expect as its assertion test library, able to fulfill the need.

The application of PWA concept on this e-Learning enhance its performance especially on making pages loads faster and function well while offline.

Keywords: e-Learning, Progressive Web Apps, offline, MochaJS

KATA PENGANTAR

Puji Tuhan penulis ucapkan atas selesainya Tugas Akhir yang berjudul "Platform e-Learning untuk Pembelajaran Pemrograman Web Menggunakan Konsep Progressive Web Apps"

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis, karena dengan ini penulis dapat belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan selama menempuh perkuliahan di Teknik Informatika ITS.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Kedua orang tua penulis yang senantiasa memberikan dukungan dan doanya.
2. Bapak Rizky Januar Akbar, S.Kom., M.Eng. selaku pembimbing I yang selalu membimbing dan memotivasi penulis selama pengerjaan Tugas Akhir.
3. Wijayanti Nurul Khotimah, S.Kom., M.Sc. selaku pembimbing II yang selalu membimbing dan memotivasi penulis selama pengerjaan Tugas Akhir.
4. Teman-teman ITS EXPO 2014, 2015, dan khususnya 2016 yang selalu menghibur dan memberikan banyak kenangan dan pelajaran berharga.
5. Teman-teman KMK ITS 2013 dan KBC yang selalu menghibur, memberi dukungan dan kenangan selama perkuliahan di ITS.
6. Teman-teman Angkatan 2013 yang selalu memberi dukungan dan kenangan selama perkuliahan di Teknik Informatika ITS.
7. Serta semua pihak yang turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Surabaya, Juni 2017

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

E-learning sebagai salah satu metode pembelajaran di perkuliahan yang sangat potensial. Tak jarang banyak instansi pendidikan yang menggunakan platform *e-Learning* sebagai media belajar para mahasiswanya. Apabila dipergunakan dengan sepenuhnya, *e-Learning* bisa menjadi alat bantu proses belajar yang efektif, karena peserta didik bisa belajar dengan sendiri dan dari tempat yang tidak terbatas, selama masih ada koneksi internet.

Yang sering menjadi kendala adalah ketersediaan platform *e-Learning* yang memenuhi kebutuhan dan stabil walaupun dalam koneksi internet yang minim atau kondisi offline sekalipun sehingga masih dapat menikmati layanan *e-Learning* dengan baik.

Progressive Web Apps (PWA) adalah penamaan untuk konsep baru yang dikemukakan oleh Alex Russell dan Frances Berriman pada tahun 2015 [1]. Konsep ini mencakup penerapan teknologi baru dari web *browser* seperti *service worker*s dan app manifest. PWA memiliki karakteristik utama dapat diandalkan (*reliable*), cepat (*fast*), dan menarik (*engaging*), memastikan pengguna mendapatkan pengalaman terbaik dalam menggunakan suatu aplikasi web walaupun dalam koneksi internet yang minim atau *offline* sekalipun.

Dalam tugas akhir yang penulis ajukan ini, akan dibuat platform *e-Learning* yang dapat membantu proses pembelajaran di Teknik Infromatika ITS mata kuliah Pemrograman Web, dengan *preview* hasil pekerjaan dan bantuan pengecekan secara instan, serta mampu berjalan walaupun dalam kondisi *offline* menggunakan *service worker* yang diimplementasikan melalui konsep PWA, dan menggunakan bantuan *parser*.

1.2 Rumusan Masalah

Rumusan masalah yang terdapat pada tugas akhir ini, antara lain adalah:

1. Bagaimana menerapkan konsep Progressive Web Apps pada platform *e-Learning* yang sudah ada?
2. Bagaimana membuat *code editor* masih bisa digunakan dalam kondisi *offline* menggunakan *service worker*s?
3. Bagaimana membuat antarmuka dan *user experience* yang baik pada *e-Learning* dengan penerapan Progressive Web Apps?
4. Bagaimana membuat modul penilaian yang bisa berjalan secara *offline*?

1.3 Batasan Masalah

Permasalahan Batasan masalah yang terdapat pada tugas akhir ini, yaitu sebagai berikut:

1. Platform *e-Learning* Pemrograman Web dibangun dengan melanjutkan dari platform *e-Learning* yang sudah ada pada "Platform *e-Learning* untuk Pembelajaran Bahasa Pemrograman".
2. Platform *e-Learning* ini digunakan untuk keperluan mata kuliah Pemrograman Web
3. Platform *e-Learning* ini dikhususkan mendukung bahasa pemrograman HTML, CSS, JavaScript.
4. Teknologi yang digunakan dalam pembuatan aplikasi ini adalah PHP, JavaScript, HTML, CSS, CodeMirror, CodeMirror Grammar, MochaJS, dan jquery expect.
5. Pembangunan aplikasi dilakukan menggunakan bantuan Chrome DevTools pada Chrome 52 atau lebih baru.
6. Pembuatan kunci jawaban harus sesuai dengan ketentuan *library* jquery.expect, sehingga pengguna yang membuat soal dank unci jawaban harus memahami *library* tersebut terlebih dahulu.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini yaitu membangun *e-Learning* untuk pembelajaran mata kuliah Pemrograman Web dengan konsep Progressive Web Apps (PWA). Penerapan konsep

PWA dapat mendukung aktifitas pengguna dengan kondisi internet mati (*offline*) dan dapat diandalkan dalam koneksi internet yang minim.

1.5 Manfaat

Tugas akhir ini diharapkan dapat menjadi platform *e-Learning* yang mendukung proses pembelajaran mata kuliah Pemrograman Web. Tugas akhir ini menerapkan konsep Progressive Web Apps sehingga lebih dapat diandalkan dan mudah digunakan dalam kondisi *offline* sekalipun.

1.6 Metodologi

Ada beberapa tahap dalam proses pengerjaan tugas akhir ini. Berikut ini adalah tahap-tahap dalam pembuatan tugas akhir.

1. **Studi Literatur**

   Tahap ini membahas pengumpulan dan literatur yang diperlukan dalam proses perancangan dan implementasi aplikasi yang dibangun. Literatur yang digunakan adalah sebagai berikut:

   a. Progressive Web Apps'
   b. *Cache*
   c. *Service worker*
   d. App Shell
   e. App Manifest
   f. Editor Codemirror
   g. Laravel
   h. MaterializeCSS
   i. PostgreSQL
   j. MochaJS
   k. Jquery.expect

2. **Perancangan dan Desain Sistem**

   Pada tahap ini dilakukan perancangan sistem dengan menggunakan studi literatur dan mempelajari konsep aplikasi yang akan dibuat. Dengan berbekal teori, metode, dan informasi yang sudah terkumpul pada tahap sebelumnya diharapkan dapat membantu dalam proses perancangan sistem. Tahap ini merupakan tahap yang paling penting pada bentuk awal atau *prototype* yang akan diimplementasikan.

3. **Implementasi**

   Pada tahap ini dilakukan implementasi rancangan sistem yang telah dibuat. Tahapan ini merealisasikan apa yang terdapat pada tahapan sebelumnya sehingga menjadi sebuah aplikasi dengan simulasi yang digunakan sesuai dengan apa yang telah direncanakan.

4. **Uji Coba dan Evaluasi**

   Pada tahap ini aplikasi yang telah selesai dibuat akan diuji. Pengujian dan evaluasi akan dilakukan dengan menggunakan *tools* uji coba Progressive Web Apps bernama Lighthouse, untuk melihat berapa persen aplikasi memenuhi kriteria PWA. Pengujian juga dilakukan dengan menerapkan kondisi *online* dan *offline* lalu melihat waktu respons pada Chrome DevTools.

5. **Penyusunan Laporan Tugas Akhir**

   Pada tahap ini disusun laporan tugas akhir sebagai dokumentasi pelaksanaan tugas akhir yang mencakup seluruh konsep, teori, implementasi, dan hasil yang telah dikerjakan.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II Tinjauan Pustaka

Bab ini membahas beberapa pustaka penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, proses dan perancangan antarmuka pada aplikasi.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan perangkat lunak.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dengan metode pengujian berdasarkan pada kasus penggunaan dan kebutuhan fungsional sistem berdasarkan skenario yang disusun oleh penulis.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi kode sumber berupa kelas-kelas yang digunakan pada saat implementasi sistem.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan literatur yang menjadi dasar pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Progressive Web Apps

Progressive Web Apps (PWA) adalah konsep pengalaman pengguna yang mengabungkan bagian terbaik web dan bagian terbaik *native apps*. PWA berguna bagi pengguna sejak pertama membuka halaman sebuah web dengan konsep PWA, dan seiring dengan pengguna menggunakan aplikasi web lebih banyak lagi, aplikasi akan menjadi semakin *powerful*. Aplikasi dapat dimuat dengan cepat, bahkan dalam kondisi internet yang kurang baik, bisa mengirim *push notifications*, punya ikon aplikasi di *home screen*, dan bisa berjalan dalam mode layar penuh [1].

Sebuah aplikasi PWA memiliki karakteristik sebagai berikut:

1. *Progressive*: berjalan dengan baik bagi semua pengguna, tidak memandang *browser* yang dipakai, karena dibangun dengan *progressive enhancement* dari sebuah inti aplikasi.
2. *Responsive*: ditampilkan dengan baik pada semua perangkat berbagai ukuran: *desktop*, *tablet*, *mobile*, dan apapun perangkat baru selanjutnya.
3. *Connectivity independent*: dengan *service worker* aplikasi dapat bekerja dalam kondisi *offline* atau jaringan yang lemah.
4. *App-like*: terasa seperti sebuah aplikasi, karena model *app shell* yang memisahkan fungsionalitas aplikasi dari kontennya.
5. *Fresh*: selalu *up-to-date* berkat pembaruan melalui *service worker*.

6. *Safe*: aplikasi dilayani via HTTPS untuk mencegah *snooping* dan memastikan konten tidak diubah.
7. *Discoverable*: dapat diidentifikasi sebagai "*application*" karena *manifest* W3C dan *service worker registration scope*, dan memungkinkan mesin pencari untuk menemukannya.
8. *Re-engageable*: memudahkan dalam mengajak pengguna untuk menggunakan ulang aplikasi melalui fitur seperti *push notifications*.
9. *Installable*: memungkinkan pengguna untuk menambahkan aplikasi ke *homescreen* tanpa harus kerepotan dengan *app store*.
10. *Linkable*: dengan mudah dapat membagikan aplikasi dengan membagikan URL, dan tidak memerlukan instalasi yang rumit.

PWA sepenuhnya mengandalkan *browser* pengguna dan teknologi yang ada didalamnya. Sampai saat tugas akhir ini ditulis, sudah ada 73.61% dari seluruh *browser* di seluruh dunia yang mendukung fitur *service worker*, seperti Mozilla Firefox, Google Chrome, Chrome for Android dan Opera, sementara Edge dan Safari belum mendukung fitur ini [2].

Ada tiga jenis arsitektur PWA yaitu *Server-Side Rendering* (SSR), *Client-Side Rendering* (CSR), dan gabungan keduanya. Arsitektur yang lazim digunakan sampai saat ini adalah SSR, dimana *browser* mengambil data halaman melalui HTTP/HTTPS dan data dikembalikan berupa halaman lengkap dengan data dinamis sudah di-*render* [3].

SSR bagus digunakan karena memberikan *render* pertama dengan cepat, tetapi memuat ulang sebuah halaman SSR berarti membuang seluruh data, dan akan memakan waktu ketika memuat kembali seluruh *resource* di halaman. SSR juga merupakan teknik yang sudah matang dengan banyak *tool* yang mendukung. SSR juga berjalan di semua *browser* tanpa harus memikirkan perbedaan implementasi JavaScript antar *browser*.

Arsitektur CSR adalah ketika JavaScript berjalan di *browser* untuk memanipulasi DOM (*Document Object Model*).

Keuntungan dari CSR adalah memberikan umpan balik langsung ketika pengguna menekan tombol navigasi, ketimbang harus menunggu server untuk mengetahui informasi apa yang harus ditampilkan. CSR bisa memutuskan apakah harus memuat ulang seluruh halaman atau sebagian halaman saja ketika ada data baru yang masuk.

Best practice yang disarankan adalah menggunakan gabungan keduanya, jadi *render* pertama dilakukan di server dan menggunakan data dari server. Lalu ketika klien menerima halaman tersebut, *service worker* menyimpan semua yang dibutuhkan untuk *app shell* ke dalam *cache*.

2.2 Cache

Cache interface menyediakan mekanisme penyimpanan untuk pasangan objek *Request* dan *Response* mau disimpan ke dalam *cache*, contohnya sebagai bagian dari daur hidup *service worker*. Perlu diketahui bahwa *cache interface* terbuka terhadap halaman web dan juga *workers*. *Cache* tidak harus selalu digunakan bersamaan dengan *service worker* walaupun *cache* tercantum di dalam spesifikasinya [4].

Cache digambarkan sebagai sebuah *array* berisi objek *Request* yang bertindak sebagai pasangan untuk responsnya yang disimpan di dalam *browser* [5]. Untuk menambahkan ke dalam *cache* menggunakan metode add dan addAll, dengan *string* dari URL yang akan disimpan atau objek *Request* sebagai parameternya seperti pada Kode Sumber 2.1. *Cache* akan meminta data ke server dan menyimpan responsnya ke dalam *cache*.

```
caches.open('test-cache').then(function(cache) {
  cache.add('/page/1');
});
```

Kode Sumber 2.1 Metode Cache Add

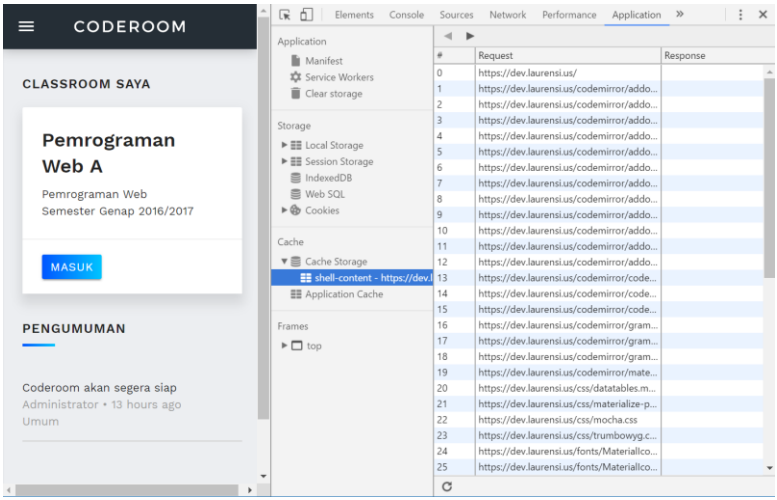
Selain metode add dan addAll, ada juga metode yang mirip yaitu metode put dengan parameter URL dan objek *response*

seperti yang ditunjukkan pada Kode Sumber 2.2. Perbedaannya dengan metode add adalah data tidak dimintakan ulang ke server, tetapi sudah dalam bentuk objek *response* karena sudah dimintakan sebelumnya pada bagian *fetch*.

```
fetch('/page/1').then(function(response) {
return caches.open('test-cache').then(function(cache) {
    return cache.put('/page/1', response);
  });
});
```

Kode Sumber 2.2 Metode Cache Put

Kita bisa melihat daftar *cache* yang disimpan oleh *browser*, dalam hal ini Chrome, dengan menyalakan DevTools pada Chrome, lalu pindah ke *tab Application* dan lihat pada bagian *Cache Storage* seperti yang ditunjukkan pada Gambar 2.1.

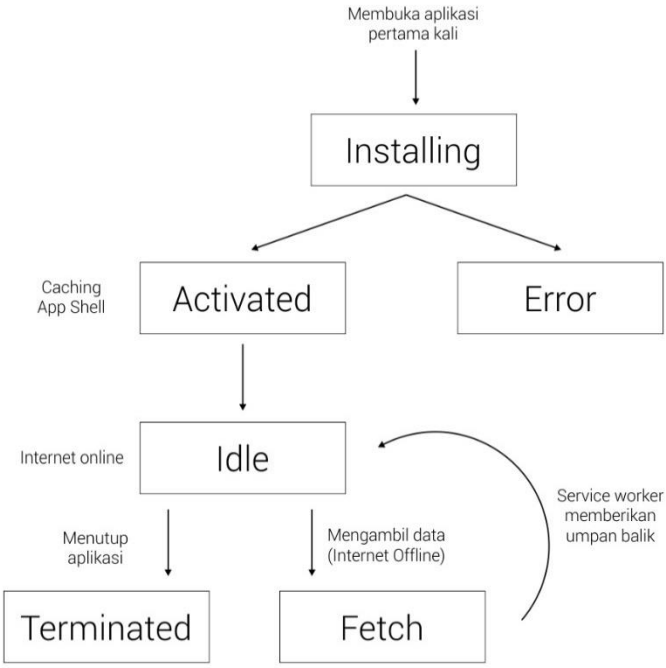


Gambar 2.1 Cache Storage pada Chrome

2.3 Service Worker

Service worker adalah salah satu jenis dari *web worker*, yaitu script yang berjalan di belakang *browser* pengguna. *Service worker* pada dasarnya adalah berkas JavaScript yang berjalan pada *thread* yang berbeda dengan *main thread browser*, menangani *network request*, *caching*, dan mengembalikan *resource* dari *cache*, dan bisa mengirimkan *push message*.

Aset web dapat disimpan sebagai *cache* lokal, sehingga dengan jaringan internet yang kurang memadai pun, pengguna tetap mendapat pengalaman yang baik. Aplikasi dapat tetap menjalankan halaman web yang sudah di-*cache* atau memberikan status koneksi tanpa *browser* menampilkan tulisan eror karena ketiadaan koneksi internet.



Gambar 2.2 Daur Hidup Service Worker

Service worker memiliki daur hidup (*life cycle*) yang berbeda dengan halaman web seperti digambarkan pada Gambar 2.2. Untuk memasang *service worker* ke laman, kita butuh meregistrasinya dengan menggunakan JavaScript yang ada di halaman web. Setelah diregistrasi, *browser* akan memulai tahap *install service worker* di latar. Setelah aktif, *service worker* akan menangani semua halaman di bawah *scope* dimana *service worker* di-*install*, lalu akan ada 2 kemungkinan *state*: *terminated* untuk menghemat memori, atau menangani *fetch* ketika ada *network request* dari halaman web.

Service worker bisa menangani berbagai jenis *request*, tetapi yang bisa disimpan kedalam *cache* hanya semua *request* jenis GET. Pada *service worker*, bisa aplikasikan strategi *caching* sesuai keinginan kita, namun tidak ada satu strategi terbaik untuk *caching* konten dinamis, dan ada banyak situasi yang bisa mempengaruhi strategi *caching* [6].

Ada 5 strategi *caching* yang umum digunakan pada *service worker*, yaitu:

- *Cache first, network fallback*: *service worker* pertama-tama memuat ke halaman berkas HTML dan JavaScript yang ada di dalam *cache*, jika memungkinkan, melewati *network*. Jika konten *cache* tidak tersedia, maka *service worker* mengembalikan respons dari *network* dan menyimpan respons ke *cache*. Strategi ini dipakai untuk menyimpan *resource* yang hampir tidak berubah, seperti gambar.
- *Network first, cache fallback*: pertama-tama mengecek apakah *network* memberikan respons, dan jika berhasil, kembalikan data sekarang ke halaman. Jika gagal, *service worker* mengembalikan data dari *cache*. Strategi ini dipakai ketika membutuhkan data yang selalu baru seperti respons API, tapi butuh menampilkan sesuatu ketika *network* tidak bisa dicapai.
- *Cache/network race*: Satu buah *request* yang sama dijalankan bersamaan ke *network* dan ke *cache*. Pada

kebanyakan kasus, data dari *cache* dimuat terlebih dahulu dan langsung ditampilkan ke halaman. Sementara respons dari *network* digunakan untuk memperbarui data di *cache*. Proses ini terjadi di latar sehingga tidak menggangu *rendering* konten dari *cache.* Strategi ini digunakan ketika konten sering diperbarui secara berkala, seperti artikel, media sosial, *timeline* dan papan skor game.

- *Network only*: hanya mengecek *network*, tidak ada data disimpan ke dalam *cache*. Ketika jaringan gagal, maka *request* gagal. Strategi ini dipakai ketika hanya data terbaru yang bisa ditampilkan di halaman.
- *Cache only*: data yang ditampilkan hanya data yang ditambahkan ke *cache* saat instalasi *service worker* berlangsung. Strategi ini dipakai ketika hanya menampilkan data statis di halaman.

2.4 App Shell

Menggunakan arsitektur *app shell* adalah salah satu cara untuk membuat PWA yang memuat konten dengan *reliable* dan instan pada layar pengguna, sama seperti yang kita lihat pada aplikasi *native*. *App shell* merujuk pada *resources* lokal yang dibutuhkan oleh aplikasi web untuk memuat kerangka antarmuka aplikasi. Sebuah *app shell* selalu memuat di dalamnya HTML, terkadang dengan JavaScript dan CSS, dan mungkin juga dengan *static resource* lainnya yang merupakan bagian struktur halaman. Namun, konten sebenarnya yang spesifik terhadap suatu halaman tidak ikut disimpan. Dengan kata lain, *app shell* mencakup bagian aplikasi yang tidak atau jarang berubah dan bisa disimpan ke *cache*, sehingga dapat langsung dimuat ke halaman saat membuka halaman pada kunjungan selanjutnya [3].

Sebuah *app shell* harus dapat dimuat dengan cepat, menggunakan data sesedikit mungkin, menggunakan asset yang tidak berubah dari *cache* lokal, memisahkan konten dan navigasi, bisa mengambil dan menampilkan konten khusus halaman

(HTML, JSON, dan lainya), dan secara opsional *cache* konten dinamis.

2.5 App Manifest

App manifest berisi informasi tentang aplikasi, seperti nama, pemilik, ikon, dan deskripsi, dalam bentuk file teks. Manifest memperbolehkan pengguna untuk menginstal aplikasi web ke *homescreen* sebuah smartphone Android, sehingga memberikan pengguna akses yang lebih cepat dan pengalaman yang lebih baik. App Manifest biasanya disimpan dalam file manifest.json seperti yang ditunjukkan pada Kode Sumber 2.3.

```
{
  "short_name": "App shell",
  "name": "App shell",
  "start_url": "/index.html",
  "icons": [{
        "src": "images/icon-128x128.png",
        "sizes": "128x128",
        "type": "image/png"
    }],
  "display": "standalone",
  "orientation": "portrait",
```

Kode Sumber 2.3 App Manifest

2.6 Editor CodeMirror

CodeMirror [7] adalah *text editor* yang dimplementasikan menggunakan JavaScript untuk *browser*. CodeMirror dikhususkan untuk menyunting kode, dan memiliki banyak mode bahasa pemrograman dan *addons*, serta memiliki API *programming* dan sistem tema CSS yang memungkinkan pengguna mengkostumisasi CodeMirror agar sesuai kebutuhan dan menambahkan fungsionalitas baru.

CodeMirror sangat mudah untuk digunakan, hanya perlu menambahkan *file* JavaScript dan CSS ke dalam document HTML lalu menjalankan fungsi JavaScript seperti yang ditunjukkan pada Kode Sumber 2.4. CodeMirror dipanggil dengan menyertakan

berkas JavaScript codemirror.js agar variable objek CodeMirror bisa dipanggil, lalu menggunakan fungsi fromTextArea dan *selector* pada *textarea* yang mau dijadikan *code editor*.

```
<script src="lib/codemirror.js"></script>
<link rel="stylesheet" href="lib/codemirror.css">
<script>
var myCodeMirror = CodeMirror.fromTextArea(
   document.querySelector("#code textarea") );
</script>
```

Kode Sumber 2.4 Penggunaan CodeMirror

Dalam tugas akhir ini, akan digunakan berbagai *addons* yang ada pada CodeMirror yaitu:

- lint/lint.js: sebagai antarmuka untuk *annotation* akan kesalahan penulisan kode pada *editor*.
- fold/foldcode.js: digunakan untuk menutup atau membuka kembali tampilan *tag* HTML yang sejenis.
- fold/foldgutter.js: digunakan bersama dengan *foldcode* untuk menampilkan tanda panah ke bawah yang berada disamping nomor baris kode, ketika ditekan, tombol akan mengaktifkan fungsi *foldcode*.
- mode/htmlmixed.js, css.js, atau javascript.js: digunakan untuk menentukan mode yang dipakai pada *editor*, pemilihan mode ini berpengaruh pada pewarnaan kode sumber

2.7 Laravel

Laravel adalah aplikasi *open source* yang berupa *framework* dengan model MVC (*Model, View, Controller*) untuk membangun halaman web dinamis dengan menggunakan PHP. Laravel memudahkan *developer* untuk membuat aplikasi web dengan cepat dan mudah dibandingkan dengan membuatnya dari awal. *Framework* itu sendiri adalah suatu kerangka kerja yang berupa sekumpulan folder yang memuat berkas PHP yang menyediakan

class libraries, *helpers*, *plugins*, dan lainnya. *Framework* menyediakan konfigurasi dan teknik *coding* tertentu.

2.8 MaterializeCSS

Materialize CSS [8] adalah *framework front-end* yang terdiri dari pustaka-pustaka CSS dan JS yang sudah didesain mengikuti prinsip desain Material milik Google [9]. *Framework* ini memiliki komponen dan animasi untuk memberikan umpan balik visual kepada pengguna. Sistem *responsive* yang baik juga mendukung tampilan yang optimal untuk segala ukuran layar, untuk pengalaman pengguna yang konsisten.

2.9 PostgreSQL

PostgreSQL [10] adalah suatu basis data *management system* yang cukup dikenal di kalangan *developer*. Selain karena *open source*, DBMS ini dianggap memiliki sistem yang kuat. DBMS ini telah dikembangkan selama 15 tahun dan telah digunakan oleh beberapa perusahaan raksasa perangkat lunak di dunia. PostgreSQL juga bisa digunakan di semua sistem operasi besar yang ada.

2.10 MochaJS

MochaJS [11] adalah kerangka kerja tes JavaScript yang berjalan pada Node.js dan pada *browser*, yang bertujuan memudahkan testing *asynchronous*. Tes pada Mocha berjalan secara berurutan, fleksibel dan dengan hasil laporan yang akurat, juga dengan pemetaan *uncaught exception* pada tes yang sesuai.

Untuk melakukan tes, yang pertama kali kita lakukan yaitu menulis kerangka seperti pada Kode Sumber 2.5. Fungsi describe digunakan untuk pengelompokan test, sedangkan it merepresentasikan satu *test case*, lalu di dalam fungsi it ditambahkan fungsi *assertion* dari *library* seperti expect.js.

```
describe('Challenges', function () {
   it('Should have a div', function () {
        // assertion test here
   });
});
```

Kode Sumber 2.5 Kerangka Mocha Test

Untuk menjalankan seluruh tes mocha, perlu dilakukan *setup* lalu memanggil fungsi *run* seperti pada Kode Sumber 2.6.

```
mocha.setup('bdd');
window.DONE = false;
mocha.run(function () {
   window.DONE = true;
});
```

Kode Sumber 2.6 Menjalankan Testing Mocha

2.11 Jquery.expect

Jquery.expect atau $expect [12] adalah DOM (*Document Object Model*) *assertion library* sederhana yang dibangun di atas jQuery dan terinspirasi dari *library* expect.js milik LearnBoost. Expect dibangun untuk melakukan *assertion* dengan bahasa sederhana dan menggunakan *selector* milik jQuery. Pada Kode Sumber 2.7 terdapat contoh penggunaan *assertion* jquery.expect dengan beberapa fungsi yang ada.

```
$expect('div').to.exist();
$expect('li').to.be.lessThan(5);
$expect('input').to.be('[type=text]');
$expect('.link-1').to.have.text('Codecademy');
$expect('div.container').to.have.siblings('.pane').that
.has.css('float', 'left');
```

Kode Sumber 2.7 Contoh Assertion Expect

2.12 Aplikasi Serupa

Aplikasi yang dibangun dalam tugas akhir ini tergolong jenis aplikasi web kursus *online* interaktif dengan materi *web development*. Ada banyak aplikasi serupa yang sudah ada antara lain Codecademy [13] dan FreeCodeCamp [14].

Kedua aplikasi ini sama-sama menyediakan berbagai materi *web development* dengan gratis, cukup dengan mendaftarkan akun pada web tersebut, hanya saja pada Codecademy tersedia fitur premium dimana pengguna bisa membayar untuk mendapatkan fitur lebih, sedangkan FreeCodeCamp tidak berbayar sama sekali dan merupakan aplikasi *open source*.

Dari sisi arsitektur, aplikasi pada tugas akhir ini berbeda dalam hal kerangka kerja *back-end* yang digunakan, kedua aplikasi tersebut menggunakan JavaScript. Pada sisi antarmuka memiliki kemiripan dimana ada deskripsi dan perintah pada halaman menjawab soal berada di sisi kiri, *code editor* pada bagian tengah, dan tampilan *browser* pada sebelah kanan.

Dari sisi dukungan untuk mengerjakan tugas secara *offline*, pada FreeCodeCamp ketika sebuah soal sudah dibuka, penilaian bisa dijalankan secara *offline*, tetapi tidak bisa berpindah ke halaman lain, sedangkan pada Codecademy, pengguna diharuskan untuk selalu terkoneksi dengan internet, dan ketika *offline* penilaian tidak bisa dijalankan dengan indikasi bertuliskan *'Lost Connection to Codecademy'*.

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini akan dijelaskan hal-hal yang berkaitan dengan perancangan sistem yang akan dibuat dalam tugas akhir ini, dimulai dari deskripsi umum mengenai perangkat lunak yang akan dibuat, perancangan proses-proses yang ada, dan arsitektur umum sistem.

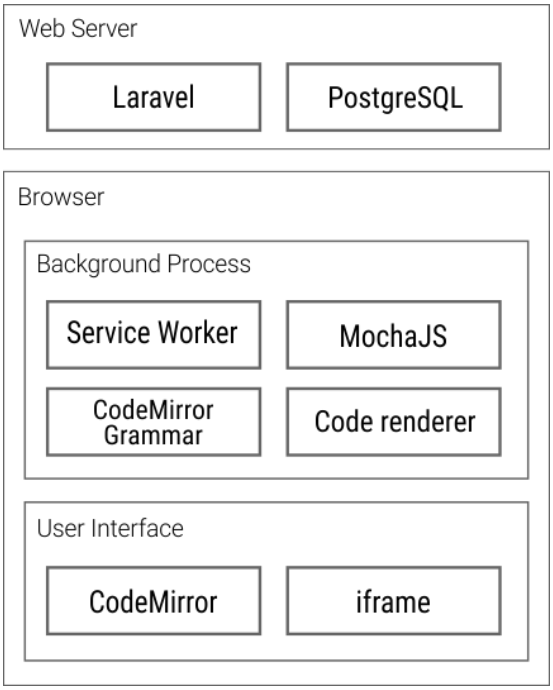
3.1 Deskripsi Umum

Dalam tugas akhir ini dibangun sebuah perangkat lunak berbasis web untuk pembelajaran Pemrograman Web dengan konsep Progressive Web Apps (PWA). Aplikasi dijalankan melalui *browser* pengguna. Di dalam aplikasi ini pengguna dapat melakukan pembelajaran interaktif dengan menulis langsung pada *text editor* dan dapat melihat hasilnya dalam bentuk tampilan serupa *browser*.

Tugas akhir ini berfokus pada pengembangan *classroom management* yang sudah ada dan konsep PWA untuk mendukung pembelajaran *offline*. Yang dimaksudkan *classroom management* adalah pengaturan mahasiswa dan dosen kedalam *classroom* lalu memungkinkan dosen untuk membuat soal dan mahasiswa bisa menjawab soal tersebut. Untuk permasalahan lain seperti pengelolaan pengguna, pengelolaan hak akses, dan pengelolaan data master, tidak akan dibahas secara rinci dalam buku tugas akhir ini.

3.2 Arsitektur Sistem

Secara garis besar, sistem dibagi menjadi 3 bagian yaitu web server, *background process* dan antarmuka pada *browser* seperti yang digambarkan pada Gambar 3.1.



Gambar 3.1 Diagram Arsitektur Sistem

Semua komponen seperti pada Gambar 3.1 memiliki peranan masing-masing. Berikut adalah penjelasan serta fungsi komponen-komponen yang ada di dalam arsitektur sistem *e-Learning*:

- Web server
  - Laravel: *framework* PHP dengan model MVC, sebagai penghubung basis data, digunakan sebagai kerangka kerja *server side rendering*
  - PostgreSQL sebagai tempat penyimpanan data, yang dibagi menjadi 2 yaitu basis data *access control* dan basis data *e-Learning*

- *Background process* di *browser*
  - *service worker*: *web worker* yang bekerja sebagai perantara *event request* dan respons antara aplikasi dan web server, serta *cache*
  - MochaJS: *testing framework* untuk melakukan proses penilaian *front-end* di dalam iframe dengan bantuan jquery.expect.
  - CodeMirror Grammar: kumpulan *script* yang memberikan umpan balik kepada input pengguna berupa kesalahan penulisan yang tidak sesuai pada *editor* CodeMirror di *browser*
  - Code renderer: bertugas mengambil *script* yang dituliskan oleh pengguna pada CodeMirror dan menampilkannya di dalam iframe
- User interface:
  - CodeMirror: sebagai *code editor*, tempat pengguna menulis HTML, CSS, JS
  - Iframe: komponen HTML untuk menampilkan dokumen HTML di dalam dokumen HTML, dalam hal ini dokumen hasil code renderer

3.3 Perancangan Kemampuan *Offline*

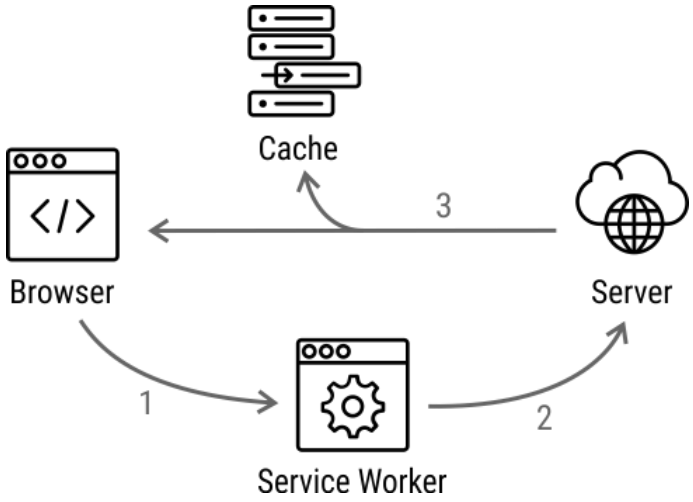
Untuk memenuhi konsep *Progressive Web Apps* dari aplikasi elearning ini, maka diperlukan kebutuhan non-fungsional yaitu kemampuan aplikasi untuk berjalan ketika internet sedang dalam kondisi offline atau jaringan lamban.

Ada 3 komponen utama agar semua aplikasi web bisa dikatakan sebagai PWA yaitu: *app shell*, *app manifest*, dan *service worker*. Disini *service worker* memiliki peranan paling banyak di dalam konsep PWA. *Service worker* bekerja sebagai pengatur *event fetch* dari *browser*, lalu *service worker* memutuskan apakah *request* akan diteruskan ke server atau ke *cache* berdasarkan kondisi *network online* atau *offline*.

Dari banyak strategi *caching service worker* yang bisa dipakai sesuai kebutuhan seperti yang telah dijelaskan pada Subbab 2.3, pada tugas akhir ini diimplementasikan strategi *network first, cache fallback.*

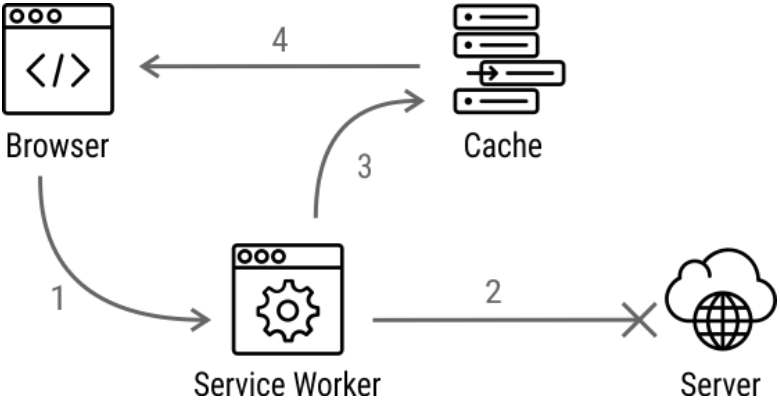
S*ervice worker* akan meneruskan setiap *request* berjenis *GET* dari halaman web ke server dalam kondisi *online*, lalu menduplikasi respons server dan disimpan ke dalam *cache* di *browser*, lalu respons server diteruskan kembali ke halaman web seperti digambarkan pada Gambar 3.2. Setiap halaman yang pernah dikunjungi oleh pengguna secara otomatis akan tersedia dalam versi *offline*-nya, hanya saja pengguna tidak bisa mengirim data ke server.

Ketika kondisi *offline* atau tidak ada respons dari server, maka akan dikembalikan halaman dari *cache* jika halaman tersebut sudah ada di dalam *cache* sebelumnya seperti digambarkan pada Gambar 3.3.



Gambar 3.2 Service Worker Kondisi Online

Strategi *network first* dalam kondisi online memastikan selalu data yang terbaru dari server ditampilkan ke aplikasi dan menghindari kesalahan CSRF token Laravel pada *form*. Strategi ini memiliki kelemahan ketika pengguna mengalami koneksi jaringan yang lamban dan harus menunggu *request* gagal mencapai server terlebih dahulu baru pengguna mendapat respons dari *cache fallback*.



Gambar 3.3 Service Worker Kondisi Offline

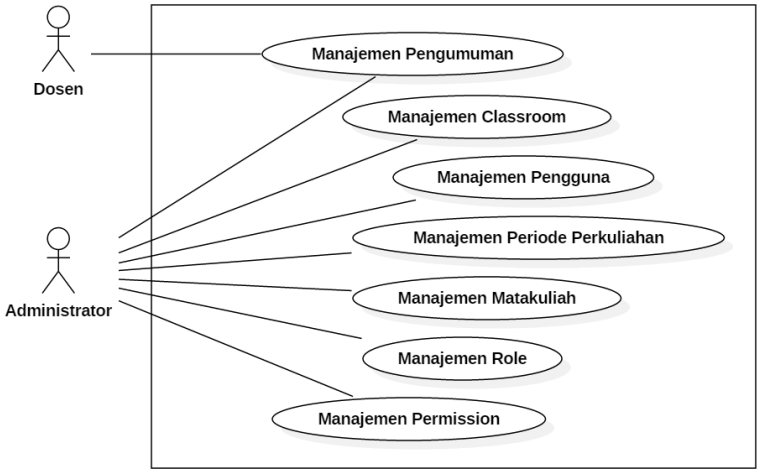
3.4 Perancangan Kasus Penggunaan

Kasus penggunaan mewakili kebutuhan fungsional sistem. Perancangan diagram kasus penggunaan ini dibagi berdasarkan pengelompokan tugas-tugas dari pengguna yang ada. Pengguna dibagi berdasarkan role administrator, dosen, dan mahasiswa. Administator bertugas untuk melakukan manajemen-manajemen yang mendukung sistem dan proses pembelajaran diluar kelas. Sedangkan dosen dan mahasiswa, yang merupakan pusat dari segala tujuan diadakannya aplikasi ini, melakukan kegiatan-kegiatan yang berhubungan dengan proses pembelajaran. Oleh karena itu, berkaitan dengan proses pembelajaran yang ditujukan ke mahasiswa, kasus penggunaan dibagi menjadi kasus

penggunaan non-pembelajaran dan kasus penggunaan pembelajaran.

3.4.1 Perancangan Kasus Penggunaan Non-Pembelajaran

Kasus penggunaan ini tidak melibatkan mahasiswa dalam alur prosesnya, dan dititikberatkan pada aktor Administrator, sedangkan dosen hanya mendapat manajemen pengumuman. Diagram kasus penggunaan dapat dilihat pada Gambar 3.4. Semua Kasus Penggunaan ini tidak didukung sepenuhnya dengan kemampuan *offline*. Misalnya, aktor tetap bisa melihat daftar pengumuman dalam kondisi *offline*.



Gambar 3.4 Kasus Penggunaan Non Pembelajaran

3.4.1.1 Manajemen Pengumuman

Manajemen pengumuman terdiri dari kasus penggunaan melihat daftar pengumuman, membuat pengumuman baru dan menghapus pengumuman.

Kasus penggunaan ini bukan merupakan tujuan dari disusunnya tugas akhir ini. Tetapi sengaja dibuat sebagai bagian

dari pembuatan elearning pada umumnya. Jadi rincian dari kasus penggunaan ini tidak akan dijelaskan dalam buku ini.

3.4.1.2 Manajemen Classroom

Manajemen *classroom* (kelas) terdiri dari kasus penggunaan melihat daftar kelas, menambah kelas baru, mengubah kelas, mengubah *enrollment* pengguna di kelas, dan menghapus kelas.

Kasus penggunaan ini bukan merupakan tujuan dari disusunnya tugas akhir ini. Tetapi sengaja dibuat sebagai bagian dari pembuatan elearning pada umumnya. Jadi rincian dari kasus penggunaan ini tidak akan dijelaskan dalam buku ini.

3.4.1.3 Manajemen Pengguna

Manajemen pengguna terdiri dari kasus penggunaan meilhat daftar pengguna, menambah pengguna baru, mengubah pengguna, dan menghapus pengguna.

Kasus penggunaan ini bukan merupakan tujuan dari disusunnya tugas akhir ini. Tetapi sengaja dibuat sebagai bagian dari pembuatan elearning pada umumnya. Jadi rincian dari kasus penggunaan ini tidak akan dijelaskan dalam buku ini.

3.4.1.4 Manajemen Periode Perkuliahan

Manajemen periode perkuliahan terdiri dari kasus penggunaan melihat daftar periode perkuliahan, menambah periode baru, mengubah periode, dan menghapus periode.

Kasus penggunaan ini bukan merupakan tujuan dari disusunnya tugas akhir ini. Tetapi sengaja dibuat sebagai bagian dari pembuatan elearning pada umumnya. Jadi rincian dari kasus penggunaan ini tidak akan dijelaskan dalam buku ini.

3.4.1.5 Manajemen Matakuliah

Manajemen matakuliah terdiri dari kasus penggunaan melihat daftar matakuliah, menambah matakuliah, mengubah matakuliah, dan menghapus matakuliah.

Kasus penggunaan ini bukan merupakan tujuan dari disusunnya tugas akhir ini. Tetapi sengaja dibuat sebagai bagian dari pembuatan elearning pada umumnya. Jadi rincian dari kasus penggunaan ini tidak akan dijelaskan dalam buku ini.

3.4.1.6 Manajemen Role

Manajemen *role* terdiri dari kasus penggunaan melihat daftar *role*, menambah *role*, mengubah *role*, menghapus *role*, dan memasangkan *role* ke pengguna.

Kasus penggunaan ini bukan merupakan tujuan dari disusunnya tugas akhir ini. Tetapi sengaja dibuat sebagai bagian dari pembuatan elearning pada umumnya. Jadi rincian dari kasus penggunaan ini tidak akan dijelaskan dalam buku ini.

3.4.1.7 Manajemen Permission

Manajemen *permission* terdiri dari kasus penggunaan melihat daftar *permission*, menambah *permission*, mengubah *permission*, dan menghapus *permission*.

Kasus penggunaan ini bukan merupakan tujuan dari disusunnya tugas akhir ini. Tetapi sengaja dibuat sebagai bagian dari pembuatan elearning pada umumnya. Jadi rincian dari kasus penggunaan ini tidak akan dijelaskan dalam buku ini.

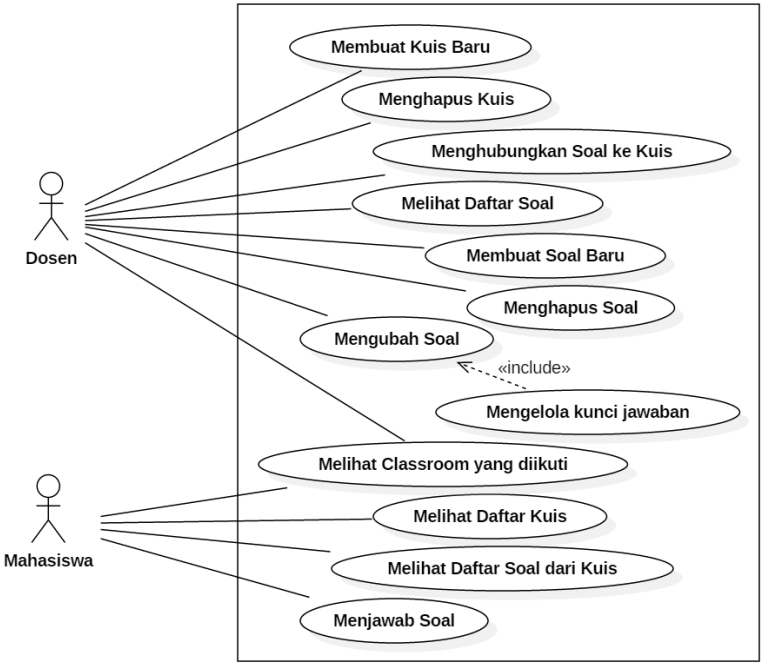
3.4.2 Perancangan Kasus Penggunaan Pembelajaran

Kasus penggunaan ini melibatkan dosen dan mahasiswa sebagai pusat aktifitas pembelajaran. Diagram kasus penggunaan ini dapat dilihat pada Gambar 3.5 dan kode kasus penggunaan pada Tabel 3.1.

Kasus penggunaan yang membutuhkan kemampuan *offline* yaitu kasus penggunaan dengan aktor mahasiswa yaitu kasus

penggunaan melihat kelas yang diikuti, melihat daftar kuis, melihat daftar soal dari kuis, dan menjawab soal.

Kasus penggunaan dengan aktor dosen tidak didukung kemampuan *offline* secara khusus tetapi tetap bisa melihat kelas, kuis, atau soal sebagai fitur bawaan dari *service worker*, tetapi dosen tidak bisa menambah, mengubah, atau menghapus kelas, kuis, atau soal.



Gambar 3.5 Kasus Penggunaan Pembelajaran

Tabel 3.1 Daftar Kode Kasus Penggunaan Pembelajaran

Kode Kasus Penggunaan	Nama Kasus Penggunaan	Kemampuan Offline
UC-01	Membuat Kuis Baru	Tidak
UC-02	Menghapus Kuis	Tidak

Tabel 3.1 Daftar Kode Kasus Penggunaan Pembelajaran

UC-03	Menghubungkan Soal ke Kuis	Tidak
UC-04	Melihat Daftar Soal	Tidak
UC-05	Membuat Soal Baru	Tidak
UC-06	Menghapus Soal	Tidak
UC-07	Mengubah Soal	Tidak
UC-08	Mengelola Kunci Jawaban	Tidak
UC-09	Melihat Classroom yang Diikuti	Ya
UC-10	Melihat Daftar Kuis	Ya
UC-11	Melihat Daftar Soal dari Kuis	Ya
UC-12	Menjawab Soal	Ya

3.4.2.1 Kasus Penggunaan Membuat Kuis Baru

Pada kasus penggunaan ini, dosen dapat membuat kuis baru pada *classroom*. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.2.

Tabel 3.2 Spesifikasi Kasus Penggunaan Membuat Kuis Baru

Kode	UC-01
Nama	**Membuat Kuis Baru**
Aktor	Dosen
Deskripsi	Dosen menambah kuis pada *classroom*
Kondisi Awal	Dosen telah login dan ada di halaman utama
Kondisi Akhir	Kuis ditambahkan ke basis data
Alur Kejadian Normal	
1. Dosen menekan tombol masuk pada *classroom* yang ingin dipilih	

Tabel 3.2 Spesifikasi Kasus Penggunaan Membuat Kuis Baru

2. Sistem menampilkan halaman daftar kuis *classroom* yang dipilih 3. Dosen menekan tombol “Tambah Quiz” 4. Sistem menampilkan formulir Buat Quiz Baru dan daftar kuis yang sudah ada di kelas tersebut 5. Dosen mengisi nama quiz, tanggal mulai dan tanggal selesai, lalu menekan tombol “Buat” 6. Sistem menyimpan kuis baru ke basis data
Alur Kejadian Alternatif
Tidak ada

3.4.2.2 Kasus Penggunaan Menghapus Kuis

Pada kasus penggunaan ini, dosen dapat Menghapus Kuis. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.3

Tabel 3.3 Spesifikasi Kasus Penggunaan Menghapus Kuis

Kode	UC-02
Nama	**Menghapus Kuis**
Aktor	Dosen
Deskripsi	Dosen bisa menghapus kuis yang ada pada *classroom*
Kondisi Awal	Dosen ada di halaman *classroom* yang ingin dihapus kuisnya
Kondisi Akhir	Kuis dihapus dari basis data
Alur Kejadian Normal	
1. Dosen menekan tombol merah dengan lambang tempat sampah pada kuis yang ingin dihapus 2. Sistem menampilkan *pop-up modal* konfirmasi menghapus kuis tersebut 3. Dosen menekan tombol “Delete”	

Tabel 3.3 Spesifikasi Kasus Penggunaan Menghapus Kuis

4. Sistem menghapus kuis tersebut dari basis data 5. Sistem menampilkan halaman *classroom* dengan notifikasi "Quiz berhasil dihapus"
Alur Kejadian Alternatif
1. Dosen menekan tombol merah dengan lambang tempat sampah pada kuis yang ingin dihapus 2. Sistem menampilkan *pop-up modal* konfirmasi menghapus kuis tersebut 3. Dosen menekan tombol "Delete" 4. Sistem gagal menghapus kuis tersebut dari basis data 5. Sistem menampilkan halaman *classroom* dengan notifikasi "Quiz gagal dihapus"

3.4.2.3 Kasus Penggunaan Menghubungkan Soal ke Kuis

Pada kasus penggunaan ini dosen dapat menghubungkan soal ke kuis. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.4

Tabel 3.4 Spesifikasi Kasus Penggunaan Menghubungkan Soal ke Kuis

Kode	UC-03
Nama	**Menghubungkan Soal ke Kuis**
Aktor	Dosen
Deskripsi	Dosen bisa menghubungkan soal yang sudah dibuat sebelumnya ke kuis yang sudah ada
Kondisi Awal	Dosen ada di halaman *classroom* yang ingin disunting
Kondisi Akhir	Soal yang diinginkan terhubung ke kuis
Alur Kejadian Normal	

Tabel 3.4 Spesifikasi Kasus Penggunaan Menghubungkan Soal ke Kuis

1. Dosen menekan tombol sunting berwarna kuning dengan simbol pensil pada kuis yang ingin disunting soal-soalnya 2. Sistem menampilkan daftar seluruh soal yang ada di bank soal dan soal yang terhubung dengan kuis 3. Dosen memberi tanda centang pada soal-soal yang ingin dihubungkan ke kuis lalu menekan tombol “Simpan” 4. Sistem menghubungkan soal yang diberi tanda centang ke kuis tersebut
Alur Kejadian Alternatif
1. Dosen menekan tombol sunting berwarna kuning dengan simbol pensil pada kuis yang ingin disunting soal-soalnya 2. Sistem menampilkan daftar seluruh soal yang ada di bank soal dan soal yang terhubung dengan kuis 3. Dosen menghilangkan tanda centang pada soal-soal yang ingin dihubungkan ke kuis lalu menekan tombol “Simpan” 4. Sistem menghapus hubungan soal yang dihilangkan tanda centangnya dari kuis tersebut

3.4.2.4 Kasus Penggunaan Melihat Daftar Soal

Pada kasus penggunaan ini, dosen dapat melihat daftar seluruh soal yang ada di dalam bank soal. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.5

Tabel 3.5 Spesifikasi Kasus Penggunaan Melihat Daftar Soal

Kode	UC-04
Nama	**Melihat Daftar Soal**
Aktor	Dosen
Deskripsi	Dosen bisa melihat daftar seluruh soal yang ada di dalam sistem

Tabel 3.5 Spesifikasi Kasus Penggunaan Melihat Daftar Soal

Kondisi Awal	Dosen sudah login dan ada di halaman utama
Kondisi Akhir	Daftar soal ditampilkan
Alur Kejadian Normal	
1. Dosen menekan tautan "Lihat Bank Soal" 2. Sistem menampilkan bank soal	
Alur Kejadian Alternatif	
1. Dosen menekan tombol "Bank Soal" pada menu navigasi samping 2. Sistem menampilkan bank soal	

3.4.2.5 Kasus Penggunaan Membuat Soal Baru

Pada kasus penggunaan ini, dosen dapat membuat soal baru. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.6.

Tabel 3.6 Spesifikasi Kasus Penggunaan Membuat Soal Baru

Kode	UC-05
Nama	**Membuat Soal Baru**
Aktor	Dosen
Deskripsi	Dosen bisa membuat soal baru
Kondisi Awal	Dosen sudah login dan ada di tampilan bank soal
Kondisi Akhir	Soal baru ditambahkan ke basis data
Alur Kejadian Normal	
1. Dosen menekan tombol "Add Question" 2. Sistem menampilkan halaman membuat soal baru 3. Dosen mengisi topik, deskripsi, dan kode *template* 4. Dosen menekan tombol "Save" 5. Sistem menyimpan soal ke basis data 6. Sistem menampilkan halaman soal yang baru saja dibuat	

Tabel 3.6 Spesifikasi Kasus Penggunaan Membuat Soal Baru

Alur Kejadian Alternatif
Tidak Ada

3.4.2.6 Kasus Penggunaan Menghapus Soal

Pada kasus penggunaan ini, dosen dapat menghapus soal yang sudah ada. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.7:

Tabel 3.7 Spesifikasi Kasus Penggunaan Menghapus Soal

Kode	UC-06
Nama	**Menghapus Soal**
Aktor	Dosen
Deskripsi	Dosen bisa menghapus soal yang sudah ada
Kondisi Awal	Dosen sudah login dan ada soal di bank soal yang ingin dihapus
Kondisi Akhir	Soal dihapus dari basis data
Alur Kejadian Normal	
1. Dosen menekan tautan “Lihat Bank Soal” di halaman utama 2. Sistem menampilkan bank soal 3. Dosen menekan tombol “Delete” pada soal yang ingin dihapus 4. Sistem menampilkan *pop-up modal* konfirmasi untuk menghapus 5. Dosen menekan tombol “Delete” 6. Sistem menghapus soal dari basis data	
Alur Kejadian Alternatif	
3. Dosen menekan tombol “Bank Soal” pada menu navigasi samping 4. Sistem menampilkan bank soal	

Tabel 3.7 Spesifikasi Kasus Penggunaan Menghapus Soal

5. Dosen menekan tombol “Delete” pada soal yang ingin dihapus 6. Sistem menampilkan *pop-up modal* konfirmasi untuk menghapus 7. Dosen menekan tombol “Delete” 8. Sistem menghapus soal dari basis data

3.4.2.7 Kasus Penggunaan Mengubah Soal

Pada kasus penggunaan ini, dosen dapat mengubah soal. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.8.

Tabel 3.8 Spesifikasi Kasus Penggunaan Mengubah Soal

Kode	UC-07
Nama	**Mengubah Soal**
Aktor	Dosen
Deskripsi	Dosen bisa mengubah soal yang sudah ada
Kondisi Awal	Dosen sudah login dan ada soal di bank soal yang ingin dihapus
Kondisi Akhir	Soal diperbaharui di basis data
Alur Kejadian Normal	
1. Dosen menekan tautan “Lihat Bank Soal” di halaman utama 2. Sistem menampilkan bank soal 3. Dosen menekan tombol “Edit” pada soal yang ingin diubah 4. Sistem menampilkan halaman “*Editing Question*” 5. Dosen mengubah isi soal tersebut lalu menekan tombol “Save” 6. Sistem menyimpan perubahan soal ke basis data	
Alur Kejadian Alternatif	

Tabel 3.8 Spesifikasi Kasus Penggunaan Mengubah Soal

Tidak Ada

3.4.2.8 Kasus Penggunaan Mengelola Kunci Jawaban

Pada kasus penggunaan ini dosen dapat mengelola kunci jawaban dari sebuah soal, yang terdiri dari kasus penggunaan menambah kunci jawaban, kasus penggunaan mengubah kunci jawaban dan kasus penggunaan menghapus kunci jawaban. Semua kasus penggunaan terjadi pada halaman yang sama yaitu halaman menyunting soal. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.9, Tabel 3.10 dan Tabel 3.11.

Tabel 3.9 Spesifikasi Kasus Penggunaan Menambah Kunci Jawaban

Kode	UC-08.1
Nama	**Menambah Kunci Jawaban**
Aktor	Dosen
Deskripsi	Dosen menambah kunci jawaban pada sebuah soal
Kondisi Awal	Dosen sudah login dan sudah ada soal yang ingin ditambahkan kunci jawabannya
Kondisi Akhir	Kunci jawaban ditambahkan ke basis data
Alur Kejadian Normal	
1. Dosen menekan tautan “Lihat Bank Soal” di halaman utama 2. Sistem menampilkan bank soal 3. Dosen menekan tombol “Edit” pada soal yang ingin diubah 4. Sistem menampilkan halaman “*Editing Question*” 5. Dosen mengisi formulir “*Challenges*” dan “*Message*” disebelah kanan 6. Sistem menyimpan perubahan soal ke basis data	

Tabel 3.9 Spesifikasi Kasus Penggunaan Menambah Kunci Jawaban

Kode	UC-08.1
Alur Kejadian Alternatif	
Tidak Ada	

Tabel 3.10 Spesifikasi Kasus Penggunaan Mengubah Kunci Jawaban

Kode	UC-08.2
Nama	**Mengubah Kunci Jawaban**
Aktor	Dosen
Deskripsi	Dosen mengubah kunci jawaban dari sebuah soal
Kondisi Awal	Dosen sudah login dan sudah ada kunci jawaban pada soal
Kondisi Akhir	Kunci jawaban diperbaharui di basis data
Alur Kejadian Normal	
1. Dosen menekan tautan “Lihat Bank Soal” di halaman utama 2. Sistem menampilkan bank soal 3. Dosen menekan tombol “Edit” pada soal yang ingin diubah 4. Sistem menampilkan halaman “*Editing Question*” 5. Dosen menekan simbol edit pada “*Challenges*” 6. Sistem menampilkan *pop-up modal* “*Edit Challenges*” 7. Dosen mengubah isi formulir kunci jawaban lalu menekan tombol “*Save*” 8. Sistem menyimpan perubahan kunci jawaban ke basis data	
Alur Kejadian Alternatif	
Tidak Ada	

Tabel 3.11 Spesifikasi Kasus Penggunaan Menghapus Kunci Jawaban

Kode	UC-08.3
Nama	**Menghapus Kunci Jawaban**
Aktor	Dosen
Deskripsi	Dosen menghapus kunci jawaban dari sebuah soal
Kondisi Awal	Dosen sudah login dan sudah ada kunci jawaban pada soal
Kondisi Akhir	Kunci jawaban dihapus dari basis data
Alur Kejadian Normal	
1. Dosen menekan tautan "Lihat Bank Soal" di halaman utama 2. Sistem menampilkan bank soal 3. Dosen menekan tombol "Edit" pada soal yang ingin diubah 4. Sistem menampilkan halaman "*Editing Question*" 5. Dosen menekan simbol delete pada "*Challenges*" 6. Sistem menampilkan *pop-up modal* konfirmasi menghapus kunci jawaban 7. Dosen menekan tombol "Delete" 8. Sistem menghapus kunci jawaban dari basis data	
Alur Kejadian Alternatif	
Tidak Ada	

3.4.2.9 Kasus Penggunaan Melihat Classroom yang Diikuti

Pada kasus penggunaan ini, dosen dan mahasiswa dapat melihat classroom yang diikuti, kasus penggunaan ini menjadi kasus penggunaan yang penting sehingga ditempatkan di halaman utama sebagai tampilan *default*. Rincian alur kasus penggunaan ini dijelaskan pada

Tabel 3.12 Spesifikasi Kasus Penggunaan Melihat Classroom yang Diikuti

Kode	UC-09
Nama	**Melihat Classroom yang Diikuti**
Aktor	Dosen atau Mahasiswa
Deskripsi	Dosen atau mahasiswa bisa melihat classroom yang diikuti
Kondisi Awal	Dosen atau mahasiswa sudah login
Kondisi Akhir	Sistem menampilkan daftar classroom yang diikuti dosen atau mahasiswa
Alur Kejadian Normal	
1. Dosen atau mahasiswa login ke dalam system 2. Sistem menampilkan halaman utama dengan daftar classroom yang diikuti	
Alur Kejadian Alternatif	
Tidak Ada	

3.4.2.10 Kasus Penggunaan Melihat Daftar Kuis

Pada kasus penggunaan ini, mahasiswa dapat melihat daftar kuis dari kelas yang diikuti. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.13.

Tabel 3.13 Spesifikasi Kasus Penggunaan Melihat Daftar Kuis

Kode	UC-10
Nama	**Melihat Daftar Kuis**
Aktor	Mahasiswa
Deskripsi	Mahasiswa bisa melihat kuis dari classroom
Kondisi Awal	Mahasiswa sudah login
Kondisi Akhir	Sistem menampilkan daftar classroom yang diikuti dosen atau mahasiswa

Tabel 3.13 Spesifikasi Kasus Penggunaan Melihat Daftar Kuis

Alur Kejadian Normal
1. Mahasiswa menekan tombol masuk pada classroom yang ingin dilihat daftar kuisnya 2. Sistem menampilkan daftar quiz dari classroom tersebut
Alur Kejadian Alternatif
Tidak Ada

3.4.2.11 Kasus Penggunaan Melihat Daftar Soal dari Kuis

Kasus Penggunaan UC-11 merupakan kasus penggunaan Melihat Daftar Soal dari Kuis. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.14.

Tabel 3.14 Spesifikasi Kasus Penggunaan Melihat Daftar Kuis

Kode	UC-11
Nama	**Melihat Daftar Soal dari Kuis**
Aktor	Mahasiswa
Deskripsi	Mahasiswa bisa melihat daftar soal dari kuis
Kondisi Awal	Mahasiswa sudah login
Kondisi Akhir	Sistem menampilkan daftar soal dari kuis
Alur Kejadian Normal	
1. Mahasiswa menekan tombol masuk pada classroom yang ingin dilihat daftar kuisnya 2. Sistem menampilkan daftar kuis dari classroom tersebut 3. Mahasiswa menekan tombol Masuk pada kuis 4. Sistem menampilkan daftar soal pada kuis tersebut	
Alur Kejadian Alternatif	
Tidak Ada	

3.4.2.12 Kasus Penggunaan Menjawab Soal

Kasus Penggunaan UC-11 merupakan kasus penggunaan Menjawab Soal. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.15.

Tabel 3.15 Spesifikasi Kasus Penggunaan Menjawab Soal

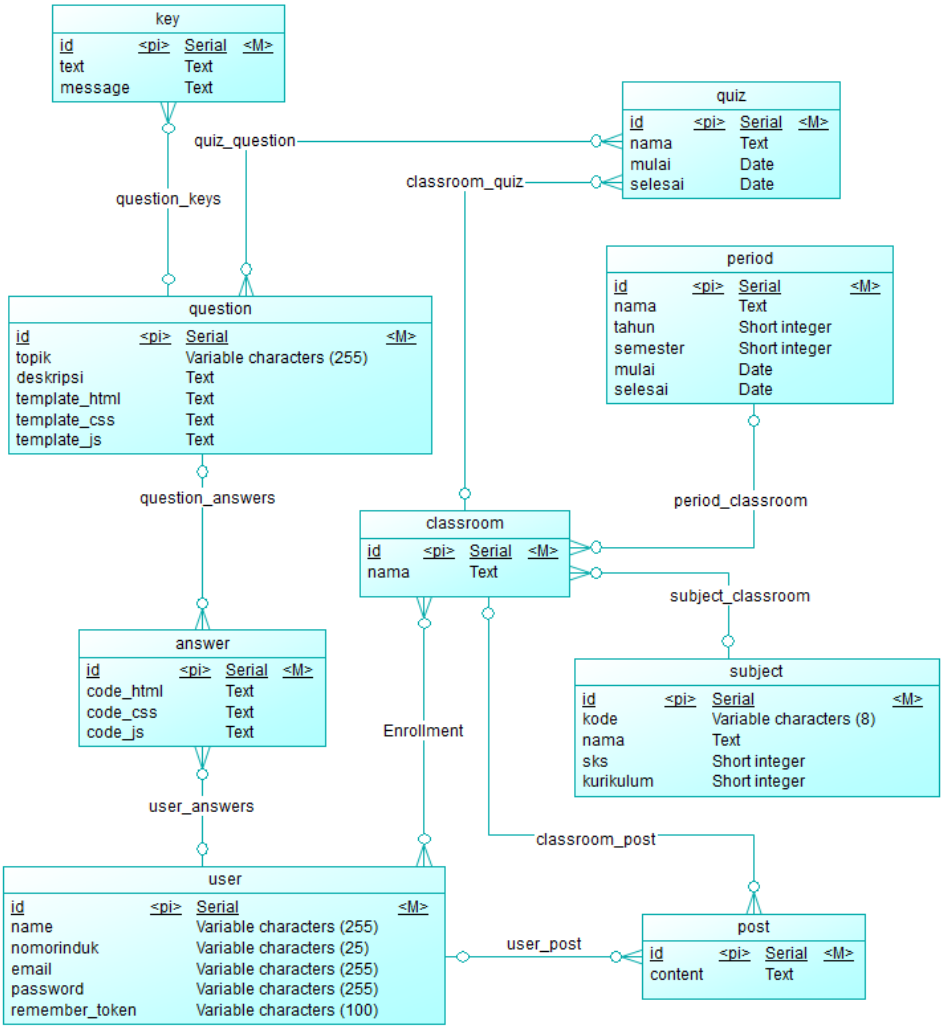
Kode	UC-12
Nama	**Menjawab Soal**
Aktor	Mahasiswa
Deskripsi	Mahasiswa bisa menjawab soal dan mendapatkan penilaian
Kondisi Awal	Mahasiswa sudah login
Kondisi Akhir	Sistem menyimpan jawaban mahasiswa ke basis data
Alur Kejadian Normal	
1. Mahasiswa menekan tombol tanda panah pada daftar soal 2. Sistem menampilkan soal yang dipilih 3. Mahasiswa menjawab soal dengan mengisi jawaban pada *code editor* 4. Mahasiswa menekan tombol “Run” untuk menjalankan proses penilaian 5. Sistem menampilkan hasil penilaian untuk setiap kunci jawaban 6. Jika semua penilaian benar maka sistem menampilkan *pop-up modal* sukses 7. Mahasiswa menekan tombol “Next Question” 8. Sistem menyimpan jawaban mahasiswa dan status selesai pada soal ke basis data 9. Sistem menampilkan daftar soal dengan tanda centang pada soal yang sudah diselesaikan	

Tabel 3.15 Spesifikasi Kasus Penggunaan Menjawab Soal

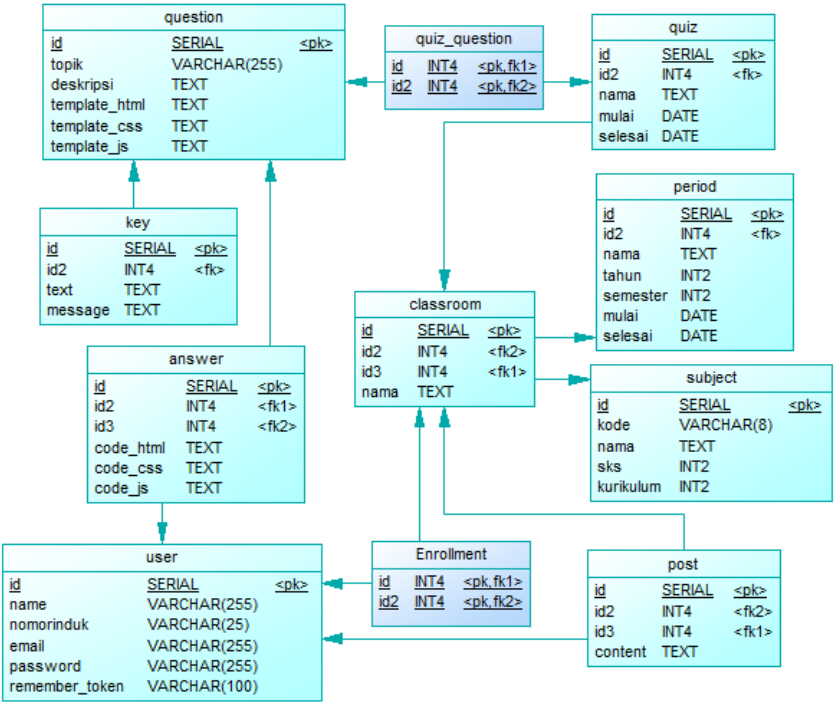
Alur Kejadian Alternatif
1. Mahasiswa menekan tombol tanda panah pada daftar soal 2. Sistem menampilkan soal yang dipilih 3. Mahasiswa menjawab soal dengan mengisi jawaban pada *code editor* 4. Mahasiswa menekan tombol “Run” untuk menjalankan proses penilaian 5. Sistem menampilkan hasil penilaian untuk setiap kunci jawaban 6. Jika ada penilaian berstatus salah maka sistem menampilkan status kesalahan 7. Sistem menyimpan jawaban dan status belum selesai pada soal ke basis data

3.5 Perancangan Basis Data

Perangkat lunak yang digunakan untuk mengelola dan memanajemen basis data dalam sistem ini adalah PostgreSQL. Perancangan basis data *e-Learning* yang berupa CDM dapat dilihat pada Gambar 3.6 dan PDM dapat dilihat pada Gambar 3.7.



Gambar 3.6 CDM Basis Data e-Learning



Gambar 3.7 PDM Basis Data e-Learning

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan Implementasi

Aplikasi web *e-Learning* diimplementasi pada server *shared hosting* berlokasi di Singapura dengan spesifikasi sebagai berikut:

- Perangkat keras
  - CPU 16 *cores* (*shared*)
  - Memori RAM 2.00 GB
- Perangkat lunak
  - Sistem Operasi: linux
  - Kernel: 3.10.0-427.36.1.lve1.4.44.el7.x86_64
  - Apache Version: 2.4.25
  - PHP Version: 5.6.29
  - PostgreSQL Version: 9.2.18

4.2 Implementasi Kebutuhan Fungsional

Pembahasan implementasi akan dibagi berdasarkan kasus penggunaan pembelajaran. Pada setiap kasus penggunaan akan dibahas implementasi kode program dan antarmuka. Implementasi kasus penggunaan ini akan dibagi berdasarkan kelompok *model*-nya menjadi sebagai berikut:

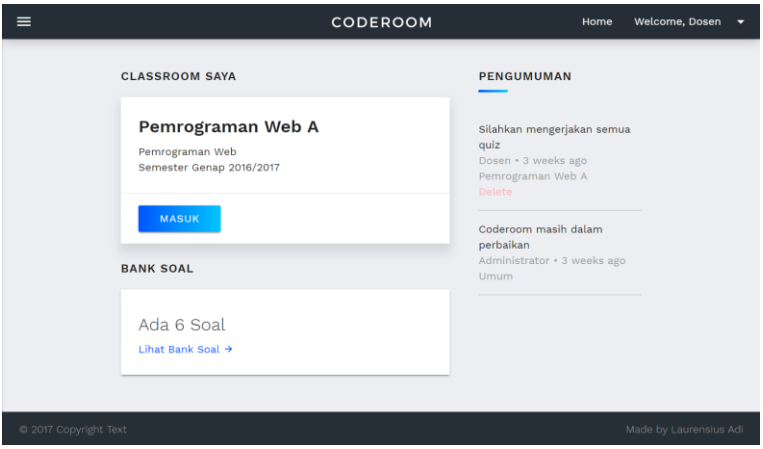
1. Mengelola Kuis, yang mencakup:
   a. UC-01: Membuat Kuis Baru
   b. UC-02: Menghapus Kuis
   c. UC-10: Melihat Daftar Kuis
2. Mengelola Soal dan Kuis, yang mencakup:
   a. UC-03: Menghubungkan Soal ke Kuis
   b. UC-11: Melihat Daftar Soal dari kuis

3. Mengelola Soal, yang mencakup:
    a. UC-04: Melihat Daftar Soal
    b. UC-05: Membuat Soal Baru
    c. UC-06: Menghapus Soal
    d. UC-07: Mengubah Soal
4. Mengelola Kunci Jawaban: yang mencakup:
    a. UC-08.1: Menambah Kunci Jawaban
    b. UC-08.2: Mengubah Kunci Jawaban
    c. UC-08.3: Menghapus Kunci Jawaban
5. Melihat Classroom yang diikuti, yang mencakup: UC-09
6. Menjawab Soal, yang mencakup: UC-12

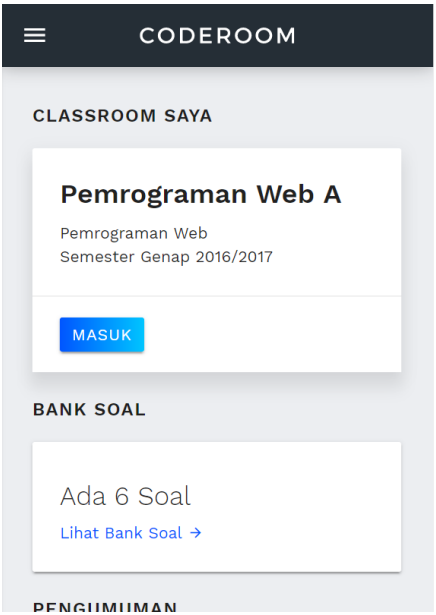
Laravel sebagai kerangka kerja memiliki gaya tersendiri untuk menangani kasus CRUD ke dalam basis data berupa *Resource Controller*. Ketika membuat *controller* menggunakan *command line* dan menambahkan parameter *resource* akan dibuat *controller* dengan 7 fungsi bawaan di dalamnya, yaitu *index*, *create*, *store*, *show*, *edit*, *update*, *delete*. Tidak semua fungsi akan dipakai pada implementasi sebuah proses, karena pertimbangan pengalaman pengguna. Contohnya, fungsi *show* yang menampilkan detail sebuah entri akan sedikit dipakai karena detail entri tersebut sudah ditampilkan pada daftar entri.

Salah satu karakteristik dalam PWA yaitu *Responsive*, dimana aplikasi dapat ditampilkan pada berbagai ukuran layar gadget dan masih menyajikan pengalaman pengguna yang sama. Pada tugas akhir ini akan digunakan kerangka kerja CSS bernama Materialize seperti yang dijelaskan pada Subbab 2.8.

Pada bagian implementasi antarmuka akan banyak digunakan elemen *card view* yang memudahkan pengaturan tata letak antar elemen di dalam satu buah halaman, sehingga dapat memisahkan antar bagian satu dengan yang lainnya seperti yang ditunjukkan pada Gambar 4.1. Informasi yang ditampilkan dalam sebuah elemen *card view* juga enak dipandang dalam layar kecil sekalipun seperti pada Gambar 4.2.



Gambar 4.1 Antarmuka Halaman Utama pada Dosen

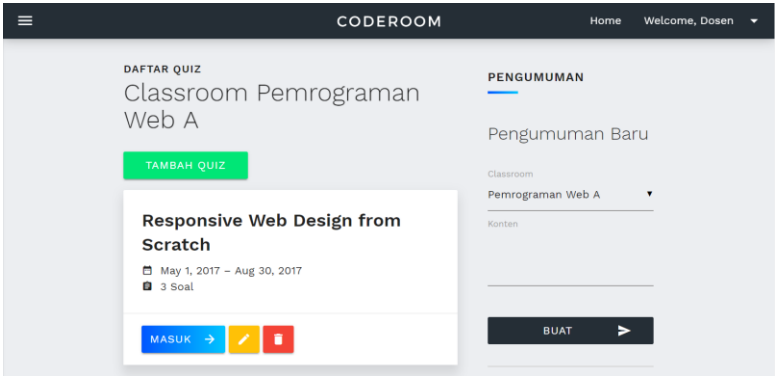


Gambar 4.2 Antarmuka Halaman Utama pada Layar Ukuran *Mobile*

4.2.1 Implementasi Proses Mengelola Kuis

Di dalam proses mengelola kuis, dosen dapat melihat daftar kuis, membuat kuis baru, dan menghapus kuis. Kelas yang mengatur aktivitas ini adalah kelas QuizController.

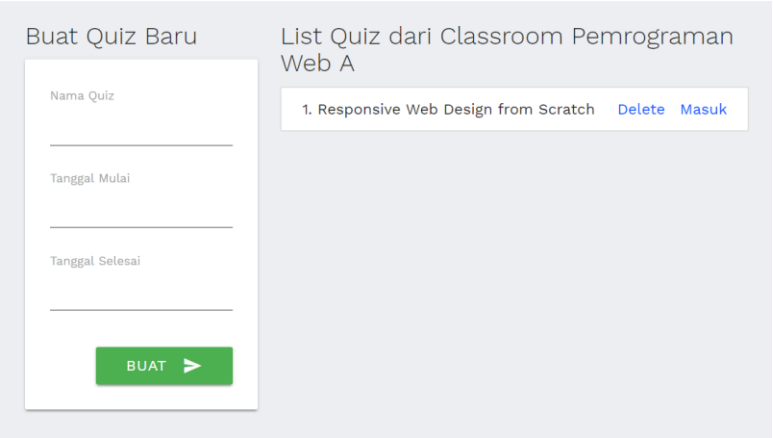
Dosen memulai proses ini dari halaman *Classroom* yang dipilih seperti ditunjukkan pada Gambar 4.3. Pada halaman *Classroom* ditampilkan daftar kuis yang ada, dengan detail tanggal mulai dan selesai serta jumlah soal yang ada pada kuis tersebut. Di atas daftar kuis, terdepat tombol "Tambah Quiz" untuk menambah kuis baru, yang ditampilkan pada halaman berbeda.



Gambar 4.3 Antarmuka Classroom

Pada halaman *Classroom* ini juga ditambahkan bagian formulir untuk membuat pengumuman pada kelas yang bersangkutan untuk memudahkan dosen menambahkan pengumuman.

Di dalam proses Membuat Kuis Baru, dosen dapat membuat kuis baru untuk *Classroom* yang dipilih seperti yang ditunjukkan pada Gambar 4.4. Satu buah *classroom* bisa memiliki banyak kuis. Daftar kuis yang sudah ada ditampilkan bersebelahan dengan formulir untuk memudahkan dosen melihat daftar kuis. Proses menyimpan kuis baru ditunjukkan pada Kode Sumber 4.1.



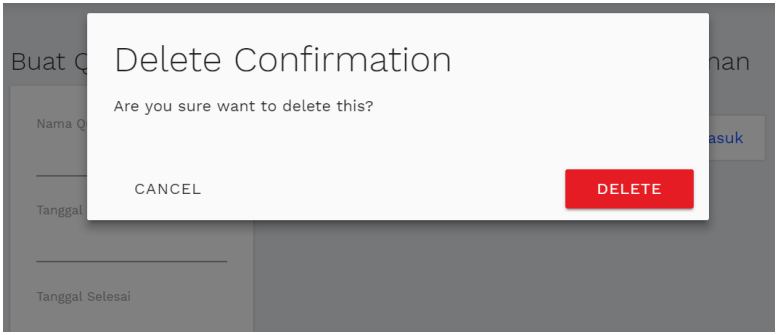
Gambar 4.4 Antarmuka Membuat Kuis Baru

```
public function store(Request $request, $id)
{
    $this->validate($request, [
        'nama' => 'required',
        'mulai' => 'required',
        'selesai' => 'required',
        ]);

    $quiz = new Quiz;
    $quiz->classroom_id = $id;
    $quiz->nama = $request->nama;
    $quiz->mulai = $request->mulai;
    $quiz->selesai = $request->selesai;
    $quiz->save();

    return redirect('classroom/'.$id.'/quiz')-
>with('message', 'Quiz baru berhasil ditambahkan');
}
```

Kode Sumber 4.1 Proses Membuat Kuis Baru

Dosen dapat menghapus kuis yang ada dengan menekan tulisan “Delete” di sebelah kanan tulisan kuis pada daftar kuis yang ditunjukkan pada Gambar 4.4. Setelah itu akan muncul *pop-*

up modal berisi konfirmasi untuk menghapus kuis seperti yang ditunjukkan pada Gambar 4.5.



Gambar 4.5 Antarmuka Konfirmasi Menghapus Kuis

4.2.2 Implementasi Proses Mengelola Soal dan Kuis

Di dalam proses mengelola soal dan kuis, terdapat implementasi kode program untuk menghubungkan soal dengan kuis dan melihat daftar soal dari kuis. Proses ini melibatkan actor mahasiswa pada bagian Melihat Daftar Soal dari Kuis, dan dosen dalam Menghubungkan Soal dengan Kuis.

Proses Mengelola Soal dan Kuis ini diatur dalam kelas QuizQuestionContoller yang menjadi jembatan antara kuis dan soal yang memiliki hubungan *many-to-many*-. Hubungan *many-to-many* ini juga dideklarasikan pada masing-masing model *Quiz* dan *Question* seperti pada Kode Sumber 4.2 dan Kode Sumber 4.3.

```
class Quiz extends Model
{
    public function questions()
    {
        return $this->belongsToMany('App\Question',
       'elearningnew.quiz_questions', 'quiz_id',
       'question_id')->withTimestamps();
    }
}
```

Kode Sumber 4.2 Model Quiz

```
class Question extends Model
{
    public function quizes()
    {
        return $this->belongsToMany('App\Quiz',
       'elearningnew.quiz_questions', 'question_id',
       'quiz_id')->withTimestamps();
    }
}
```

Kode Sumber 4.3 Model Question

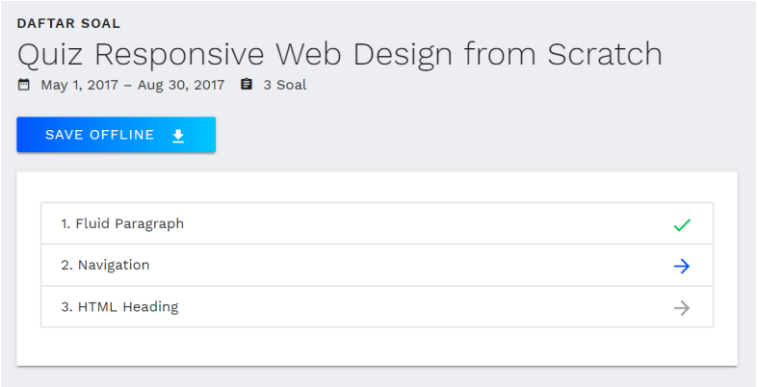
Implementasi kode program Mengelola Soal dan Kuis diatur pada kelas QuizQuestionController, yang memiliki 3 fungsi yaitu index() untuk menampilkan daftar soal dari kuis, fungsi show() untuk menampilkan soal, dan fungsi update() untuk menghubungkan soal ke kuis.

```
$questions = $quiz->questions()->orderBy('created_at',
'asc')->get();
```

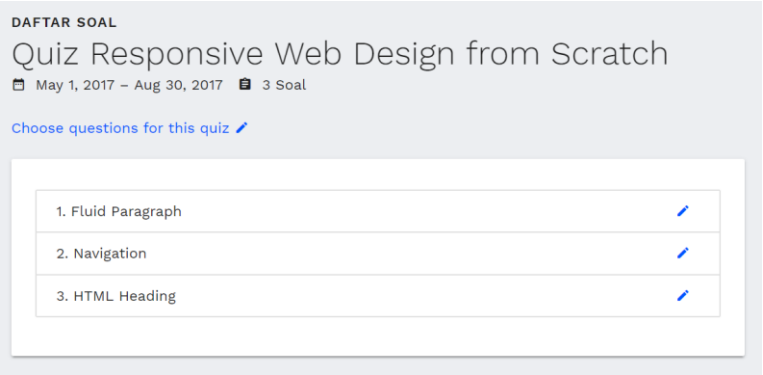
Kode Sumber 4.4 Menampilkan Daftar Soal dari Kuis

Pada fungsi index() seperti ditunjukkan pada Kode Sumber 4.4, daftar soal diambil dari basis data menggunakan fungsi *eloquent model* yang sudah dideklarasikan pada Kode Sumber 4.2. Fungsi *index* ini digunakan untuk menampilkan soal dari kuis pada mahasiswa dan dosen. Mahasiswa dapat melihat daftar soal dari kuis seperti yang ditunjukkan pada Gambar 4.6.

Pada tampilan antarmuka daftar soal dari kuis untuk dosen, dosen bisa mengubah soal yang dipilih dan memilih soal-soal untuk kuis tersebut dengan menekan tombol yang sesuai seperti yang ditunjukkan pada Gambar 4.7

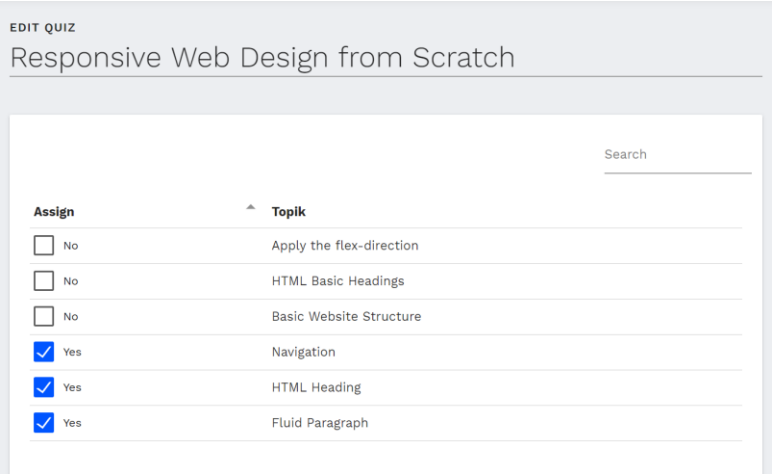


Gambar 4.6 Antarmuka Menampilkan Daftar Soal dari Kuis dengan Aktor Mahasiswa



Gambar 4.7 Antarmuka Menampilkan Daftar Soal dari Kuis dengan Aktor Dosen

Dosen dapat memilih soal-soal yang akan ditampilkan pada kuis dengan Mengubah Kuis seperti yang ditampilkan pada Gambar 4.8. Proses ini diimplementasikan pada fungsi update(), soal dan kuis dihubungkan atau dilepas hubungannya menggunakan fungsi *sync* dan *detach* seperti ditunjukkan pada Kode Sumber 4.5.



Gambar 4.8 Antarmuka Menghubungkan Soal ke Kuis

```
if (!empty($request['data::'.$question_id])) {
    $question = Question::find($question_id);
    $question->quizes()->sync([$id]);
} elseif (empty($request['data::'.$question_id])) {
    $question = Question::find($question_id);
    $question->quizes()->detach([$id]);
}
```

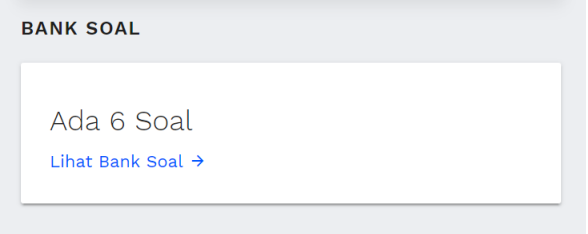
Kode Sumber 4.5 Menghubungkan Soal ke Kuis

4.2.3 Implementasi Proses Mengelola Soal

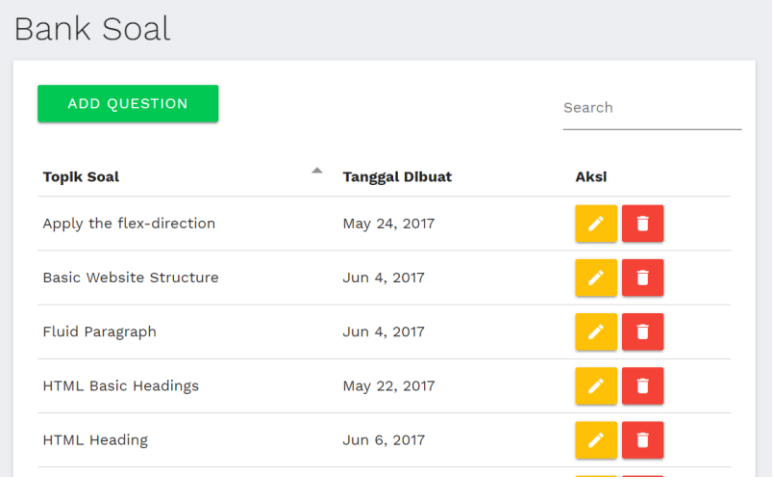
Proses Mengelola Soal sepenuhnya dijalankan oleh actor dosen. Di dalam proses mengelola soal, terdapat kasus penggunaan melihat daftar soal, membuat soal baru, menghapus soal, dan mengubah soal. Aktivitas ini diatur dalam kelas QuestionController.

Soal menjadi objek penting yang sering diakses oleh dosen selain *classroom* yang diikuti, jadi diberikan akses dari navigasi samping (*sidebar*) dan halaman utama ke Bank Soal seperti pada Gambar 4.9. Setelah menekan *link* Bank Soal, akan ditampilkan halaman Bank Soal dengan daftar soal seperti pada Gambar 4.10.

Dosen dapat menambah soal dengan menekan tombol “Add Question”, mengubah soal dengan menekan tombol bersimbol *edit* dan menghapus dengan menekan tombol bersimbol *delete* pada soal tersebut.



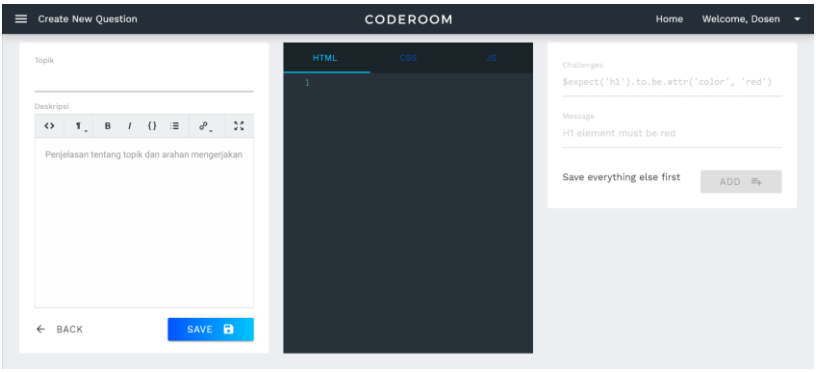
Gambar 4.9 Antarmuka Akses Bank Soal pada Halaman Utama



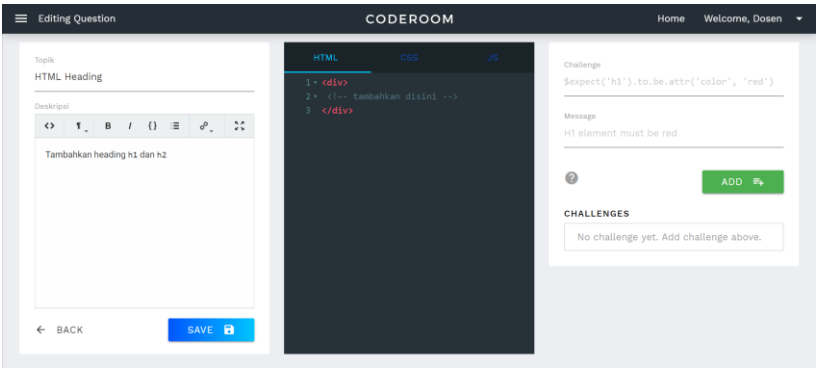
Gambar 4.10 Antarmuka Bank Soal

Pada antarmuka Membuat Soal Baru seperti pada Gambar 4.11, digunakan *editor* teks WYSIWYG (What You See Is What You Get) bernama Trumbowyg untuk memudahkan penulisan deskripsi. Pada *editor* tersebut dosen bisa menambahkan *tag* HTML seperti <code> untuk menujukkan bahwa bagian teks yang dimaksud adalah sebuah kode sumber.

Setelah mengisi topik, deskripsi, dan *template* kode sumber HTML, CSS, atau JS, lalu menekan tombol "Save" akan ditampilkan antarmuka mengubah soal seperti ditunjukkan pada Gambar 4.12. Antarmuka mengubah soal juga dapat dibuka dengan menekan tombol *edit* pada halaman Bank Soal. Pada saat menekan tombol "Save" terjadi proses penyimpanan soal ke dalam basis data seperti yang ditunjukkan pada Kode Sumber 4.6.



Gambar 4.11 Antarmuka Membuat Soal Baru



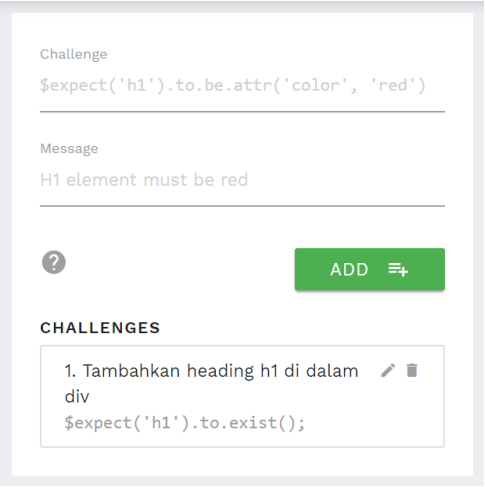
Gambar 4.12 Antarmuka Mengubah Soal

```
public function store(Request $request)
{
    $question = new Question;
    $question->topik = $request->topik;
    $question->deskripsi = $request->deskripsi;
    $question->template_html = $request->html;
    $question->template_css = $request->css;
    $question->template_js = $request->js;
    $question->save();

    return redirect('question/'.$question->id.'/edit')-
>with('message', 'Soal berhasil dibuat');
}
```

Kode Sumber 4.6 Proses Menyimpan Soal Baru

4.2.4 Implementasi Proses Mengelola Kunci Jawaban

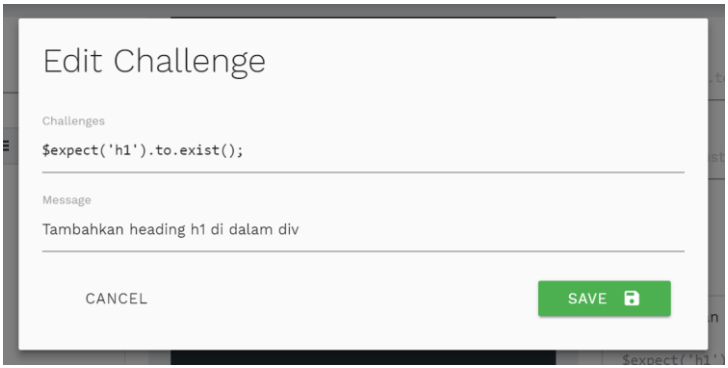
Dosen dapat menambahkan, mengubah dan menghapus kunci jawaban pada soal yang sudah dibuat. Seluruh proses mengelola kunci jawaban ini terjadi di dalam halaman Mengubah Soal seperti yang ditunjukkan pada Gambar 4.12.



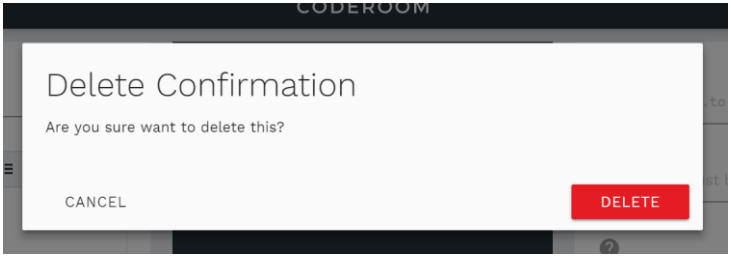
Gambar 4.13 Antarmuka Kunci Jawaban

Untuk menambah kunci jawaban baru, dosen mengisi formulir “Challenges” pada Mengubah Soal. Pada formulir tesebut, terdapat bagian *challenge* yang harus diisi sesuai ketentuan jquery.expect. Ketika sudah mengisi *challenge* dan *message* lalu menekan tombol “Add”, maka kunci jawaban akan ditambahkan ke dalam basis data dan ditampilkan di bawah formulir seperti yang ditunjukkan pada Gambar 4.13.

Dosen juga dapat mengubah kunci jawaban dengan menekan ikon *edit* pada kunci jawaban, maka akan muncul *pop-up modal* untuk mengubah kunci jawaban seperti ditunjukkan pada Gambar 4.14. Untuk menghapus kunci jawaban, dosen dapat menekan ikon *delete* pada kunci jawaban, lalu akan muncul dialog konfirmasi seperti pada Gambar 4.15.



Gambar 4.14 Antarmuka Mengubah Kunci Jawaban



Gambar 4.15 Antarmuka Menghapus Kunci Jawaban

4.2.5 Implementasi Proses Melihat Classroom yang diikuti

Proses melihat classroom diimplementasikan dengan menggunakan *eloquent model* dengan menghubungkan pengguna dengan kelas menggunakan tabel *enrollment* yang menjadi perantara hubungan *many-to-many* antara model *User* dan model *Classroom* yang dideklarasikan pada kelas *User* seperti yang bisa dilihat pada Kode Sumber 4.7.

```
class User extends Authenticatable
{
    public function classrooms()
    {
        return $this->belongsToMany('App\Classroom',
       'elearningnew.enrollment', 'user_id',
       'classroom_id')->withTimestamps();
    }
}
```

Kode Sumber 4.7 Model User

```
public function handle($request, Closure $next)
{
    $enrollmentId = $request->route('id');
    $enrollment = Enrollment::find($enrollmentId);
    if($enrollment->user_id !== Auth::id()) {
        return redirect('home')->with('error', 'User
not enrolled on classroom');
    }
    else {
        return $next($request);
    }
}
```

Kode Sumber 4.8 Middleware EnrollementCheck

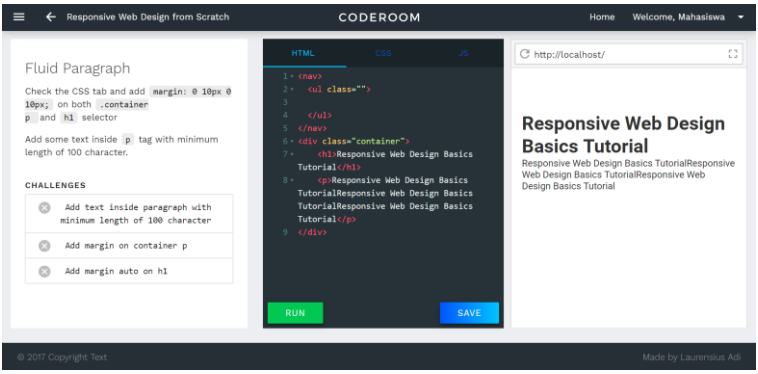
Pada perancangan basis data, tabel kuis dihubungkan ke tabel classroom, dan semua pengguna mengakses *route* yang sama yang merupakan sebuah kesalahan perancangan karena menyebabkan halaman *offline* pada jawaban mahasiswa semuanya

memiliki URL yang sama. Untuk menangani masalah ini perlu perubahan pada *route* sehingga mengakses kelas menggunakan *id* dari *enrollment* dan ditambahkan pengecekan terhadap *role* dosen dan mahasiswa untuk setiap *route* classroom menggunakan *middleware* tambahan seperti yang ditunjukkan pada Kode Sumber 4.8.

4.2.6 Implementasi Proses Menjawab Soal

Ketika mahasiswa menekan salah satu tanda panah pada daftar soal, akan ditampilkan antarmuka mengerjakan soal seperti pada Gambar 4.16. Proses menampilkan halaman menjawab soal ini diatur pada kelas QuizQuestionContoller pada fungsi *show*. Di dalam proses menampilkan data soal yaitu topik dan deskripsi, *template* kode HTML, CSS, dan JS, data tersebut secara bersamaan disimpan kedalam basis data tabel *answer*, sehingga yang selalu ditampilkan ke antarmuka adalah dari tabel *answer* dan tidak perlu melakukan pengecekan disisi *controller* setiap kali mahasiswa membuka halaman soal.

Di dalam fungsi menampilkan soal juga menampilkan kunci jawaban tetapi perlu dimodifikasi, seperti ditunjukkan pada Kode Sumber 4.9, agar bisa bekerja dengan MochaJS dan jquery.expect karena kunci jawaban perlu dicocokan pada *iframe* yang merupakan *document* yang berbeda di dalam halaman soal.



Gambar 4.16 Antarmuka Mengerjakan Soal

```
function str_replace_first($from, $to, $subject)
{
    $from = '/'.preg_quote($from, '/').'/';
    return preg_replace($from, $to, $subject, 1);
}
foreach ($keys as $key) {
    $first = str_replace_first("').","')).", $key-
      >checklist);
    $second = str_replace("expect('",
      "expect($('#iframe').contents().find('", $first);
    $key->checklist = $second;
}
```

Kode Sumber 4.9 Modifikasi Kunci Jawaban

```
public function update(Request $request, $id, $quiz_id,
$question_id)
{
    if(request()->ajax())
    {
        $answerId = Answer::where('question_id',
         $request->qid)->where('user_id', Auth::id())-
         >get()->first()->id;
        $answer = Answer::find($answerId);
        $answer->code_html = $request->html;
        $answer->code_css = $request->css;
        $answer->code_js = $request->js;
        $answer->save();
    }
}
```

Kode Sumber 4.10 Menyimpan Soal

Proses menyimpan jawaban soal diimplementasikan pada kelas AnswerController pada fungsi *update* yang bisa dilihat pada Kode Sumber 4.10. Jawaban akan disimpan ke basis data setiap kali mahasiswa menekan tombol “Save” pada antarmuka menjawab soal menggunakan fungsi *ajax* dari jQuery seperti ditunjukkan pada Kode Sumber 4.11.

```
$(".update").click(function(e) {
    e.preventDefault();
    var url = "/classroom/{{ $classroom-
    >enrollmentId($classroom) }}/quiz/{{ $quiz->id
    }}/question/{{ $question->id }}/answer";
    var $answer = {};
    $answer.qid = '{{ $question->id }}';
    $answer.html = html_editor.getValue();
    $answer.css = css_editor.getValue();
    $answer.js = js_editor.getValue();

    $.ajax({
        headers: {'X-CSRF-Token' : $('meta[name=csrf-
        token]').attr('content')},
        type: "POST",
        url: url,
        data: $answer,
        cache: false,
        success: function(data){
            render();
            Materialize.toast('Saved', 3000, 'blue');
        },
        error: function(data) {
            Materialize.toast('Not Saved', 3000,
            'materialize-red');
        }
    });
});
```

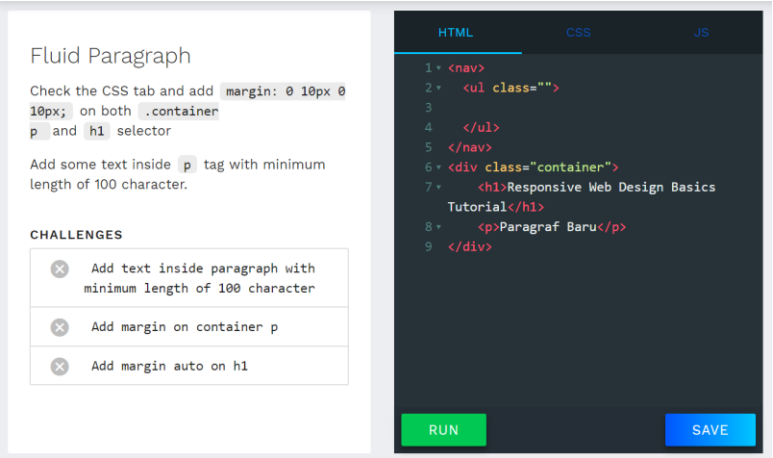
Kode Sumber 4.11 Fungsi *Ajax Update*

Untuk menyelesaikan sebuah soal, mahasiswa harus memenuhi *challenges* yang diberikan di soal dengan menekan tombol “Run” seperti yang ditunjukkan pada Gambar 4.17. Ketika tombol *run* ditekan, fungsi MochaJS akan dijalankan dan akan melakukan *assertion test* dengan kunci jawaban terhadap iframe hasil render dari kode HTML, CSS, dan JS yang ditulis oleh mahasiswa seperti yang ditunjukkan oleh Kode Sumber 4.12. Ketika MochaJS selesai melakukan penilaian akan ditampilkan

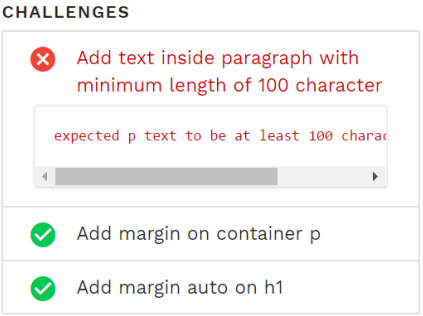
status setiap penilaian kunci jawaban seperti ditunjukkan pada Gambar 4.19.

```
describe('Challenges', function () {
it('Add text inside paragraph with minimum length of
    100 character', function () {
    $expect($('#iframe').contents().find('p')).to.have
    .text(100); });
it('Add margin on container p', function () {
    $expect($('#iframe').contents().find('.container
    p')).to.have.css('margin','0px 10px 0px 10px');
    });
it('Add margin auto on h1', function () {
    $expect($('#iframe').contents().find('h1')).to.hav
    e.css('margin','0px 10px 0px 10px'); });
});
window.DONE = false;
$(function () {
document.getElementById("judge").addEventListener("clic
    k", function(){
        var challenges = $("#mocha ul");
        if(window.DONE == true) { location.reload(); }
        else if(window.DONE == false) {
            var challenges = $("#mocha ul");
            if("challenges") {
                challenges.remove();
            }
            mocha.run(function () {
                window.DONE = true;
                var failures =
    document.querySelectorAll('.test.fail').length;
                if(!failures){
                    console.log("No failures");
                    $('#modalnext').modal('open');
                }
            Materialize.toast('Checked', 3000, 'blue');
            });
        }
    });
});
```

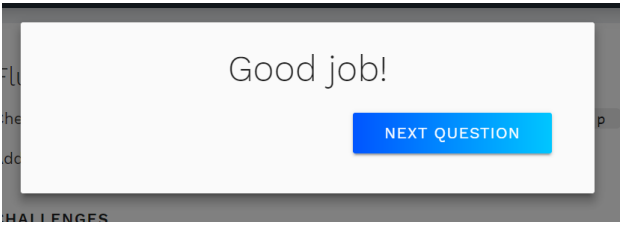
Kode Sumber 4.12 Modul Penilaian MochaJS



Gambar 4.17 Antarmuka Menjawab Soal



Gambar 4.18 Antarmuka Hasil Penilaian



Gambar 4.19 Antarmuka *Modal* Menjawab Soal dengan Benar

Jika *assertion test* menyatakan semua *challenges* telah lulus maka akan ditampilkan *pop-up modal* yang tersambung dengan fungsi *done* pada kelas AnswerController seperti pada Kode Sumber 4.13.

```
public function done($id, $quiz_id, $question_id) {
    $answerId = Answer::where('question_id',
    $question_id)->where('user_id', Auth::user()->id)-
    >get()->first()->id;
    $answer = Answer::find($answerId);
    $answer->done = true;
    $answer->save();
    return redirect('/classroom/'. $id .'/quiz/'.
    $quiz_id '/question/')->with('message','Question
    done');
}
```

Kode Sumber 4.13 Menandakan Soal Selesai Dijawab

4.3 Implementasi Kemampuan *Offline*

Pada bagian ini akan dijelaskan mengenai implementasi kebutuhan non-fungsional yaitu kemampuan aplikasi untuk berjalan secara *offline* dengan *service worker* dan menyimpan soal-soal dari kuis agar bisa dikerjakan secara *offline*.

4.3.1 Implementasi Service Worker

Implementasi kemampuan *offline* berpusat pada *service worker* dan terdiri dari berkas JavaScript *service worker* dan berkas JavaScript untuk meregistrasi *service worker*. *Service worker* digunakan untuk menangani koneksi, respons dan *request* maka harus diimplementasikan pada *domain* dengan protokol HTTPS agar terhindar dari pembajakan.

Pada awal pengembangan *e-Learning* ini, *service worker* yang diimplementasikan menggunakan strategi *caching cache first, network fallback*. Strategi ini dipilih karena waktu responsnya yang cepat. Setiap *Request* yang diminta oleh halaman, dilakukan pengecekan terlebih dahulu apakah ada

Request yang sama di dalam *cache*, jika ada maka akan dikembalikan *Response* dari *cache*.

Ternyata strategi *cache first* memiliki kekurangan yang fatal karena *token* Laravel yang digunakan oleh Laravel untuk mengenali *Request* ke server adalah dari *Request* sebelumnya, maka sering terjadi eror pada *form* yang ada di aplikasi, terutama *login*.

Pada versi *service worker* selanjutnya digunakan *service worker* seperti yang telah dirancang pada Subbab 3.3.3, strategi *service worker* yang digunakan adalah *network first, cache fallback*.

Service worker yang baru ini sudah tidak mengalami kesalahan pada *token* Laravel, tetapi pengembalian respons ke halaman membutuhkan waktu yang lama karena menunggu *Response* dari server, atau menunggu sampai server tidak memberikan *Response*, barulah mengambil *Response* dari *cache*.

Untuk mengatasi hal ini ditambahkan satu buat pilihan *fetch* data yaitu ketika data yang dimintakan adalah data statis yang sudah ada di dalam *cache* yang bertipe JavaScript, CSS, gambar, atau *font* maka *Response* akan langsung dikembalikan dari *cache*.

Service worker yang diimplementasikan bisa dilihat pada Kode Sumber 4.14.

```
var cacheName = "shell-content";
var filesToCache = [
    "/css/materialize-prod.css",
    "/css/datatables.materialize.css",
    "/css/mocha.css",
    "/codemirror/codemirror.css",
    "/codemirror/material.css",
    "/codemirror/addon/lint/lint.css",
    "/codemirror/addon/hint/show-hint.css",
    "/codemirror/addon/fold/foldgutter.css",
    "/js/mocha.js", "/js/expect.js",
    "/js/jquery.expect.js",
    "/js/jquery.datatables.min.js",
    "/js/jquery.min.js",
    "/js/materialize.min.js",
```

```
    "/js/code-render.js",
    "/js/code-grammar.js",
    "/css/trumbowyg.css",
    "/js/wyg.js",
    "/fonts/worksans/WorkSans-Light.ttf",
    "/fonts/worksans/WorkSans-Regular.ttf",
    "/fonts/worksans/WorkSans-SemiBold.ttf",
    "/fonts/worksans/WorkSans-Bold.ttf",
    "/fonts/MaterialIcons-Regular.ttf",
    "/fonts/MaterialIcons-Regular.woff",
    "/fonts/MaterialIcons-Regular.woff2",
    "/images/coderoom-logo-white.svg",
    "/images/coderoom-logo-white.svg",
    "/images/coderoom-type-emblem.svg",
    "/images/favicon32x32.png",
    "/images/icons.svg",
    "/images/icons/icon-72x72.png",
    "/images/icons/icon-96x96.png",
    "/images/icons/icon-128x128.png",
    "/images/icons/icon-144x144.png",
    "/images/icons/icon-152x152.png",
    "/images/icons/icon-192x192.png",
    "/images/icons/icon-384x384.png",
    "/images/icons/icon-512x512.png",
    "/offline"
];

self.addEventListener("install", function(event) {
    self.skipWaiting();
    event.waitUntil(
        caches.open(cacheName).then(function(cache) {
            return cache.addAll(filesToCache);
        })
    );
});

self.addEventListener("activate", function(event) {
  event.waitUntil(
    caches.keys().then(function(keyList) {
      return Promise.all(keyList.map(function(key) {
        if (key !== cacheName) {
          return caches.delete(key);
        }
```

```
      }));
  }).then(function(){
      return self.clients.claim();
  })
  );
});

self.addEventListener("fetch", function(event) {
    if (event.request.method !== "GET") { return; }
    if (event.request.url !== "/sw.js" &&
event.request.url.split('.').pop().match(/^(js|css|png|svg
|ttf|woff|woff2)$/)) {
      event.respondWith(
          caches.match(event.request)
          .then(function(response) {
            var networkResponse = fetch(event.request);
              caches.open(cacheName)
              .then(function(cache) {
                cache.put(event.request, networkResponse);
              });
              return response;
          })
      );
    }
    else {
        event.respondWith(
            fetch(event.request).then(function(response) {
                var networkResponse = response.clone();
                caches.open(cacheName)
                .then(function(cache) {
                cache.put(event.request, networkResponse);
                });
                return response;
            }).catch(function() {
                return caches.match(event.request);
            })
        );
    }
});
```

Kode Sumber 4.14 Service Worker

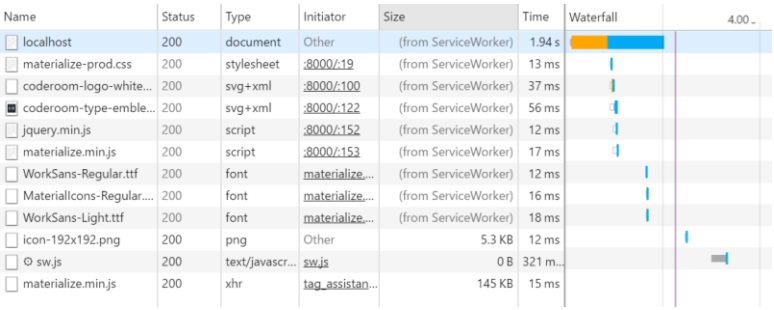
Variabel cacheName digunakan untuk memberi nama pada penyimpanan *cache* pada *browser* pengguna, sedangkan variable filesToCache menyimpan *array* nama-nama berkas yang akan di-*cache* ketika *service worker* di-*install*.

Fungsi addEventListener install akan menjalankan *service worker* ke tahap *install*, dimana seluruh file yang akan di-*cache* disimpan ke dalam *cache*.

Fungsi addEventListener activate berjalan ketika *install* telah selesai dan akan dilakukan pengecekan terhadap *cache* pada *browser*, ketika sudah ada *cache* yang lama maka akan dihapus, lalu ditambahkan *cache* baru.

Fungsi addEventListener fetch menjadi pusat aktivitas *service worker* dimana seluruh *event request* GET akan diatur, sedangkan *request* lain seperti POST, PUT, dan DELETE akan diteruskan ke server.

Di dalam fetch ada pengecekan kondisi apabila URL *request* bertipe js, css, png, svg, ttf, woff, atau woff2 maka akan dikembalikan langsung *response* berupa berkas statis dari *cache* dan versi *cache* akan selalu diperbarui setiap *request*. Desain seperti ini mengurangi waktu untuk memuat *resource* ke halaman web cukup signifikan. Waktu yang dibutuhkan untuk memuat sebuah *resource* dari *cache* hanya puluhan mili detik seperti ditunjukkan pada Gambar 4.20.



Gambar 4.20 *Request Table* pada Chrome DevTools

Semua *request* selain berkas statis akan diteruskan ke server seperti yang sudah dijelaskan pada Gambar 3.2 ketika kondisi *online*. Lalu *response* dari server akan disimpan ke dalam *cache*. Ketika kondisi *offline*, maka akan dikembalikan *response* dari *cache*.

4.3.2 Implementasi Menyimpan Soal dari Kuis secara Offline

Menyimpan soal dari kuis secara offline membutuhkan mekanisme mengambil halaman soal-soal dan menyimpannya ke dalam *cache* seperti ditunjukkan pada Kode Sumber 4.15.

```
caches.open('shell-content').then(function(cache) {
    cache.addAll(pages);
}
```

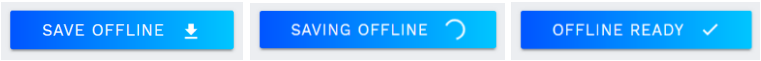
Kode Sumber 4.15 Menambahkan Halaman ke Cache

Fungsi *cache.addAll* diberikan parameter *pages* yang berisi *array* halaman-halaman yang mau disimpan, dalam hal ini URL dari setiap soal yang ada pada kuis. Permasalahan yang muncul dengan menyimpan langsung URL ke dalam *cache* adalah ketika menggunakan *fetch*, *request* tidak menyertakan *credentials* seperti *cookies*, lalu laravel akan me-*redirect Request* ke halaman login karena mekanisme yang dimiliki Laravel untuk melindungi pengguna dari serangan. Untuk menangani masalah tersebut, pada *Request* disertakan *credentials: include*. Kode program yang menangani proses ini ditunjukkan pada Kode Sumber 4.16.

```
<script>
var cacheButton = document.querySelector('.offline-btn');
if(cacheButton) {
    cacheButton.addEventListener('click', function(event)
{
        event.preventDefault();
        var pages = [];
```

```
        @foreach($questions as $question)
        pages.push(new Request('/classroom/{{
       Enrollment::where('classroom_id',$classroom->id)-
       >where('user_id', Auth::id())->first()->id
       }}/quiz/{{ $quiz->id }}/question/{{ $question->id
       }}', {credentials: 'include'}),);
        @endforeach

        caches.open('shell-content').then(function(cache){
            $('.offline-btn').html('Saving offline <div
             class="preloader-wrapper tiny right active"
             style="top:8px;margin-left:15px"><div
             class="spinner-layer spinner-white-only"><div
             class="circle-clipper left"><div
             class="circle"></div></div><div class="gap-
             patch"><div class="circle"></div></div><div
             class="circle-clipper right"><div
             class="circle"></div></div></div></div>')
            var updateCache = cache.addAll(pages);
            updateCache.then(function() {
                 $('.offline-btn').html('Offline Ready <i
                 class="material-icons right">check</i>');
            }).catch(function (error) {
                 Materialize.toast('Quiz could not be
                 saved offline', 4000, 'materialize-red');
            });
        });
    });
}
</script>
```

Kode Sumber 4.16 Menyimpan Soal dari Kuis secara *Offline*



Gambar 4.21 Tombol Penyimpanan *Offline*

Fungsi menyimpan soal dari kuis secara *offline* ini berjalan ketika menekan tombol “Save Offline”, lalu akan ditampilkan *loading spinner* dan ketika proses penyimpanan selesai, tombol

berubah menjadi “Offline Ready” seperti ditunjukkan pada Gambar 4.21. Setelah itu soal-soal dapat diakses secara *offline* dan dapat dikerjakan serta dilakukan pengecekan jawaban, tetapi tidak bisa menyimpan jawaban karena membutuhkan koneksi ke basis data.

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dibahas mengenai Pengujian dari segi fungsionalitas aplikasi. Pengujian fungsionalitas dibagi menjadi beberapa skenario fungsionalitas yang terdapat pada aplikasi

5.1 Lingkungan Pelaksanaan Pengujian

Lingkungan pengujian yang akan digunakan untuk melakukan implementasi adalah sebagai berikut:

- Perangkat keras
  - Intel(R) Core(TM) i5-7200U CPU @ 2.50 GHz ~2.70 GHz
  - Memori RAM 8.00 GB
- Perangkat lunak
  - Sistem Operasi: Windows 10 Education 64-bit version 1607 build 14393.1198
  - Google Chrome 58.0.3029.110 (64-bit)
  - Lighthouse 2.1.0

5.2 Skenario Pengujian

Pengujian terhadap kebutuhan non-fungsional akan dilakukan dengan 2 cara, yaitu pengujian PWA menggunakan Lighthouse dan pengujian koneksi dan respons menggunakan Chrome DevTools.

5.2.1 Pengujian Menggunakan Lighthouse

Untuk menguji kebutuhan non-fungsional, pengujian dilakukan menggunakan *extension* Google Chrome yaitu Lighthouse [15]. Lighthouse adalah aplikasi *open source* yang dibangun oleh Google untuk menguji konsep PWA dengan aspek-aspeknya, *performance*, *accessibility* dan *best practices*. Lighthouse dapat dijalankan dari *browser* pengguna berupa ekstensi pada Google Chrome.

Sebuah aplikasi bisa dikatakan sebagai PWA apabila memenuhi standar PWA *Checklist* yang meliputi:

- Meregistrasi *Service Worker*
- Merespons dengan kode 200 ketika *offline*
- Aplikasi masih menampilkan konten walaupun tanpa dukungan JavaScript
- Menggunakan HTTPS
- Me-*redirect* lalu lintas HTTP ke HTTPS
- Halaman dapat dimuat cukup cepat pada jaringan 3G
- Pengguna bisa diminta untuk instalasi *web app* ke *homescreen*
- Menggunakan *custom splash screen*
- *Address bar* cocok dengan warna *brand*
- Memiliki *tag* <meta name="viewport"> dengan *width* atau *initial-scale*
- Konten memiliki ukuran yang benar dengan *viewport*

Pengujian menggunakan Lighthouse hanya bisa dijalankan untuk satu pengujian pada satu *browser client*. Untuk melakukan pengujian menggunakan Lighthouse pada banyak *client* digunakan layanan *third-party* Calibre [16]. Pengujian pada Calibre dilakukan secara otomatis menggunakan *emulation* yang menyerupai penggunaan *client* pada *browser* Chrome.

5.2.2 Pengujian Network Response Menggunakan Chrome DevTools

Untuk menguji kinerja *service worker* pada platform *e-Learning* digunakan Chrome DevTools dengan berbagai kondisi. Pengujian ini bertujuan untuk menunjukkan kinerja *service worker* pada aplikasi web dengan melihat jumlah *request*, waktu yang dibutuhkan untuk *load* semua konten halaman, dan jumlah data yang dikirim.

Ada empat kondisi skenario pengujian yaitu:

1. Kunjungan Halaman : pertama kali
   Service Worker : belum ada

	Cache halaman	: belum ada
	Internet	: *online*
2.	Kunjungan Halaman	: setelah pertama
	Service Worker	: sudah aktif
	Cache halaman	: sudah ada
	Internet	: *online*
3.	Kunjungan Halaman	: setelah pertama
	Service Worker	: sudah aktif
	Cache halaman	: belum ada
	Internet	: *online*
4.	Kunjungan Halaman	: setelah pertama
	Service Worker	: sudah aktif
	Cache halaman	: sudah ada
	Internet	: *offline*

5.3 Evaluasi Pengujian

Pada subbab ini akan ditunjukkan hasil pengujian dan evaluasi hasil pengujian dari skenario yang sudah dibuat pada subbab sebelumnya.

5.3.1 Evaluasi Pengujian Menggunakan Lighthouse

Pada subbab ini akan dibahas hasil pengujian menggunakan Lighthouse dengan bantuan Calibre. Pengujian dilakukan sebanyak 30 kali pada 4 kriteria penilaian yaitu *PWA, Performance, Accessibility,* dan *Best Practices*. Hasil pengujian bisa dilihat padaTabel 5.1.

Tabel 5.1 Hasil Pengujian Lighthouse Menggunakan Calibre

No	PWA	Performance	Accessibility	Best Practices
1.	100	95	97	100
2.	100	85	97	100
3.	100	92	97	100
4.	100	96	97	100
5.	100	95	97	100
6.	100	53	97	100

Tabel 5.1 Hasil Pengujian Lighthouse Menggunakan Calibre

7.	100	83	97	100
8.	100	96	97	100
9.	100	94	97	100
10.	100	93	97	100
11.	100	63	97	100
12.	100	62	97	100
13.	100	95	97	100
14.	100	94	97	100
15.	100	96	97	100
16.	100	95	94	100
17.	100	92	97	100
18.	100	93	97	100
19.	100	92	97	100
20.	100	94	97	100
21.	100	95	97	100
22.	100	80	97	100
23.	100	96	97	100
24.	100	68	97	100
25.	100	69	97	100
26.	100	96	97	100
27.	100	94	97	100
28.	100	96	97	100
29.	100	72	97	100
30.	100	62	97	100

Hasil pengujian menggunakan Lighthouse, aplikasi web yang dibangun bisa dikatakan sudah memenuhi kriteria sebuah *Progressive Web Apps* dengan nilai 100 dari 100. Aplikasi juga sudah mendapat nilai yang baik pada ketiga kriteria lainnya, kriteria *performance* dengan rata-rata 85 dari 100, kriteria *accessibility* dengan rata-rata 97 dari 100, dan kriteria *best practices* dengan rata-rata 100 dari 100.

Pada kriteria *performance* nilai yang didapat memiliki rentang yang cukup besar yakni dari 53 sampai 96. Rentang nilai yang besar ini disebabkan karena nilai *performace* sangat bergantung pada kondisi jaringan internet. Ketika jaringan internet lamban, maka akan dihasilkan nilai rendah.

Simulasi menggunakan bantuan Calibre memiliki sedikit kekurangan karena tidak menggambarkan berbagai *device* dan jaringan yang mungkin digunakan pengguna pada praktik sebenarnya karena pengujian dilakukan dalam *emulation*.

5.2.3 Evaluasi Pengujian Network Response Menggunakan Chrome DevTools

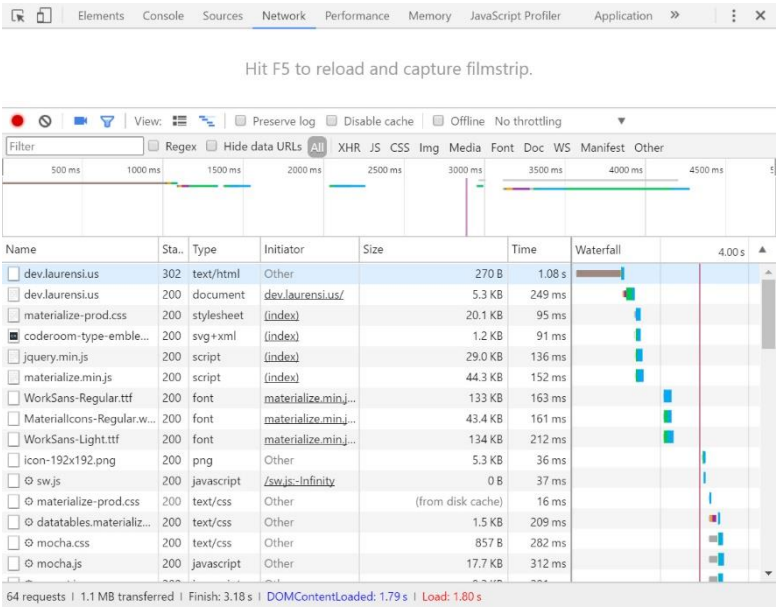
Pada subbab ini akan dijelaskan hasil pengujian *network response* menggunakan Chrome DevTools. Pembahasan dibagi berdasarkan skenario yang telah direncanakan sebelumnya.

Pada skenario pertama halaman akan diuji dengan kondisi:

- Kunjungan Halaman : pertama kali
- Service Worker : belum ada
- Cache halaman : belum ada
- Internet : online

Hasil pengujian skenario pertama ditunjukkan pada Gambar 5.1. Dari pengujian pertama diketahui ada 64 *request* yang dibuat dari halaman, 1.1 MB data diterima, waktu memuat halaman 1.8 detik, dan selesai membuat semua *request* pada detik ke 3.18.

Jumlah *request* yang banyak ini berasal dari *service worker* dimana semua berkas statis diunduh dan disimpan ke dalam *cache.* Proses ini terjadi dibelakang layar dan setelah halaman selesai dimuat sehingga tidak mengganggu pengguna dalam menggunakan aplikasi.



Gambar 5.1 ***Network Response First Visit***

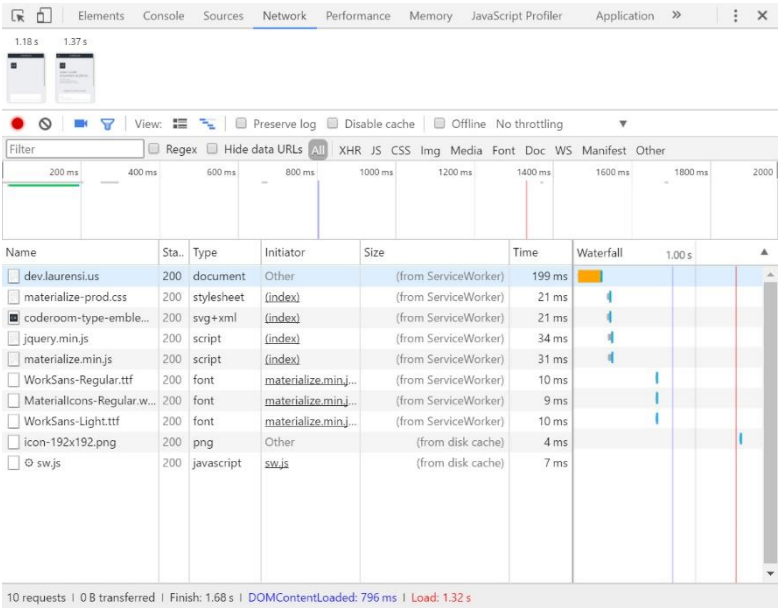
Pada skenario kedua halaman akan diuji dengan kondisi:

- Kunjungan Halaman : setelah pertama
- Service Worker : sudah aktif
- Cache halaman : sudah ada
- Internet : online

Hasil pengujian skenario kedua ditunjukkan pada Gambar 5.2. Dari pengujian kedua diketahui ada 10 *request* yang dibuat dari halaman, 0 B data diterima, waktu memuat halaman 1.32 detik, dan selesai membuat semua *request* pada detik ke 1.68.

Jumlah *request* yang dibuat halaman yang sama setelah *service worker* aktif jauh lebih sedikit dari skenario pertama karena *service worker* tidak mengunduh data baru lagi untuk berkas statis. Pada skenario ini tidak ada data yang diterima oleh halaman karena semua data sudah ada di dalam *cache* dan dikembalikan ke

halaman oleh *service worker* sehingga waktu memuat halaman lebih cepat 0.48 detik atau 26.6%.



Gambar 5.2 ***Network Response Repeat Visit***

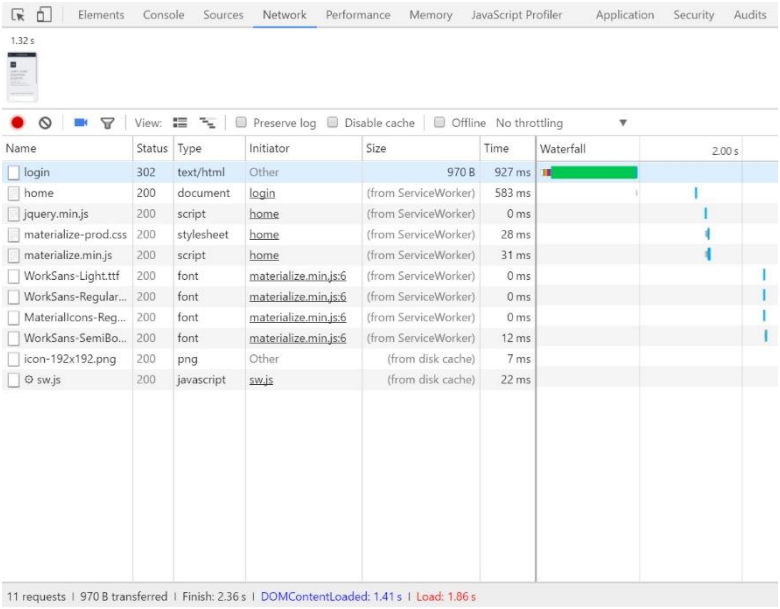
Pada skenario ketiga halaman akan diuji dengan kondisi:

- Kunjungan Halaman : setelah pertama
- Service Worker : sudah aktif
- Cache halaman : belum ada
- Internet : online

Hasil pengujian skenario ketiga ditunjukkan pada Gambar 5.3. Dari pengujian ketiga diketahui ada 11 *request* yang dibuat dari halaman, 970 B data diterima, waktu memuat halaman 1.86 detik, dan selesai membuat semua *request* pada detik ke 2.36.

Jumlah *request* yang dibuat bertambah satu karena melalui proses login terlebih dahulu. Data yang diterima bertambah karena halaman yang dikunjungi tidak tersedia di dalam *cache* sehingga

request diteruskan ke server dan data yang diterima hanyalah data HTML halaman baru tersebut, sedangkan berkas statis menggunakan data dari *cache*.

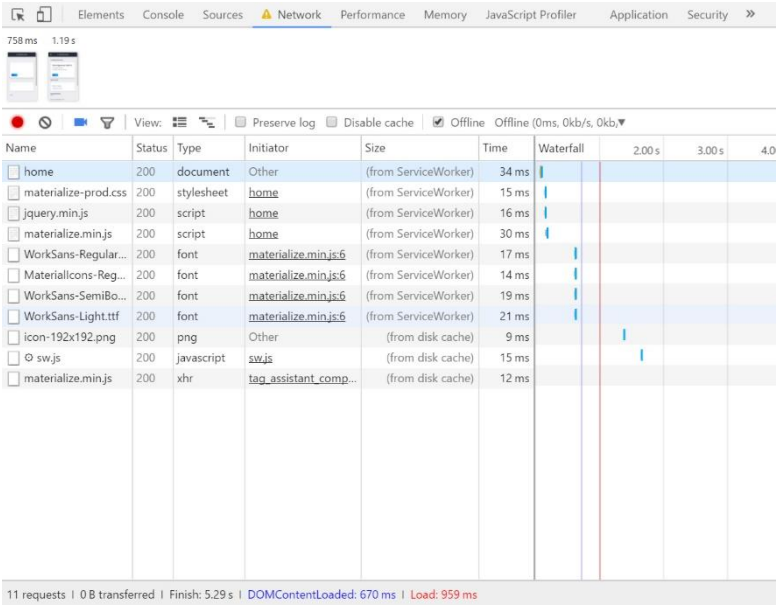


Gambar 5.3 *Network Response No Cache Page*

Pada skenario keempat halaman akan diuji dengan kondisi:

- Kunjungan Halaman : setelah pertama
- Service Worker : sudah aktif
- Cache halaman : belum ada
- Internet : online

Hasil pengujian skenario keempat ditunjukkan pada Gambar 5.4. Dari pengujian keempat diketahui ada 11 *request* yang dibuat dari halaman, 0 B data diterima, waktu memuat halaman 0.959 detik, dan selesai membuat semua *request* pada detik ke 5.29.



Gambar 5.4 ***Network Response Offline***

Pengujian ini menunjukkan aplikasi dapat berjalan dalam kondisi offline. Pada pengujian ini aplikasi memuat konten ke halaman jauh lebih cepat dari sebelumnya karena tidak menunggu respons dari server sama sekali, dimana semua respons dikembalikan dari dalam *cache*.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini diberikan kesimpulan yang diambil selama pengerjaan Tugas Akhir beserta saran-saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1 Kesimpulan

Selama proses perancangan, implementasi, dan pengujian dapat diambil kesimpulan sebagai berikut:

1. Platform *e-Learning* dengan penerapan konsep Progressive Web Apps mampu menampilkan halaman secara *offline* tetapi tidak bisa menyimpan, mengubah, atau menghapus data pada basis data.
2. Proses penilaian bisa dilakukan dalam kondisi *offline* dan memenuhi kebutuhan.
3. Platform *e-Learning* sudah memenuhi kebutuhan fungsional.
4. Hasil pengujian menggunakan Lighthouse menunjukkan rata-rata nilai 100 pada kriteria *Progressive Web Apps*, 85 pada kriteria *performance*, 97 pada kriteria *accessibility*, dan 100 pada kriteria *best practices*.
5. Waktu *loading* halaman lebih cepat 26.6% dengan *cache* dan *service worker*.

6.2 Saran

Saran untuk pengembangan dan perbaikan sistem di masa yang akan datang, di antaranya adalah sebagai berikut:

1. Aplikasi bisa dimodifikasi menggunakan arsitektur *Client Side Rendering* menggunakan kerangka kerja *front-end* JavaScript seperti Vue JS, Angular JS, atau React JS untuk aplikasi yang lebih cepat dan menghemat pengiriman data menggunakan JSON.

2. Aplikasi bisa dibuat mampu menangani *request* selain GET dalam kondisi *offline* dengan menunda *request* sampai sistem kembali online.
3. Modul penilaian menggunakan jquery expect masih harus menulis manual, bisa ditambahkan *parser* untuk *suggestion* ketika menulis kunci jawaban.

DAFTAR PUSTAKA

[1] "Your First Progressive Web App | Web," *Google Developers*. [Daring]. Tersedia pada: https://developers.google.com/web/fundamentals/getting-started/codelabs/your-first-pwapp/. [Diakses: 05-Jun-2017].

[2] "Can I use... Support tables for HTML5, CSS3, etc." [Daring]. Tersedia pada: https://caniuse.com/#feat=serviceworkers. [Diakses: 05-Jun-2017].

[3] "Introduction to Progressive Web App Architectures | Web," *Google Developers*. [Daring]. Tersedia pada: https://developers.google.com/web/ilt/pwa/introduction-to-progressive-web-app-architectures. [Diakses: 02-Jun-2017].

[4] "Cache," *Mozilla Developer Network*. [Daring]. Tersedia pada: https://developer.mozilla.org/en-US/docs/Web/API/Cache. [Diakses: 06-Jun-2017].

[5] D. Walsh, "Cache API," *David Walsh Blog*, 14-Feb-2016. .

[6] "Introduction to Progressive Web App Architectures | Caching Strategies Supported by sw-toolbox." [Daring]. Tersedia pada: https://developers.google.com/web/ilt/pwa/introduction-to-progressive-web-app-architectures#caching_strategies_supported_by_sw-toolbox. [Diakses: 05-Jun-2017].

[7] "CodeMirror." [Daring]. Tersedia pada: https://codemirror.net/. [Diakses: 01-Jun-2017].

[8] "About - Materialize." [Daring]. Tersedia pada: http://materializecss.com/about.html. [Diakses: 01-Jun-2017].

[9] "Introduction - Material design," *Material design guidelines*. [Daring]. Tersedia pada: https://material.io/guidelines/. [Diakses: 11-Jun-2017].

[10] "PostgreSQL: About." [Daring]. Tersedia pada: https://www.postgresql.org/about/. [Diakses: 05-Jun-2017].

[11]"Mocha - the fun, simple, flexible JavaScript test framework." [Daring]. Tersedia pada: https://mochajs.org/. [Diakses: 05-Jun-2017].
[12]"The jquery.expect Library," *Codecademy*. [Daring]. Tersedia pada: https://www.codecademy.com/blog/14. [Diakses: 01-Jun-2017].
[13]"Codecademy - learn to code, interactively, for free," *Codecademy*. [Daring]. Tersedia pada: https://www.codecademy.com/. [Diakses: 20-Jun-2017].
[14]"Learn to code and help nonprofits," *Free Code Camp*. [Daring]. Tersedia pada: http://www.freecodecamp.com. [Diakses: 20-Jun-2017].
[15]"Lighthouse | Web | Google Developers." [Daring]. Tersedia pada: https://developers.google.com/web/tools/lighthouse/. [Diakses: 15-Jun-2017].
[16]"Calibre - Web performance monitoring." [Daring]. Tersedia pada: https://calibreapp.com/. [Diakses: 21-Jun-2017].

BIODATA PENULIS

Laurensius Adi lahir di Jakarta pada tanggal 5 September 1995. Penulis menempuh pendidikan mulai dari SD Santo Kristoforus 2 Jakarta Barat (2001-2007), SMP Santo Kristoforus 2 Jakarta Barat (2007-2010), SMA Pangudi Luhur Van Lith Muntilan Jawa Tengah (2010-2013) dan sekarang sedang menjalani pendidikan S1 Teknik Informatika di Institut Teknologi Sepuluh Nopember Surabaya. Selama berkuliah di Teknik Informatika, penulis mengambil rumpun mata kuliah Algoritma dan Pemrograman. Selama dalam kuliah, penulis banyak belajar mengenai pemrograman berbasis web, pemrograman perangkat bergerak, dan pernah menciptakan beberapa permainan.

Selain di bidang akademik, penulis juga aktif di kepanitiaan ITS EXPO 2014 sebagai Staff Web dan Dokumentasi, ITS EXPO 2015 sebagai Koordinator Multimedia, dan ITS EXPO 2016 sebagai Konseptor Branding. Selain kegiatan kampus, penulis aktif membuat dan mengerjakan desain, *lettering* dan tipografi. Penulis juga pernah mengajar beberapa pelatihan *lettering* di Surabaya. Penulis bisa dihubungi melalui email laurensiusadi@gmail.com atau hello@laurensi.us.