

MANAJEMEN TEMPAT DUDUK PADA OXAM

CHRISTIAN S. G. PATRICK NASIRA—2017730089

1 Data Skripsi

Pembimbing utama/tunggal: **Raymond Chandra Putra, S.T., M.T.**

Pembimbing pendamping: -

Kode Topik : **RCP5103**

Topik ini sudah dikerjakan selama : **1 semester**

Pengambilan pertama kali topik ini pada : Semester **51 - Ganjil 21/22**

Pengambilan pertama kali topik ini di kuliah : **Skripsi 1**

Tipe Laporan : **B** - Dokumen untuk reviewer pada presentasi dan **review Skripsi 1**

2 Latar Belakang

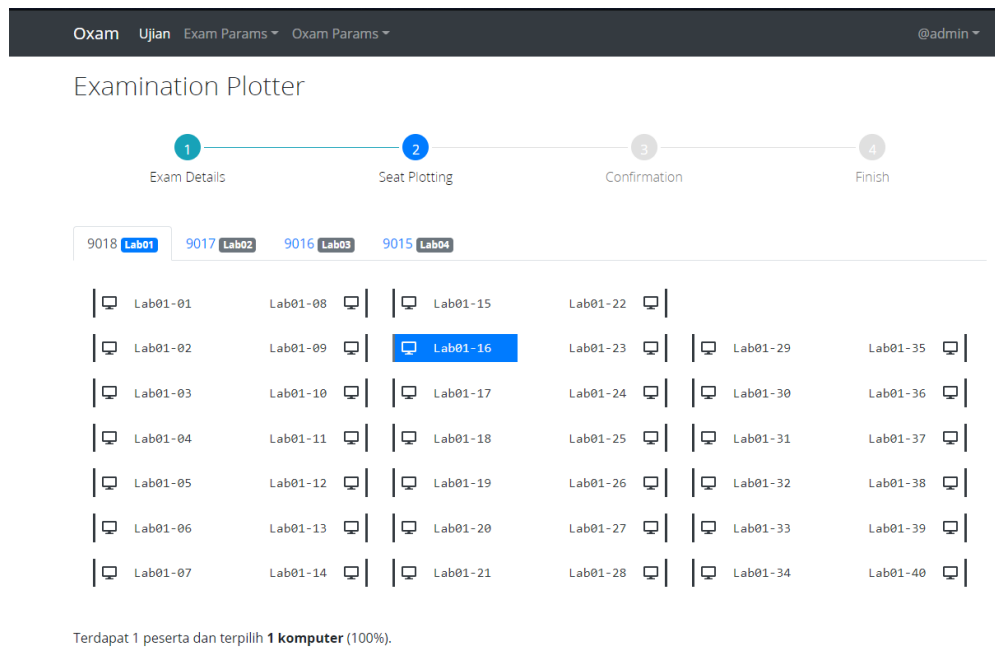
Laboratorium Komputasi Fakultas Teknologi Informasi dan Sains (FTIS) merupakan fasilitas yang disediakan untuk menunjang kegiatan perkuliahan yang bersifat praktik di lingkungan FTIS UNPAR. Sebagai contoh, kegiatan yang ditunjang adalah ujian praktik di masa UTS dan UAS bagi jurusan Informatika, Matematika, dan Fisika. Pihak-pihak yang terlibat dalam kegiatan ujian praktik ini, antara lain Admin Lab yang bertugas untuk mempersiapkan ruang ujian dan membantu apabila ada kendala pada saat ujian berlangsung, Koordinator Ujian yang bertugas untuk mengkoordinasikan jalannya ujian, Pengawas Ujian yang bertugas mengawasi jalannya ujian, serta Peserta yang akan mengikuti ujian.

Terdapat cukup banyak kendala teknis pada persiapan dan pelaksanaan ujian. Guna menangani kendala-kendala ini dibuatlah sebuah aplikasi yang bernama OXAM. OXAM merupakan sebuah aplikasi manajemen ujian di Laboratorium Komputasi FTIS atau Lab. FTIS yang dirancang untuk mempermudah Admin, Koordinator Ujian dan Pengawas untuk mempersiapkan hal-hal yang bersifat administratif terkait dengan ujian. Beberapa contoh kegiatan administratif yang kini sudah dapat ditangani menggunakan OXAM adalah: (1) pemilihan posisi duduk peserta ujian oleh Admin secara interaktif disertai pembuatan daftar hadir peserta ujian dan *script* penyebaran bahan ujian; (2) sinkronisasi waktu ujian, penambahan, maupun pengubahan waktu ujian melalui OXAM; (3) pemindahan posisi ujian peserta; (4) fitur notifikasi hal-hal terkait ujian melalui OXAM; (5) pengarsipan hasil ujian hingga pengiriman hasil ujian kepada dosen matakuliah yang dilakukan melalui OXAM.

Pada saat topik ini dikeluarkan, OXAM telah dikembangkan hingga versi yang kelima. Topik ini (RCP5103) merupakan topik lanjutan dari topik dengan kode RCP4703 yang dikerjakan oleh Gunawan Christianto (2016730011). Pengerjaan dari topik dengan kode RCP4703 telah menghasilkan sebuah sistem yang lebih baik dari versi sebelumnya. Hal ini dapat dilihat dengan adanya tampilan UI OXAM yang lebih interaktif dan tambahan fitur-fitur yang sudah disebutkan sebelumnya. Terdapat beberapa saran dan masukan yang diberikan pada saat sidang untuk topik dengan kode RCP4703. Beberapa saran dan masukan tersebut secara umum terletak pada pengaturan tempat duduk peserta ujian.

Gambar 1 merupakan tampilan pada OXAM ketika Admin melakukan pemilihan posisi ujian peserta. Pemilihan posisi ujian dilakukan secara manual dengan cara memilih satu per satu komputer yang akan digunakan. Komputer-komputer yang sudah dipilih kemudian digunakan untuk menempatkan peserta ujian secara acak. OXAM kemudian akan menghasilkan *script* berdasarkan hasil pemilihan acak tersebut. *Script* yang dihasilkan nantinya akan dijalankan di *server* untuk mendistribusikan berkas ujian.

Berdasarkan ilustrasi di atas, terdapat beberapa usulan pengembangan pada mekanisme pemilihan posisi ujian. Usulan berfokus pada pencegahan terjadinya *human error* pada saat pemilihan komputer yang dilakukan satu per satu yang sebenarnya dapat diotomatisasi berdasarkan jumlah peserta ujian. Selain itu,



Gambar 1: Pemilihan posisi duduk peserta ujian yang dilakukan secara manual.

tampilan pada saat akan memilih posisi tidak menggambarkan kondisi komputer secara *real time*. Pencatatan kondisi komputer dilakukan secara manual dan terpisah dari sistem. Hal ini mengakibatkan komputer yang dipilih mungkin saja berada dalam kondisi rusak, kosong tidak tersedia, memiliki masalah jaringan, atau masalah teknis lainnya yang dapat mengakibatkan kendala pada saat pelaksanaan ujian. Dengan adanya pemeriksaan kondisi komputer secara *real time* yang terintegrasi dengan sistem, masalah-masalah tersebut dapat terdeteksi lebih awal. Pemilihan secara acak pun dapat disesuaikan dengan ketersediaan komputer yang ada.

Kekurangan lainnya terkait dengan posisi komputer dan juga *layout* tiap ruang lab ujian yang dibuat secara *hardcode*. Jika terjadi perpindahan ruang lab yang mengakibatkan berubahnya *layout* ruangan, maka aplikasi menjadi tidak dapat digunakan. Pengembangan diharapkan dapat mempermudah penambahan atau pergantian *layout* pada aplikasi OXAM yang dilakukan secara interaktif.

OXAM saat ini hanya dapat digunakan untuk menangani ujian yang dilakukan secara langsung di ruangan lab. Hal ini dikarenakan, otentikasi yang digunakan berdasarkan *IP Address* dari komputer yang ada di ruangan lab. *IP address* dari komputer yang melakukan akses ke *endpoint* ujian akan dicocokkan dengan kumpulan *IP Address* komputer lab yang ada di *database*. Jika *IP Address* termasuk di dalam kumpulan *IP Address* maka *request* akan diproses lebih lanjut. Untuk mengatasi masalah ini, maka akan dibuat mode otentikasi berbasis token. Dengan adanya mode berbasis token, *login* dapat dilakukan tanpa bergantung pada *IP address* komputer serta peserta dapat mengikuti ujian dari luar ruangan lab. Dengan adanya mode berbasis token kegunaan OXAM dapat bertambah selain digunakan untuk UTS dan UAS, yakni digunakan pula untuk kuis maupun latihan.

Pada skripsi ini akan dilakukan pengembangan pada OXAM dengan lebih berfokus pada pengembangan mekanisme pemilihan posisi ujian, pengaturan *layout* yang dibuat agar lebih interaktif, dan penambahan mode berbasis token.

3 Rumusan Masalah

Berdasarkan deksripsi yang sudah dipaparkan di bagian 2, terdapat beberapa hal yang menjadi poin utama untuk dijadikan sebagai rumusan masalah pada skripsi ini. Beberapa hal tersebut adalah:

1. Bagaimana mekanisme yang diperlukan untuk menangani pemilihan posisi peserta ujian dengan tetap memperhatikan keadaan komputer secara *real time* dan manajemen ruang pada laboratorium komputasi FTIS secara interaktif?
2. Bagaimana mekanisme yang diperlukan untuk menerapkan mode otentikasi berbasis token dalam penggunaan OXAM?
3. Bagaimana implementasi mode otentikasi berbasis token pada OXAM?
4. Bagaimana implementasi untuk menghasilkan sistem yang dapat melakukan pemilihan posisi peserta ujian dengan tetap memperhatikan keadaan komputer secara *real time* dan sistem manajemen ruang pada laboratorium komputasi FTIS secara interaktif?

4 Tujuan

Berdasarkan bagian 2 dan bagian 3, dapat ditetapkan beberapa hal yang menjadi tujuan dari skripsi ini.

1. Menghasilkan spesifikasi kebutuhan dan rancangan untuk membuat sistem pemilihan posisi peserta ujian dengan memperhatikan keadaan atau kondisi *real time* dari komputer dan sistem manajemen ruang pada laboratorium komputasi FTIS secara interaktif .
2. Menghasilkan rancangan untuk menerapkan mode otentikasi berbasis token pada OXAM.
3. Menghasilkan sistem pemilihan posisi ujian peserta secara acak dengan memperhatikan keadaan atau kondisi *real time* dari komputer dan sistem manajemen ruang laboratorium komputasi FTIS yang dapat dilakukan secara interaktif menggunakan *canvas* yang terintegrasi dengan OXAM.
4. Menghasilkan mode ujian dengan otentikasi berbasis token pada OXAM yang dapat berjalan tanpa mengganggu mode berbasis IP *Address*.

5 Detail Perkembangan Pengerjaan Skripsi

Detail bagian pekerjaan skripsi sesuai dengan rencana kerja/laporan perkembangan terakhir :

1. **Mempelajari cara kerja OXAM dan skripsi dari Gunawan Christianto (2016730011).**
Status : Ada sejak rencana kerja skripsi.
Hasil : Dilakukan dengan cara mempelajari arsitektur aplikasi yang digunakan pada OXAM v.5.0 yang dibuat oleh Gunawan Christianto. Studi dilakukan terhadap *tools* yang digunakan, sistem, serta alur dari program OXAM.

- Tools

Terdapat beberapa *tools* yang digunakan pada OXAM v.05, yaitu:

- Docker

Docker adalah sebuah *platform containerization* yang bersifat *open-source*.¹ *Containerization* adalah pengemasan kode perangkat lunak dengan *library* dan dependensi sistem operasi (OS) yang diperlukan untuk menjalankan kode untuk membuat satu *executable lightweight*—disebut *container*—yang berjalan secara konsisten di infrastruktur apa pun. Lebih portabel dan hemat sumber daya daripada mesin virtual (VM), *container* telah menjadi unit komputasi *de facto* dari aplikasi *cloud-native* modern.²

Dengan adanya *containerization*, *developer* dapat membuat dan melakukan *deploy* aplikasi secara cepat dan aman. Pada penggunaan metode tradisional, kode dikembangkan dalam lingkungan komputasi tertentu dan ketika dipindahkan ke lokasi baru sering mengakibatkan *bug* dan kesalahan. Misalnya, ketika pengembang melakukan transfer kode dari komputer *desktop* ke mesin virtual (VM) atau dari Linux ke sistem operasi Windows. *Containerization* mengeliminasi masalah ini dengan menggabungkan kode aplikasi bersama dengan *file* konfigurasi terkait, *library*, dan dependensi yang diperlukan untuk menjalankannya. Paket perangkat lunak atau "wadah" tunggal ini diabstraksikan dari sistem operasi *host*, sehingga dapat berdiri sendiri dan menjadi portabel—dapat dijalankan di platform atau *cloud* apa pun, bebas dari masalah.³

Selain terkait dengan portabilitas dari *container*, *resource* seperti memori yang digunakan oleh *container* cenderung lebih ringan. Hal ini dikarenakan alokasi diberikan sesuai dengan kebutuhan *container*. Berbeda dengan VM yang memorinya dialokasikan dengan ukuran tertentu sejak awal dan kemudian dikunci, sehingga kelebihan alokasi menjadi tidak terpakai.

Integrasi dari perubahan pada *image* yang digunakan oleh *container* pun terbilang lebih mudah, karena perubahan cukup dilakukan pada file konfigurasi yang digunakan untuk membuat *image*. *Developer* tidak lagi harus melakukan penambahan secara manual, karena sudah dilakukan secara otomatis oleh *docker*. *Docker* juga mendukung penggunaan CI/CD yang memungkinkan untuk dilakukan *tracking* pada versi *image* yang dirubah.

```

1  # Change the image version if you want to test it on other php version.
2  # versions are listed on https://hub.docker.com/_/php?tab=tags
3  # always use the *-apache tags. It have apache to serve the thing.
4  FROM php:7.3.8-apache
5
6  RUN apt-get update && apt-get install -y --fix-missing \
7      apt-utils \
8      gnupg
9
10 RUN echo "deb http://packages.dotdeb.org jessie all" >> /etc/apt/sources.list
11 RUN echo "deb-src http://packages.dotdeb.org jessie all" >> /etc/apt/sources.list
12 RUN curl -sS --insecure https://www.dotdeb.org/dotdeb.gpg | apt-key add -
13
14 RUN apt-get update && apt-get install -y libfreetype6-dev libjpeg62-turbo-dev libpng-dev \
15     libzip-dev \
16     libssl-dev libldap2-dev libicu-dev locales locales-all
17
18 WORKDIR /var/www/html
19
20 RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/bin/ --filename=composer
21

```

Gambar 2: Isi dari Dockerfile yang digunakan pada aplikasi OXAM. Menggunakan *base image* php:7.3.8-apache yang berisi Apache httpd Debian. Terdapat instruksi-instruksi yang digunakan untuk membuat *image*. Sebagai contoh, instruksi untuk melakukan *download package* tambahan yang dibutuhkan untuk Debian dari *repository* dotdeb.org dan sebagainya.

¹Sumber berasal dari : <https://www.ibm.com/topics/docker>. Diakses pada 15 Desember 2021.

²Sumber berasal dari: <https://www.ibm.com/topics/containerization>. Diakses pada 15 Desember 2021.

³Ibid.

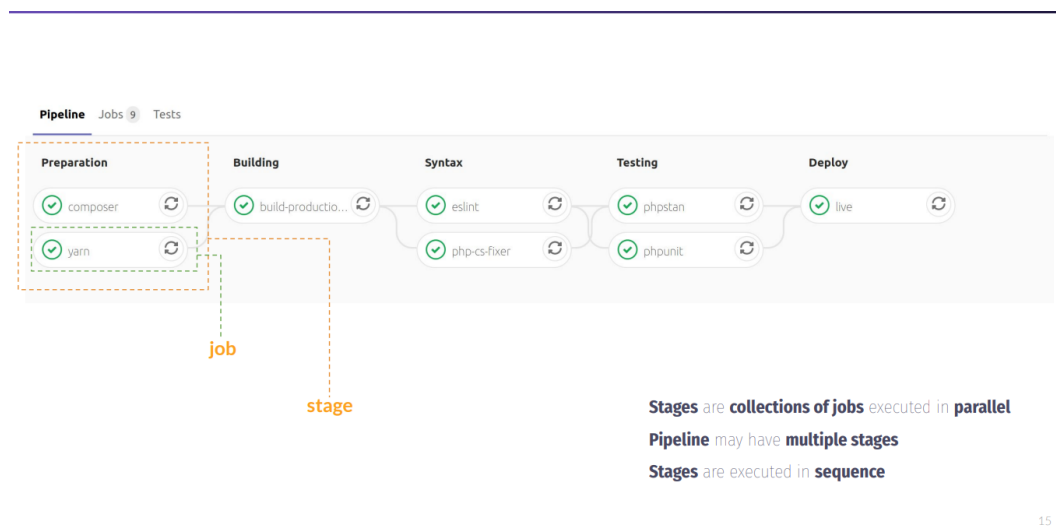
Docker menggunakan file konfigurasi yang diberi nama *Dockerfile* untuk membangun *image* yang dibutuhkan untuk membuat *container*. File konfigurasi ini berisi, *base image* yang akan digunakan serta runtutan instruksi yang dibutuhkan untuk membuat lingkungan yang dapat digunakan untuk menjalankan aplikasi secara lancar. Contoh dari *Dockerfile* yang digunakan pada aplikasi OXAM dapat dilihat di Gambar 2.

Selain *Dockerfile*, terdapat file konfigurasi lain bernama *docker-compose.yml* yang digunakan untuk menjalankan lebih dari satu kontainer sebagai sebuah *service*. Tiap *container* berjalan secara terisolasi, namun tetap dapat berinteraksi satu sama lain ketika dibutuhkan. Pada file ini, dapat ditentukan jenis-jenis *service* yang akan dijalankan, volume yang digunakan oleh *service*, *port* yang digunakan untuk menghubungkan *host* dengan *container*, dan konfigurasi-konfigurasi lainnya yang dibutuhkan oleh *service*.

– Gitlab *Continuous Integration/Continuous Delivery/Deployment (CI/CD)*

Gitlab CI/CD adalah *tool* pengembangan perangkat lunak yang menggunakan metodologi berkelanjutan.⁴ Terdapat tiga bagian utama dari CI/CD, yaitu:

- * *Continuous Integration*, dilakukan dengan cara memanfaatkan *script* untuk membangun dan melakukan uji coba pada aplikasi secara otomatis terhadap perubahan yang dilakukan oleh *developer*.
- * *Continuous Delivery*, berada selangkah setelah *Continuous Integration*. Selain melakukan uji coba, juga dilakukan *deploy* secara manual terhadap perubahan yang ada secara berkelanjutan.
- * *Continuous Deployment*, memiliki kemiripan dengan *Continuous Delivery*, namun *deploy* dari perubahan dilakukan secara otomatis.⁵



Gambar 3: Pada ilustrasi ini dapat dilihat, bagian terkecil di dalam sebuah *pipeline* adalah sebuah *job*. *Jobs* yang tersusun atau dijalankan secara paralel membentuk *stage*. Lalu kumpulan *stage* yang dieksekusi secara berurutan kemudian menjadi sebuah *pipeline*.

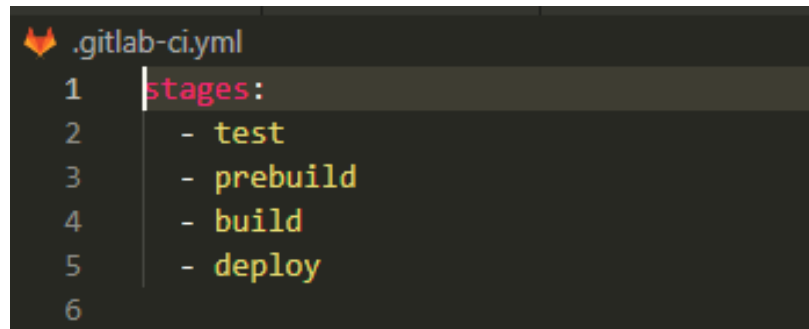
Terdapat beberapa istilah dasar terkait dengan penggunaan CI/CD, yaitu:

- * *Pipeline* yang merupakan kumpulan dari satu atau lebih *job*. Dapat dibagi menjadi kumpulan *stage*

⁴Sumber berasal dari: <https://docs.gitlab.com/ee/ci/>. Diakses pada 16 Desember 2021.

⁵Sumber berasal dari: <https://docs.gitlab.com/ee/ci/introduction/index.html#continuous-integration>. Diakses pada 16 Desember 2021.

- * *Stages* merupakan kumpulan *job* yang dijalankan secara paralel. Beberapa contoh *stage* yang umum digunakan, *Build*, *Test*, *Deploy*, dan sebagainya
- * *Jobs* dapat berupa *script* yang menjalankan tugas tertentu



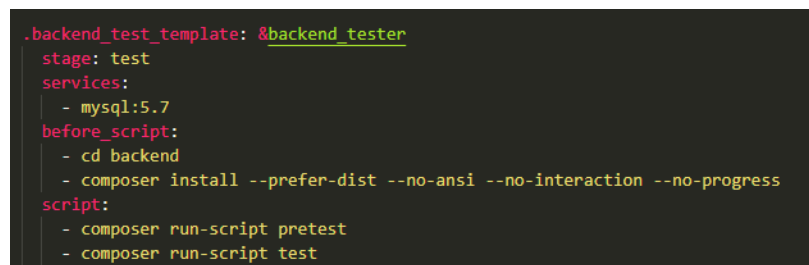
```

1 stages:
2   - test
3   - prebuild
4   - build
5   - deploy
6

```

Gambar 4: Beberapa *stage* yang ada pada CI/CD yang digunakan oleh OXAM

Konfigurasi CI/CD dilakukan dengan cara menyertakan *file* *.gitlab-ci.yml* pada direktori *root* dari aplikasi. Umumnya diawali dengan melakukan deskripsi dari *stage* yang ada pada pipeline. Contoh dari pendeskripsian *stage* pada *file* *.gitlab-ci.yml* dapat dilihat pada Gambar 4. *Stage* ini kemudian nantinya akan direferensikan pada *job* yang dideskripsikan.



```

.backend_test_template: &backend_tester
stage: test
services:
  - mysql:5.7
before_script:
  - cd backend
  - composer install --prefer-dist --no-ansi --no-interaction --no-progress
script:
  - composer run-script pretest
  - composer run-script test

```

Gambar 5: Berikut merupakan *job* *backend_test_template* yang masuk ke dalam *stage* *Test*.

Seperti yang terlihat pada Gambar 5, terdapat *before_script* dan *script* yang merupakan sebuah *task* yang akan dilakukan pada sebuah *job*. Terdapat beberapa jenis eksekusi *script*, yaitu:

- * *before_script*: dijalankan terlebih dahulu sebelum perintah yang ada pada *script* dijalankan
- * *after_script*: dijalankan ketika keseluruhan *job* selesai

Selain *script*, dapat disertakan pula jenis *service* yang digunakan pada sebuah *job*. Seperti yang terlihat pada Gambar 5, *service* yang digunakan untuk *job* tersebut adalah *service* basis data yang menggunakan *mysql* versi 5.7. Selain mencatumkan jenis *service* dapat pula dicantumkan jenis *image* yang digunakan untuk menjalankan sebuah *job*. *Image* yang dicantumkan akan digunakan oleh *Docker executor* untuk menjalankan *job* yang bersangkutan.

Jika dilihat kembali, pada Gambar 5 *job* *backend_test_template* diawali dengan tanda titik. Tanda ini digunakan untuk menandai sebuah *job* sebagai *hidden job*, sehingga tidak langsung dieksekusi. *Job* ini akan dieksekusi apabila diikutsertakan ke *job* lainnya dengan melakukan referensi ke *anchor* yang dimiliki oleh *hidden job*. Pada Gambar 5, *anchor* untuk *job* tersebut didefinisikan setelah tanda titik dua (:) dengan menambahkan tanda *ampersand* (&) sebelum nama dari *anchor* sesuai yang terlihat pada Gambar 6 (“&backend_tester”).

```

64 # test buat support php7.3
65 test:backend:php7.3:
66   image: edbizarro/gitlab-ci-pipeline-php:7.3-alpine
67   <<: *backend_tester
68   <<: *dont_run_things_in_this_job
69   cache:
70     key: php7-test-cache
71     paths:
72       - backend/vendor/
73

```

Gambar 6: Berikut merupakan *job* test:backend yang masuk ke dalam *stage* Test.

Pada *job* test:backend yang dapat dilihat pada Gambar 6, *job* backend_test_template diikutsertakan dengan cara mereferensikan *hidden job*. Untuk mereferensikan *hidden job* digunakan tanda *asterisk* diikuti dengan nama dari *anchor*. Tanda “<<” digunakan untuk melakukan *merge* isi dari *anchor* “backend_tester”.

- **Sistem yang digunakan oleh OXAM**

OXAM terdiri dari 2 buah subsistem, yang dirancang menggunakan library yang berbeda, yaitu:

- **Backend**

Back-end merupakan subsistem dari sebuah aplikasi *web* yang tidak terlihat dan tidak terhubung secara langsung dengan *client*. *Back-end* menjadi perantara yang menghubungkan infrastruktur seperti basis data dengan bagian *front-end* yang menangani *client*. Tugas dari *back-end* sendiri cukup bervariasi, seperti menangani *request* halaman *web*, melakukan pengolahan data baik untuk disimpan ataupun diperbaharui di basis data, menangani *upload* maupun *download* dari *file*, dan masih banyak lagi.

Proses data yang dilakukan oleh *back-end* dilakukan di sisi server. Umumnya *client* tidak akan menyadari ada subsistem ini, karena interaksi dilakukan hanya dengan *front-end*. Pada OXAM v.5.0, subsistem ini diimplementasikan menggunakan *framework* FatFree dan menggunakan arsitektur REST.

- **Frontend**

Berbeda dengan *back-end*, *front-end* merupakan subsistem yang berinteraksi secara langsung dengan *client*. Tugas dari *front-end* sendiri adalah untuk menampilkan data secara interaktif dan menarik kepada *client*, menangani input yang diberikan oleh *client* sesuai alur tertentu kemudian diteruskan ke *back-end* untuk diproses lebih lanjut. Karena terletak di sisi *client*, subsistem ini tidak menutup kemungkinan untuk dimanfaatkan sebagai celah untuk menyerang *server*. Sanitasi input yang diberikan dari bagian *front-end* ke *back-end* perlu dilakukan untuk mencegah sistem mengalami hal yang tidak diinginkan. Pada OXAM v.5.0, subsistem *front-end* diimplementasikan menggunakan *library* React.js.

2. Mempelajari React.js sebagai *library front-end*.

Status : Ada sejak rencana kerja skripsi.

Hasil : React.js merupakan *library* Javascript yang digunakan membuat *user interface*.⁶ Terdapat beberapa kelebihan yang dimiliki oleh *library* ini sehingga menjadi salah satu *library* yang banyak digunakan saat ini. Salah satu kelebihan dari React.js adalah sifatnya yang deklaratif. Pemrograman deklaratif berfokus untuk membangun *logic* dari program tanpa benar-benar mendeskripsikan alur yang dibutuhkan. React.js memiliki bentuk yang mirip dengan HTML. Bentuk deklaratif pada HTML

⁶<https://reactjs.org/>. Diakses pada 31 Desember 2021

sebagai contoh adalah ketika mendeklarasikan elemen `` agar browser menampilkan sebuah gambar. Proses atau *flow* agar gambar tersebut tampil di browser tidak menjadi fokus utama. Fokus utama yang ingin dicapai adalah “apa”, dalam hal ini sebuah gambar dengan *source* asal gambar `image.jpg`.

Sifat lain dari React.js adalah berbasis komponen. Pengguna dapat mendeklarasikan komponen dengan *state* tersendiri, kemudian membentuk komponen tersebut menjadi sebuah UI yang kompleks. Selain itu, React.js memungkinkan pengguna untuk “Pelajari sekali, tulis dimanapun” atau disebut juga “*Learn once, write anywhere*” (LOWA).⁷

Javascript XML (JSX)

React.js menggunakan sintaks Javascript XML atau disingkat JSX. Javascript XML merupakan turunan dari Javascript yang memungkinkan penggunaan HTML dalam Javascript. Untuk mendeklarasikan sebuah komponen React menggunakan sintaks JSX, pengguna dapat membuat elemen dalam bentuk yang mirip dengan elemen pada HTML. Sebuah elemen memiliki *tag* buka dan juga *tag* penutup. Sebuah *tag* diawali dengan tanda “<” dan diakhiri dengan tanda “>”. Untuk *tag* penutup setelah diawali dengan tanda “<” diikuti dengan tanda *slash* (/) lalu diakhiri dengan tanda “>”.



```

1  import ReactDOM from "react";
2  const pokemon = "Pikachu";
3  const pokemonVoice = "Pika, pika";
4  const element = (
5    <div>
6      
7      <h2>
8        {pokemon} (voice): {pokemonVoice}
9      </h2>
10     </div>
11   );
12
13   ReactDOM.render(element, document.getElementById("root"));
14

```

Gambar 7: Potongan kode yang menunjukkan penggunaan elemen HTML pada Javascript

Gambar 7 merupakan contoh potongan kode JSX, yang menggunakan elemen HTML *Heading 2* untuk mengeluarkan sebuah gambar diikuti dengan potongan teks `Pikachu (voice): Pika, pika` yang akan *render* ke elemen HTML dengan id *root*. Penggunaan elemen HTML yang memerlukan atribut tertentu tetap dimungkinkan dalam penggunaan JSX. Seperti yang terlihat pada Gambar 7, penggunaan elemen HTML *img* yang memerlukan atribut *src* akan *render* secara normal.

Komponen pada React

Pada React.js terdapat 2 cara untuk membuat komponen React, yaitu dengan mendeklarasikan komponen sebagai sebuah *class* dan menjadikan *class* tersebut turunan dari `React.Component` (*class component*) atau dengan mendeklarasikannya sebagai sebuah fungsi (*function component*).

⁷<https://reactjs.org/>. Diakses pada 31 Desember 2021



Gambar 8: Potongan kode yang menunjukkan deklarasi *functional component* `Pokeball0` dan *class component* `Pokeball1`.

Jika dilihat pada Gambar 8, *functional component* memiliki cara deklarasi komponen yang lebih *simple* dibandingkan dengan *class component*. Kedua komponen sama-sama menerima input masukan melalui `props` sebagai parameter pada *functional component* dan sebagai atribut pada *class component*.

State pada React

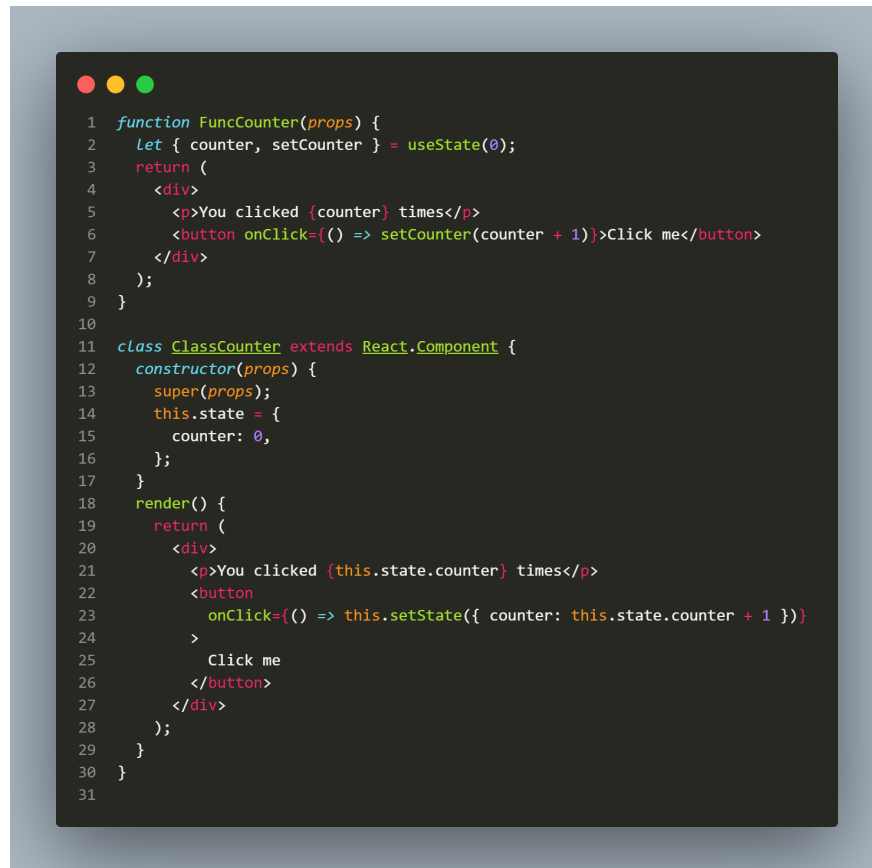
Pada Gambar 8 terlihat input untuk komponen dimasukkan melalui `props`. `Props` (singkatan dari *properties*) dan `state` merupakan objek pada Javascript. Keduanya menyimpan informasi yang mempengaruhi *output* hasil *render*, namun memiliki perbedaan yang cukup penting untuk diperhatikan; `props` dimasukkan sebagai input ke komponen (sama seperti parameter pada fungsi), sedangkan `state` dikelola di dalam komponen (mirip dengan variabel yang dideklarasikan dalam suatu fungsi).⁸

`Props` memiliki sifat yang *immutable* atau tidak dapat diubah-ubah. Berbeda dengan `state` yang merupakan *observable object* yang menyimpan data yang nilainya dapat berubah sewaktu-waktu. Karena dikelola di dalam komponen, `state` dapat digunakan untuk mengontrol perilaku (*behavior*) dari komponen. Perubahan pada `state` mengakibatkan *render* ulang dilakukan pada komponen.

Sebelum adanya React versi 16.8, penggunaan `state` tidak dimungkinkan pada *functional component*. Penggunaan `state` hanya dimungkinkan dengan mendeklarasikan *class component*. Pada versi 16.8, `state` dan fitur lainnya pada React dapat digunakan tanpa perlu mendeklarasikan *class component* (menggunakan *functional component*). Pada *class component* `state` masih dikelola sebagai atribut, sedangkan pada *functional component* `state` dikelola menggunakan *hook*. *Hook* merupakan fungsi spesial pada React yang memungkinkan untuk menghubungkan fungsi ke fitur React. *Hook* hanya dapat digunakan pada *functional component*.⁹

⁸<https://reactjs.org/docs/faq-state.html>. Diakses pada 31 Desember 2021

⁹<https://reactjs.org/docs/hooks-state.html>



Gambar 9: Potongan kode yang menunjukkan deklarasi *functional component* Pokeball10 dan *class component* Pokeball11.

Gambar 9 menunjukkan pengelolaan **state** pada *functional component* dan *class component*. Kedua komponen akan melakukan *render* sebuah tombol dan juga teks yang akan menampilkan jumlah aksi tekan yang sudah dilakukan pada tombol. Setiap kali tombol ditekan, kedua komponen akan melakukan *render* ulang dengan nilai **counter** yang sudah diperbaharui. *Hook useState* digunakan untuk mengelola **state** pada *functional component*.

3. Mempelajari FatFree sebagai *framework* yang digunakan untuk *back-end*.

Status : Ada sejak rencana kerja skripsi.

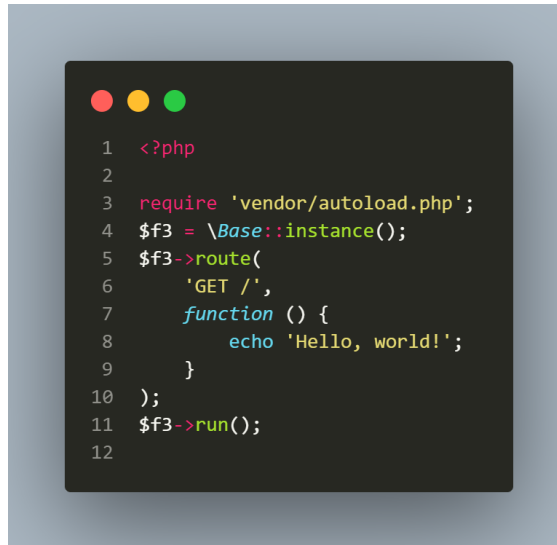
Hasil : Fatfree merupakan *micro-framework* yang *powerful* namun mudah digunakan untuk membangun aplikasi *web* yang dinamis.¹⁰ Fatfree sesuai dengan yang diimplikasikan oleh namanya *fat free* memiliki ukuran yang terbilang cukup kecil (diringkas dalam satu *file* berukuran 65KB) memungkinkan *developer* untuk menciptakan aplikasi *web* dengan eksekusi yang lebih cepat.

Fatfree merupakan salah satu *framework* dengan dokumentasi yang sangat baik. Selain itu *effort* yang diperlukan untuk mempelajari *framework* ini hampir tidak ada. FatFree memberikan kebebasan untuk menyelesaikan lebih banyak pekerjaan dengan waktu yang lebih sedikit.¹¹ Penggunaan *composer* bersamaan dengan Fatfree mendukung pengelolaan *package* yang lebih terpusat. Dengan menambahkan Fatfree sebagai *dependencies* pada *file composer.json* dan memanggil perintah *composer install*, *package* yang dibutuhkan terkait dengan Fatfree akan diunduh secara otomatis.

¹⁰<https://github.com/bcosca/fatfree>. Diakses pada 20 Desember 2021.

¹¹<https://fatfreeframework.com/3.7/getting-started>. Diakses pada 20 Desember 2021.

Hello World from Fatfree

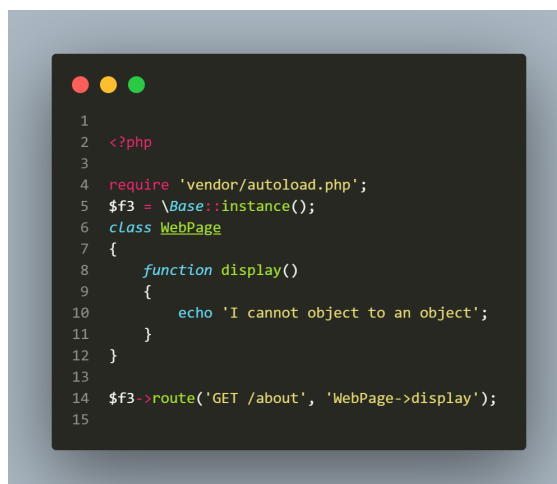
A screenshot of a code editor with a dark background and light-colored text. The code is PHP and uses the Fatfree framework. It includes a file 'vendor/autoload.php', creates an instance of the Base class, and defines a route for the GET method on the root path, which echoes 'Hello, world!'.

```
1  <?php
2
3  require 'vendor/autoload.php';
4  $f3 = \Base::instance();
5  $f3->route(
6      'GET /',
7      function () {
8          echo 'Hello, world!';
9      }
10 );
11 $f3->run();
12
```

Gambar 10: Potongan kode *Hello World* menggunakan Fatfree. Penggunaan bersamaan dengan *composer* memudahkan *include file* yang dibutuhkan oleh sebuah proyek cukup dengan hanya mengikutsertakan *file* *autoload.php*

Potongan kode pada Gambar 10, dilakukan definisi sebuah halaman yang dapat diakses melalui URL *slash* (/). Ketika *client* melakukan akses ke halaman yang terletak pada `http://127.0.0.1/` akan ditampilkan tulisan “Hello, world!”. Contoh pada Gambar 10, merupakan *route* yang akan menangani *request* menggunakan metode HTTP GET. Metode HTTP *request* lainnya dapat diimplementasikan dengan menyesuaikan parameter yang diberikan pada bagian `$f3->router()`.

Request ke sebuah *path* dapat ditangani dengan menyertakan fungsi yang menjadi *handler* seperti yang dapat dilihat pada Gambar 10. Penerapan dengan cara ini dapat mengacaukan *namespace global* dengan nama fungsi. Fatfree memungkinkan *developer* untuk melakukan *mapping* routing antara sebuah *path* dengan Kelas OOP dan juga method yang menjadi handler.

A screenshot of a code editor with a dark background and light-colored text. The code shows a class 'WebPage' with a 'display' method that echoes an error message. A route is then defined for the GET method on the path '/about', mapping it to the 'display' method of the 'WebPage' class.

```
1
2  <?php
3
4  require 'vendor/autoload.php';
5  $f3 = \Base::instance();
6  class WebPage
7  {
8      function display()
9      {
10         echo 'I cannot object to an object';
11     }
12 }
13
14 $f3->route('GET /about', 'WebPage->display');
15
```

Gambar 11: *Mapping* antara sebuah *path* */about* dengan kelas *WebPage* dengan *method* *display()* sebagai *handler*.

Penggunaan *mapping* seperti pada Gambar 11 memungkinkan desain kelas yang lebih seragam. Selain itu, perubahan pada fungsi dilakukan pada kelas yang bersangkutan, tidak bertebaran dan memudahkan *developer* untuk melakukan *maintain* pada kode.



Gambar 12: Token @count yang disertakan sebagai parameter pada URL *request*.

Seperti yang dapat dilihat pada Gambar 12, token dapat disertakan sebagai bagian dari sebuah URL. Token ini dapat diakses melalui *global array* PARAMS pada variabel \$f3. Ketika *client* melakukan *request* ke URL `http://127.0.0.1/deer/8` akan dikembalikan keluaran “8 deers in the jungle”. Ketika *client* melakukan *request* ke URL `http://127.0.0.1/deer/majestic` akan dikembalikan keluaran “ majestic deers in the jungle”.

4. Mengumpulkan kebutuhan dan melakukan analisis dampak.

Status : Ada sejak rencana kerja skripsi.

Hasil : Pada bagian ini, kebutuhan untuk penelitian dikumpulkan dengan melihat kendala yang ada pada OXAM versi 5.0 dan juga fitur-fitur yang dapat dikembangkan berdasarkan usulan yang sudah dibahas pada bagian 2.

- **Kebutuhan dan Analisis Dampak**

- **Kebutuhan**

Berdasarkan penjabaran yang ada pada bagian 2, terdapat beberapa hal yang dapat diperbaiki dan fitur yang dapat ditambahkan pada OXAM v.5.0. Beberapa hal tersebut diantaranya adalah sebagai berikut:

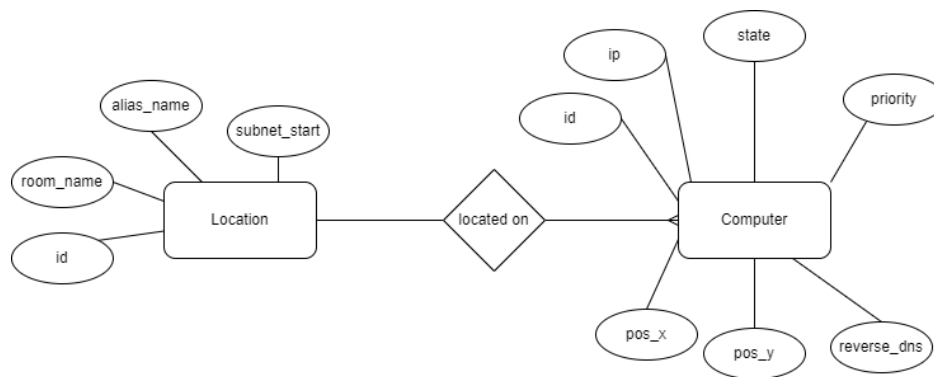
- (a) Manajemen ruang lab pada aplikasi OXAM yang dapat dibuat menjadi interaktif agar dapat menyesuaikan dengan perubahan layout ruangan.
- (b) Pemilihan posisi ujian yang dapat diotomatisasi, sehingga tidak perlu sepenuhnya dilakukan secara manual oleh Tim Admin.
- (c) Status komputer pada ruang lab dan aplikasi OXAM yang tidak sinkron/terintegrasi secara *real time*.
- (d) Penambahan mode ujian dengan otentikasi berbasis token yang mendukung pelaksanaan ujian dari luar ruangan lab.
- (e) Pemindahan posisi ujian peserta (*migrator*) yang disesuaikan dengan sistem manajemen ruang lab yang akan dibuat.

- **Manajemen Ruang dan Komputer**

Pada aplikasi OXAM v.5.0, pemetaan ruang lab dan komputer yang ada masih dilakukan secara *hardcode*. Data mengenai ruangan lab yang ada saat ini disimpan dengan memetakan secara manual. Sehingga apabila terjadi perubahan kedepannya terhadap *layout* dari ruang lab, maka aplikasi OXAM tidak dapat digunakan. Usulan yang diberikan adalah untuk membuat sebuah sistem manajemen ruang lab yang dapat digunakan untuk melakukan penambahan ruang lab, perubahan, dan penyimpanan yang dilakukan secara interaktif.

Kebutuhan untuk melakukan manajemen ruang ini secara spesifik dikhususkan untuk Tim Admin. Beberapa fitur yang minimal tersedia pada sistem ini setelah didiskusikan dengan Dosen Pembimbing adalah sebagai berikut:

- * Tim Admin dapat melihat *list* ruang lab yang ada
- * Tim Admin dapat menambahkan ruang lab yang baru
- * Tim Admin dapat melakukan *edit* pada detail mengenai ruangan
- * Tim Admin dapat melihat denah posisi komputer pada ruang lab yang bersangkutan
- * Tim Admin dapat melakukan perubahan pada komputer yang ada di denah ruang lab yang bersangkutan
- * Perubahan meliputi:
 - perubahan posisi komputer pada denah
 - perubahan status awal dari komputer pada denah
 - perubahan nilai prioritas komputer yang digunakan untuk pemilihan posisi ujian
 - perubahan nomor komputer
- * Tim Admin dapat menyimpan hasil perubahan pada komputer yang ada di denah ruang lab yang bersangkutan



Gambar 13: Entitas *Location* dan entitas *Computer*

Penambahan sistem manajemen ruang ini mengharuskan adanya perubahan pada bentuk entitas *Location* dan entitas *Computer* pada basis data. Terdapat penambahan 2 buah atribut pada entitas *Computer*, yaitu *state* dan *priority*. Atribut *state* digunakan untuk menandai status komputer di sebuah ruangan. Status sendiri terdiri dari nilai 0 atau 1. Nilai 0 berarti komputer tidak ditandai rusak oleh Tim Admin, sedangkan nilai 1 komputer ditandai sebagai komputer yang sedang rusak oleh Tim Admin.

Atribut *priority* nantinya akan digunakan untuk melakukan pemilihan posisi ujian secara otomatis. Atribut ini digunakan untuk menyimpan skala prioritas sebuah komputer dalam pemilihan posisi ujian. Komputer dengan skala prioritas lebih kecil akan didahulukan dalam pemilihan posisi ujian. Atribut lainnya yang berubah adalah *pos_x* dan *pos_y* yang sebelumnya disatukan sebagai atribut *d_pos* yang berisi koordinat x dan y dan disimpan dalam bentuk *string* JSON.

Pada entitas *Location* terdapat perubahan berupa penambahan atribut *subnet_start*. Atribut ini ditambahkan untuk mempermudah pengisian atribut *ip* pada sebuah komputer di ruangan. Sebagai contoh, pada ruangan dengan *subnet_start* 192.168.10.0, komputer dengan nomor 1 akan mendapatkan IP *address* 192.168.10.1. Pengisian nilai atribut *ip* ini akan dilakukan secara otomatis melalui sistem manajemen ruang lab ketika Tim Admin menambahkan komputer.

– Pemilihan Posisi Ujian secara Otomatis

Pada saat mempersiapkan ujian, Tim Admin akan melakukan pemilihan posisi ujian yang akan digunakan oleh peserta ujian. Pada OXAM v.5.0, pemilihan posisi ujian ini dilakukan secara manual satu per satu. Menurut usulan yang diberikan, hal ini sebenarnya dapat diotomatisasi. Selain mempercepat proses persiapan ujian, juga dapat mencegah adanya *human error* yang terjadi pada saat pemilihan posisi ujian.

The screenshot shows the 'Examination Plotter' interface. At the top, there is a progress bar with four steps: 1. Exam Details (active), 2. Seat Plotting, 3. Confirmation, and 4. Finish. Below the progress bar, there are several input fields: 'Tipe Ujian' (set to UTS), 'Mata Kuliah' (dropdown menu), 'Shift Ujian' (set to Tidak ada shift), 'Mulai pada' (date and time: 2021-12-20 16:18), and 'Selama' (duration: 120 Mnt). There is also a 'Quick Pick' section with buttons for 120 Mnt, 110 Mnt, and 105 Mnt. On the right side, there is a 'Peserta' section with a text box containing information about NPM standards and a link to the exam management page. At the bottom right, it says 'Total Peserta: 0 jiwa'.

Gambar 14: Langkah pertama dalam pembuatan ujian di Lab

Setelah Tim Admin menyelesaikan langkah pertama dengan mengisi *input* seperti yang terlihat pada gambar 14, Tim Admin akan diarahkan ke halaman *seat plotting* yang digunakan untuk memilih posisi komputer mana saja yang dapat ditempati oleh peserta ujian. Ketika berpindah ke langkah kedua, sistem akan melakukan *ping* ke komputer yang ada di lab yang terpilih secara *default*, kemudian melakukan pemilihan posisi berdasarkan prioritas komputer di lab tersebut.

Pemilihan juga memperhatikan *status* dari komputer, jika ditandai sebagai komputer rusak oleh Admin tidak akan diikutsertakan dalam pemilihan. Selain komputer dengan status ditandai rusak, pemilihan juga akan dilewati untuk komputer yang sudah terisi peserta dengan waktu ujian yang bersinggungan dengan ujian yang sedang dibuat. Pemilihan secara otomatis ini tidak bersifat absolut, dengan artian masih dapat diubah lagi oleh Tim Admin setelah pemilihan secara otomatis selesai dilakukan.

Sama seperti sistem manajemen ruang, fitur pemilihan posisi secara otomatis ini dikhususkan untuk digunakan oleh Tim Admin. Terdapat beberapa hal yang harus dapat dilakukan oleh Tim Admin, yaitu:

- * Sistem dapat melakukan *auto-select* berdasarkan konfigurasi status komputer di lab oleh Tim Admin, prioritas komputer, dan memperhatikan ketersediaan komputer terkait dengan ujian lain yang sedang berlangsung
- * Ketika Tim Admin melakukan pemilihan pada tab Lab, akan dilakukan *auto-select* posisi berdasarkan jumlah komputer yang tersedia dan juga jumlah peserta yang ada
- * Status komputer (*available, marked unavailable, not connected, occupied*)

- * Tim Admin dapat melakukan perubahan pada pemilihan posisi komputer yang dihasilkan oleh sistem

– Sinkronisasi Status Komputer di Ruang Lab dan OXAM

Sinkronisasi status komputer dilakukan oleh Tim Admin dengan menandai pada sistem status komputer yang bersangkutan sesuai dengan keadaan *real time*. Komputer yang sudah ditandai tidak tersedia, nantinya tidak akan diikutsertakan dalam pemilihan posisi ujian secara otomatis. Tim Admin juga dapat melakukan perubahan kembali untuk komputer yang ditandai tidak tersedia pada sistem. Sehingga dapat diikutsertakan kembali dalam mekanisme pemilihan posisi ujian secara otomatis.

– Pemindahan Posisi Ujian Peserta (Migrator)

Fitur pemindahan posisi peserta saat ini masih dilakukan dengan cara melakukan input secara manual posisi mahasiswa yang akan dipindahkan dan juga posisi tujuan. Fitur *migrator* ini dapat diubah disesuaikan untuk menggunakan mekanisme yang sama seperti pada saat melakukan pemilihan posisi ujian. Nantinya proses pemindahan posisi pada saat ujian berlangsung tidak dilakukan dengan mengetikkan NPM peserta, tapi dapat dilakukan secara interaktif. Pemindahan dapat dilakukan baik dengan cara *drag & drop* untuk pemindahan posisi di lab yang sama, atau dengan cara klik posisi asal dan tujuan di 2 lab yang berbeda.

– Mode Ujian Berbasis Token

Seperti yang juga sudah dibahas di bagian 2, ujian menggunakan OXAM saat ini hanya dapat dilakukan di ruangan lab. Hal ini dikarenakan otentikasi pada *endpoint* <https://oxam.labftis.net/exam> hanya akan menampilkan informasi ujian apabila diakses melalui komputer yang berada di lab (IP komputer yang mengirimkan *request* tercatat di *database*).

Mode ujian dengan otentikasi berbasis token ditambahkan untuk memungkinkan ujian dilakukan di luar ruang lab. Dengan demikian, waktu ujian menjadi lebih fleksibel. Ujian yang sifatnya dilakukan secara *shift* dapat dilakukan bersamaan. Selain itu, dengan adanya mode ini, fungsi OXAM tidak lagi terbatas hanya untuk mengatur ujian UAS maupun UTS, namun juga dapat digunakan untuk mengatur *quiz*.

Pada mode ujian dengan otentikasi berbasis token terdapat 3 aktor yang terlibat, yaitu tim admin, dosen, dan juga peserta ujian. Pada mode ini masing-masing aktor memiliki kebutuhan tersendiri, di antaranya:

- * Tim Admin
 - Tim Admin dapat membuat ujian baru tanpa melakukan pemilihan posisi ujian
 - Tim Admin dapat menyertakan file yang akan digunakan untuk ujian yang diberikan oleh Dosen yang mengampu Mata Kuliah yang akan diujikan
- * Dosen
 - Dosen dapat melakukan login dari luar untuk membuat ujian baru
 - Dosen dapat membuat ujian baru tanpa melakukan pemilihan posisi ujian
 - Dosen dapat menyertakan file yang akan digunakan untuk ujian
 - Fitur-fitur dari sistem Oxam lama seperti pengiriman notifikasi dan *timer screen* tetap berjalan
- * Peserta
 - Peserta ujian dapat melakukan login dari luar lab untuk melakukan ujian
 - Peserta ujian dapat melakukan pengunduhan file dari ujian yang akan/sedang berlangsung dari luar lab

- Peserta ujian dapat melakukan pengumpulan jawaban dari luar lab tanpa kendala
- Fitur seperti notifikasi dan timer yang terintegrasi di sistem yang lama tetap berjalan dan berfungsi pada mode berbasis token untuk peserta

Login untuk peserta ujian dan juga dosen akan dilakukan dengan mencocokkan data identitas seperti *username* dan *password* ke *Active Directory* (AD) dengan menggunakan protokol *Lightweight Directory Access Protocol* (LDAP). Jika data *login* yang disertakan sesuai dengan yang ada pada AD, maka user akan diberikan token untuk melakukan akses terkait dengan ujian.

5. Merancang dan mengimplementasi perangkat lunak.

Status : Ada sejak rencana kerja skripsi.

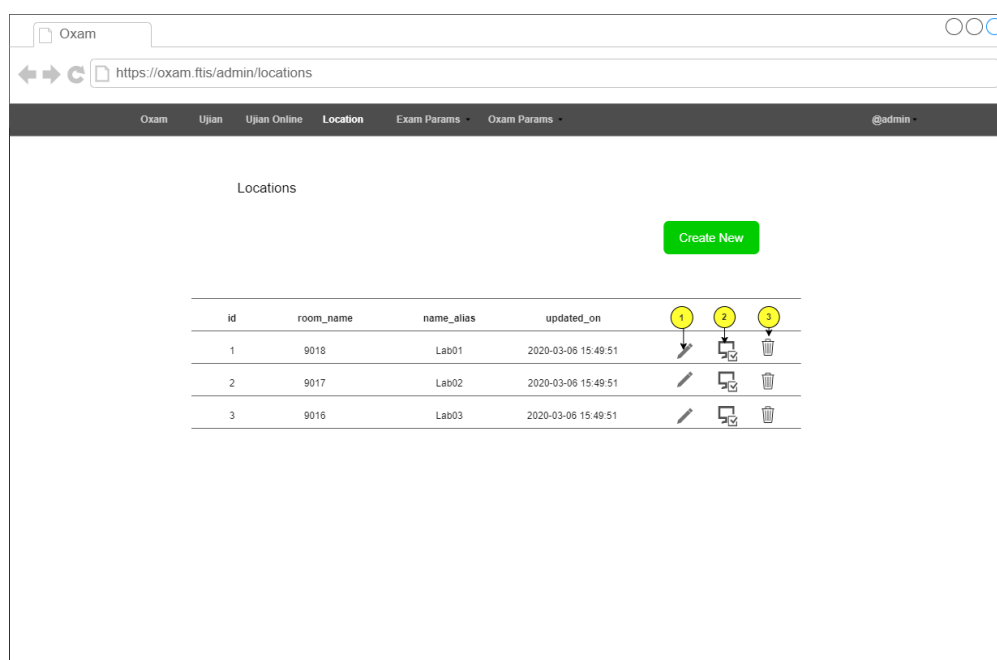
Hasil :

• Rancangan *User Interface* (UI)

Rancangan *interface* juga turut dibuat pada tahap ini untuk mempermudah identifikasi langkah-langkah yang diperlukan masing-masing proses pada fitur yang akan dibuat. Berikut merupakan rancangan *interface* yang dibuat untuk masing-masing fitur:

– Sistem Manajemen Ruang dan Komputer

Menu utama baru, yaitu *Location* ditambahkan pada *navigation bar* untuk memudahkan akses menuju halaman utama yang digunakan untuk melakukan manajemen ruang dan komputer. Sebelumnya, baik *location* maupun *computer* merupakan anak dari menu *Exam params*. Beberapa alasan menu ini ditambahkan sejajar dengan menu utama adalah (1) Implementasi fitur manajemen ruang menggunakan *endpoint* sebelumnya memisahkan *location* dari *computer*. Pada desain yang baru, ketika mengakses mengenai detail sebuah ruangan lab, maka akan ditampilkan denah dari komputer yang ada pada lab tersebut. Jika menggunakan desain *endpoint* yang lama (masuk menjadi anak dari *Exam params*) akan lebih sulit diaplikasikan; (2) Dengan memindahkan kedua menu tersebut ke satu menu utama, yaitu *Location* perubahan *endpoint* yang digunakan untuk fitur tersebut lebih bebas untuk diimplementasikan mengikuti pola *endpoint* yang digunakan oleh menu *Ujian*.



Gambar 15: Halaman utama dari menu *Location*.

Gambar 15 merupakan tampilan utama yang ada pada menu utama *Location*. Tampilan secara umum sama seperti tampilan yang digunakan *location* sebelumnya ketika masih menjadi submenu dari *Exam params*. Terdapat tombol *Create New* yang digunakan untuk menambahkan ruangan lab baru.

The screenshot shows a web browser window with the address bar displaying 'https://oxam.ftis/admin/locations/new'. The page has a dark navigation bar with links for 'Oxam', 'Ujian', 'Ujian Online', 'Location', 'Exam Params', and 'Oxam Params'. The user is logged in as '@admin'. The main content area is titled 'Locations::new()' and contains a form with the following fields:

- id**: A disabled text input field.
- room_name**: A text input field.
- alias_name**: A text input field.
- subnet_start**: A text input field.

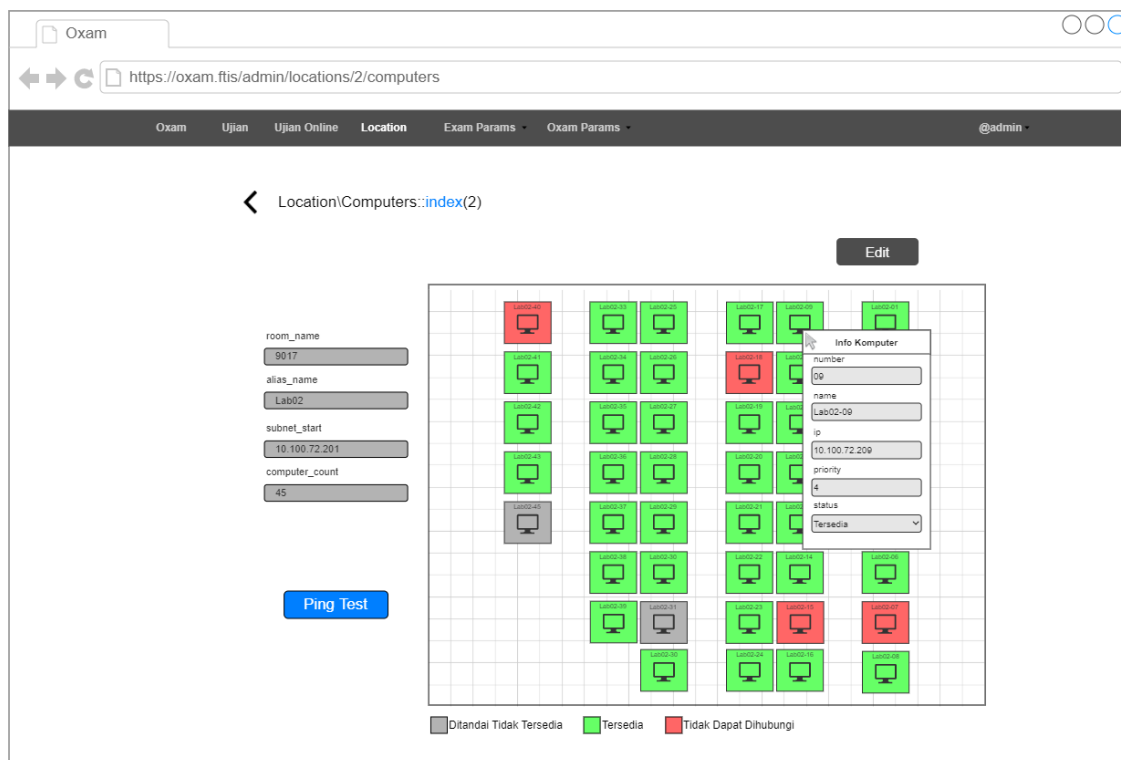
At the bottom of the form is a blue button labeled 'Save'.

Gambar 16: Halaman tambah ruangan baru.

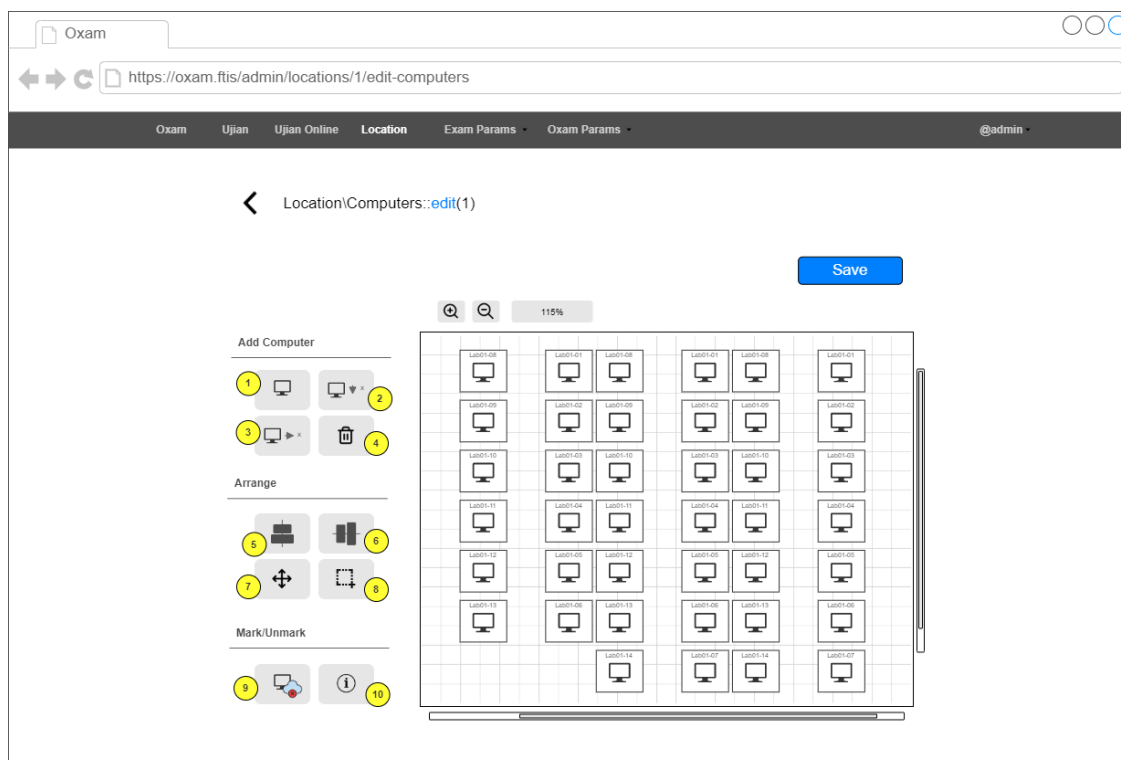
Terdapat beberapa *form* masukan yang dapat diisi, yaitu *room_name*, *alias_name*, dan *subnet_start*. Nilai dari *id* nantinya diisi secara otomatis ketika dimasukan ke basis data, sehingga sengaja dibuat *disabled*. Masukan *room_name* merupakan nomor ruangan dari sebuah ruang lab (contohnya 9017 atau 9018). Masukan *alias_name* adalah nama alias dari ruangan lab yang biasa dipakai oleh para Tim Admin (contohnya, Lab01 atau Lab02). Perbedaan yang ada pada halaman ini dibandingkan dengan halaman *new location* sebelumnya adalah penambahan masukan untuk *subnet_start*. Nilai dari *subnet_start* akan digunakan untuk mengisi data IP Address dari sebuah komputer secara otomatis, sehingga tidak perlu dituliskan secara manual ketika ditambahkan.

Tombol dengan label 1 pada Gambar 15, jika ditekan akan mengarahkan pengguna ke halaman *edit* untuk ruang lab yang dipilih. Perbedaan dari tampilan sebelumnya terletak pada tombol dengan *icon* komputer yang ditandai menggunakan label 2 pada Gambar 15. Ketika tombol tersebut ditekan, akan mengarahkan pengguna menuju halaman detail dari ruangan lab yang dipilih serta menunjukkan denah dari komputer yang ada pada ruangan lab. Tampilan tersebut dapat dilihat pada Gambar 17.

Pada halaman detail ruang lab terdapat tombol *Ping Test* yang dapat digunakan untuk melakukan *test ping* ke komputer yang ada di ruang lab tersebut. Terdapat beberapa indikator yang digunakan untuk menandakan status komputer pada sebuah ruangan lab. Komputer yang ditandai dengan *background* merah menandakan bahwa komputer tersebut tidak merespon ketika dilakukan *ping*. Komputer dengan *background* hijau menandakan bahwa komputer tersebut merespon ketika dilakukan *ping*. Sedangkan komputer dengan *background* berwarna abu-abu menandakan bahwa komputer tersebut ditandai oleh Tim Admin sebagai komputer yang tidak dapat digunakan (komputer rusak atau komputer secara fisik tidak ada).

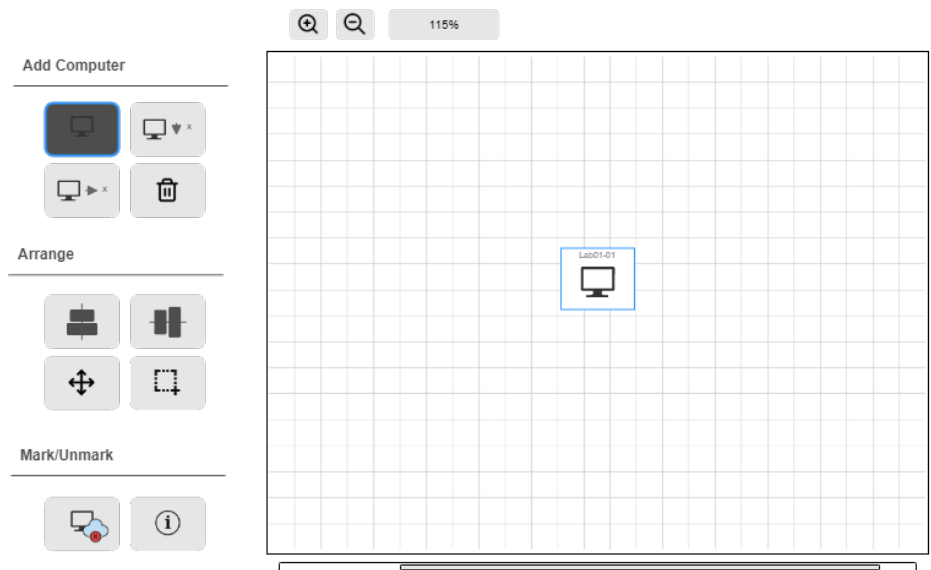


Gambar 17: Halaman detail komputer dari ruang lab 9017 atau Lab02.



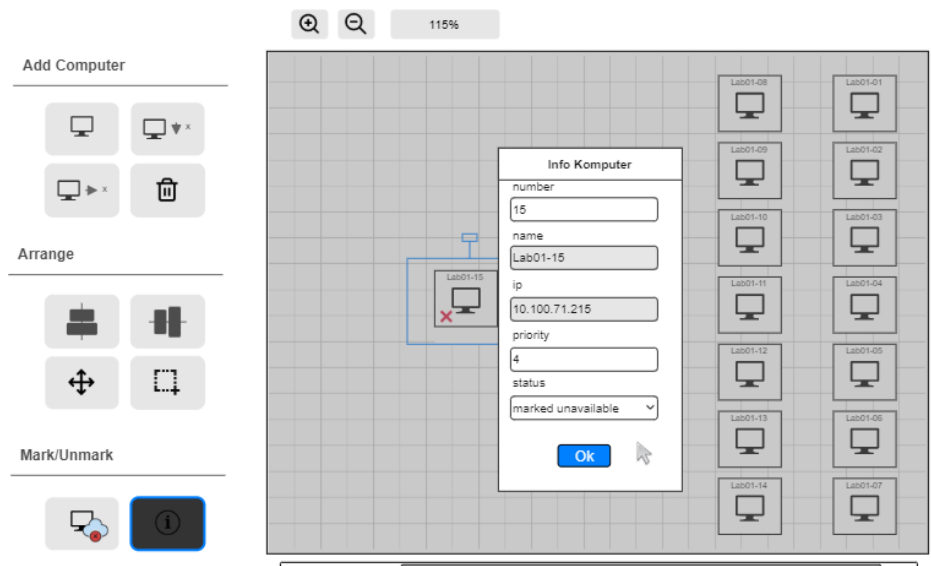
Gambar 18: Halaman edit ruangan lab secara interaktif.

Selain tombol *ping*, terdapat tombol *Edit* yang jika ditekan akan mengarahkan menuju halaman *edit layout* komputer di ruangan lab yang dipilih. Tampilan dari halaman untuk melakukan *edit layout* dari ruangan lab dapat dilihat pada Gambar 18. Terdapat beberapa tombol yang disediakan untuk memudahkan perubahan layout pada ruangan lab.



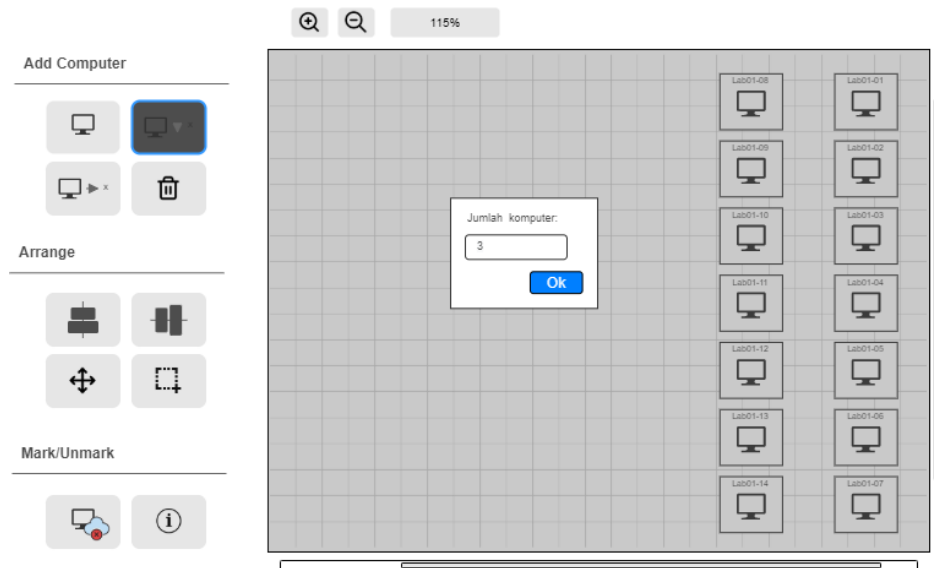
Gambar 19: Komputer yang ditambahkan secara otomatis ke tengah-tengah kanvas ketika tombol tambah komputer ditekan.

Tombol dengan label 1 pada Gambar 18 digunakan untuk menambahkan komputer ke dalam *layout* ruangan. Jika tombol ini ditekan, maka akan menambahkan secara otomatis sebuah komputer ke dalam denah seperti yang terlihat pada Gambar 19. Ketika komputer ditambahkan, data-data terkait dengan komputer tersebut seperti nomor komputer, *dns_reverse*/nama komputer, IP Address, dan koordinat dari komputer pada denah akan dihasilkan secara otomatis. Selain data-data tadi, status awal komputer akan dibuat menjadi *available*/tersedia dan diberikan nilai prioritas satu (1).

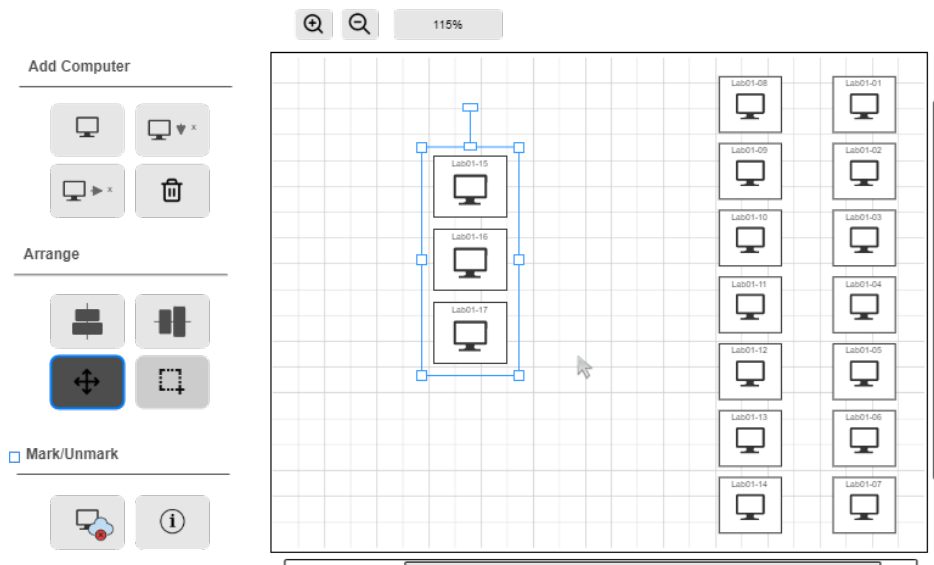


Gambar 20: Tombol *info* untuk melihat detail dari sebuah komputer pada denah.

Info dari sebuah komputer yang sudah ditambahkan ke denah dapat dilihat dengan cara menekan tombol info (tombol dengan label nomor 10 pada Gambar 18), kemudian memilih komputer yang akan dilihat infonya. Info dari sebuah komputer akan ditampilkan seperti yang terlihat pada Gambar 20. Pada bagian ini, perubahan dapat dilakukan pada detail data komputer dan akan disimpan ketika tombol “Ok” ditekan.



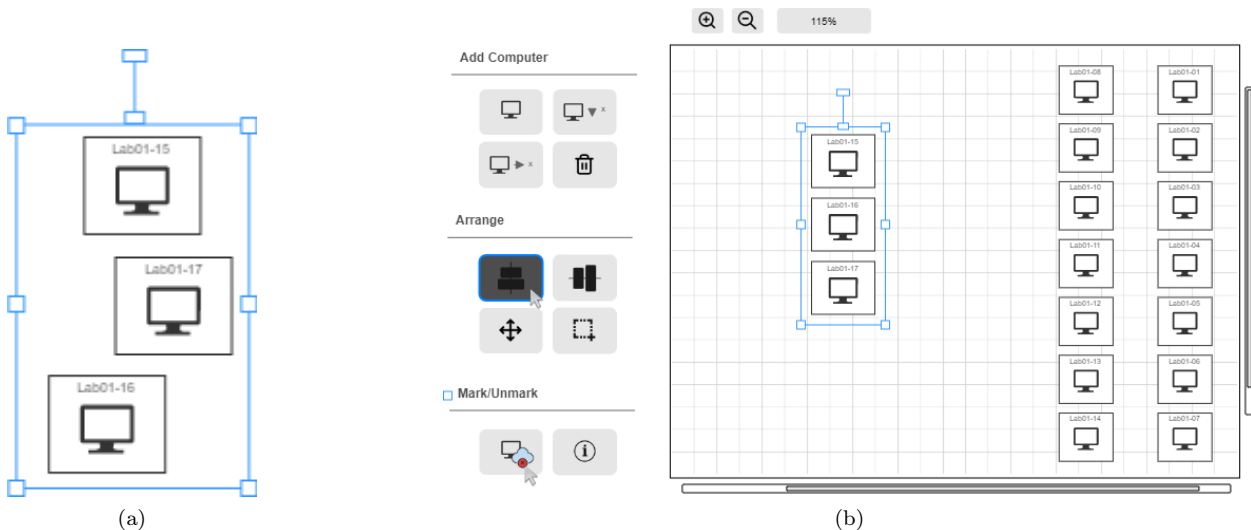
Gambar 21: Ketika tombol untuk menambahkan komputer dalam jumlah banyak ditekan, akan muncul *prompt* untuk memasukkan jumlah komputer yang akan ditambahkan.



Gambar 22: Tiga buah komputer yang ditambahkan dengan pilihan orientasi yang diatur secara vertikal.

Selain penambahan komputer satu per satu, terdapat pula tombol untuk menambahkan komputer dalam jumlah banyak. Pada Gambar 18, tombol dengan label nomor 2 digunakan untuk menambahkan komputer dalam jumlah banyak dengan susunan yang diatur secara vertikal. Ketika tombol ini ditekan maka akan memunculkan *prompt* untuk memasukkan jumlah komputer yang akan ditambahkan seperti yang terlihat pada Gambar 21. Setelah jumlah komputer yang akan ditambahkan diisi dan tombol “Ok” ditekan, maka komputer akan ditambahkan seperti yang terlihat pada Gambar 22.

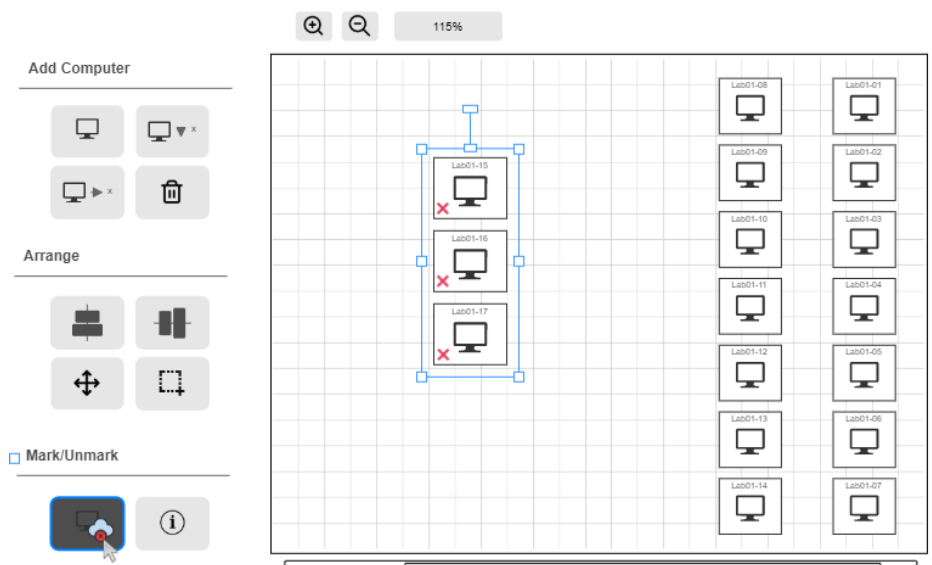
Terdapat pula tombol untuk menambahkan komputer dalam jumlah banyak dengan susunan yang diatur secara horizontal (tombol dengan label nomor 3 pada Gambar 18). Tombol dengan *icon trash can* atau tombol dengan label nomor 4 pada Gambar 18 digunakan untuk menghapus komputer dari denah.



Gambar 23: Tombol *align vertically* yang digunakan untuk merapihkan komputer yang dipilih dari keadaan pada Gambar 23a menjadi lebih rapih seperti yang terlihat pada Gambar 23b.

Selain tombol untuk menambahkan dan menghapus komputer, terdapat tombol-tombol yang dapat digunakan untuk mengatur posisi komputer secara berkelompok. Tombol dengan label nomor 5 dan 6 pada Gambar 18 memiliki fungsi secara berurut mengatur posisi komputer agar sejajar vertikal dan sejajar horizontal. Gambar 23 merupakan contoh pengaturan posisi komputer-komputer yang dipilih kemudian disejajarkan secara vertikal menggunakan tombol *align vertically*.

Kedua tombol lainnya yang digunakan untuk mengatur posisi adalah tombol *move* (tombol dengan label nomor 7 pada Gambar 18) dan tombol *select* (tombol dengan label nomor 8 pada Gambar 18). Fungsi tombol *select* digunakan untuk memilih sebuah komputer atau dapat pula memilih kumpulan komputer. Tombol *move* kemudian dapat digunakan untuk memindahkan komputer yang telah terpilih ke posisi baru.



Gambar 24: Tiga buah komputer yang terpilih ditandai sebagai komputer yang tidak dapat digunakan untuk ujian di ruangan lab.

Tombol yang terakhir adalah tombol *mark* yang dapat digunakan untuk memberi tanda kepada komputer sebagai komputer yang tidak dapat digunakan untuk ujian. Komputer-

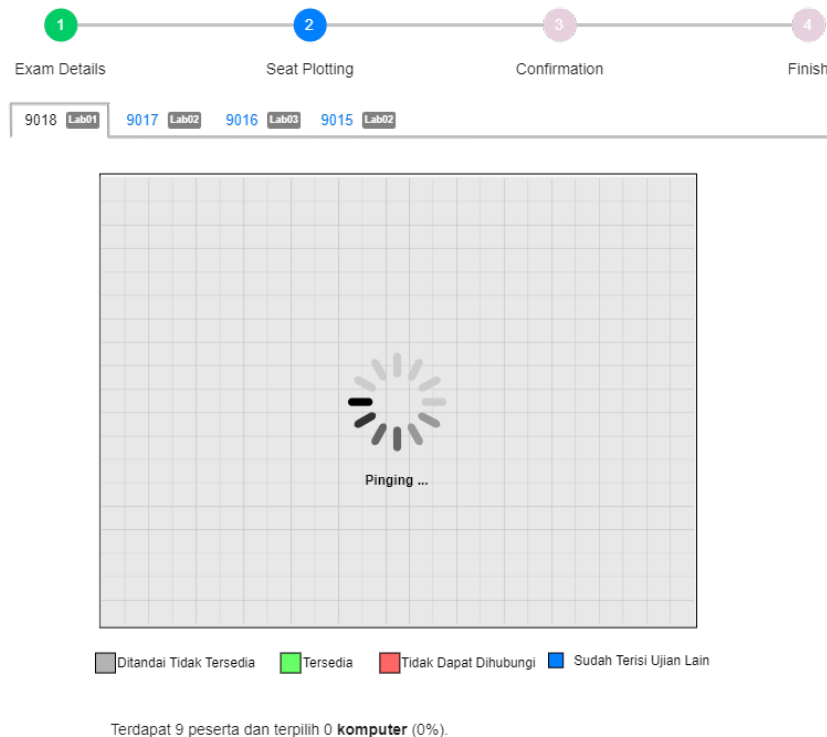
komputer yang terpilih berubah status menjadi *not available* ketika tombol ini ditekan seperti yang terlihat pada Gambar 24. Jika sebelumnya komputer sudah ditandai sebagai komputer yang tidak dapat digunakan (*not available*) dan tombol ini ditekan, maka tanda tersebut akan dihilangkan dan status komputer berganti menjadi dapat digunakan untuk ujian.

– Pemilihan Posisi Ujian secara Otomatis

Salah satu hal yang dilakukan oleh Tim Admin ketika mempersiapkan ujian di ruang lab adalah mendaftarkan entri ujian ke OXAM. Untuk mendaftarkan sebuah ujian Tim Admin akan mengisi *form* pembuatan ujian yang ada pada OXAM. Terdapat 4 tahap dalam pengisian *form* pembuatan ujian pada OXAM, yaitu pengisian detail ujian (*exam details*), pengalokasian tempat duduk untuk ujian (*seat plotting*), konfirmasi ujian (*confirmation*), dan penyelesaian (*finish*). Sistem *seat plotting* pada OXAM v.5.0 atau versi saat ini mengharuskan Tim Admin untuk melakukan pemilihan posisi ujian secara manual dengan memilih satu per satu komputer yang akan digunakan. Hal ini selain memakan waktu, juga rentan terhadap *human error*.

Pada rancangan *seat plotting* yang baru, akan diintegrasikan dengan pengaturan yang ada pada sistem manajemen ruang dan komputer. Pemilihan posisi kini akan dilakukan secara otomatis dan pemilihannya ditentukan berdasarkan nilai prioritas dari komputer serta status dari komputer yang sebelumnya telah diatur oleh Tim Admin sebelum pengisian entri ujian dilakukan.

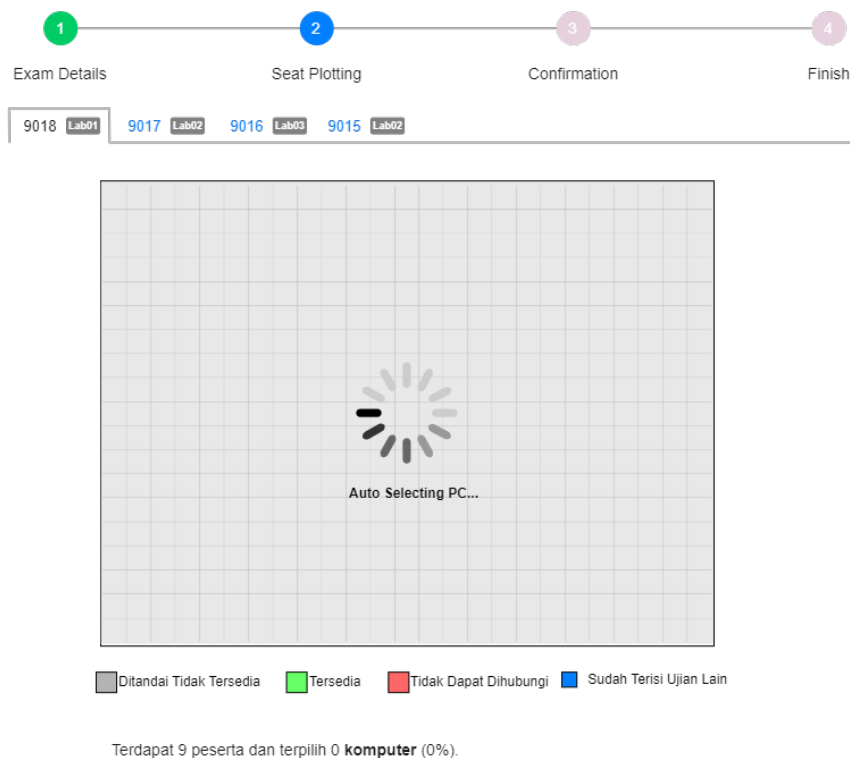
Examination Plotter



Gambar 25: Proses *ping* komputer yang berada di ruangan 9018 pada tahap kedua pembuatan entri ujian.

Tahap *seat plotting* akan diawali dengan proses *ping* ke komputer yang berada di sebuah ruang lab. Proses *ping* hanya akan dilakukan satu kali pada saat sebuah ruangan lab dipilih agar tidak memakan waktu terlalu lama. Seperti yang terlihat pada Gambar 25, tab untuk ruang 9018 terpilih secara *default*, sehingga akan dilakukan *ping* untuk komputer yang berada di ruangan ini.

Examination Plotter



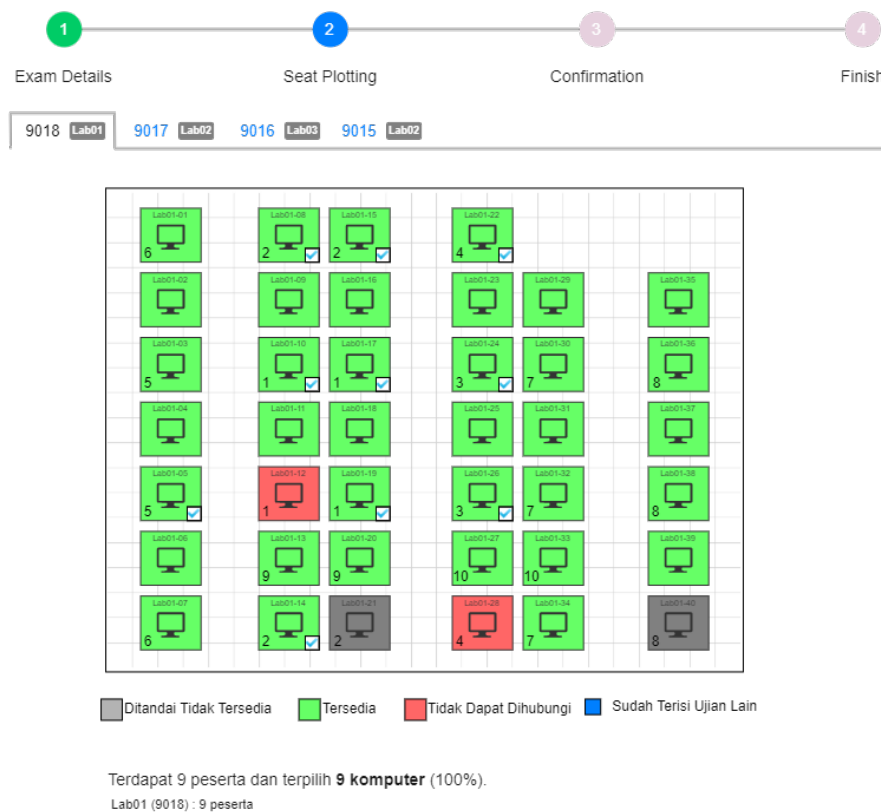
Gambar 26: Proses pemilihan posisi ujian secara otomatis berdasarkan nilai prioritas komputer dan juga status dari komputer.

Setelah proses ping selesai dilakukan, maka akan dilakukan proses pemilihan posisi ujian secara otomatis berdasarkan nilai prioritas dari komputer serta status dari sebuah komputer. Proses ini ditandai dengan *loading screen* yang terlihat pada Gambar 26. Nilai prioritas sebuah komputer dimulai dari nilai satu (1) sebagai prioritas tertinggi. Komputer yang tersedia dan belum terpilih dengan nilai prioritas rendah akan selalu diutamakan dalam pemilihan posisi ujian.

Terdapat 4 jenis status yang juga akan digunakan sebagai parameter untuk pemilihan posisi ujian. Komputer yang nantinya ditandai dengan *background* berwarna abu-abu merupakan komputer dengan status ditandai tidak tersedia (*marked not available*) oleh Tim Admin, sehingga secara otomatis tidak akan diikutsertakan dalam pemilihan posisi ujian. Komputer dengan *background* berwarna hijau merupakan komputer yang berhasil memberikan balasan pada proses *ping* dan merupakan kandidat yang dapat digunakan untuk pemilihan posisi ujian. Komputer dengan *background* berwarna merah merupakan komputer yang tidak memberikan balasan ketika proses *ping* dilakukan, sehingga tidak akan digunakan. Sedangkan komputer dengan *background* berwarna biru merupakan komputer yang sudah terisi ujian oleh peserta dari ujian lain dengan waktu dan durasi yang sama dengan ujian yang akan dibuat.

Gambar 27 merupakan tampilan setelah proses pemilihan posisi ujian secara otomatis selesai dilakukan. Komputer yang terpilih ditandai dengan tandan *checkbox*. Hasil pemilihan ini nantinya masih dapat diubah lagi oleh Tim Admin. Jika dilihat pada Gambar 27 komputer yang memiliki nilai prioritas (angka di sebelah kiri bawah komputer) terletak saling terpisah 1 komputer. Pengaturan kandidat untuk pemilihan posisi ujian sengaja dibuat seperti ini untuk mengatasi adanya tindakan curang pada saat ujian. Selain itu, dengan pengaturan posisi dengan selang satu (1) dapat memudahkan Pengawas ketika mengawasi ujian.

Examination Plotter



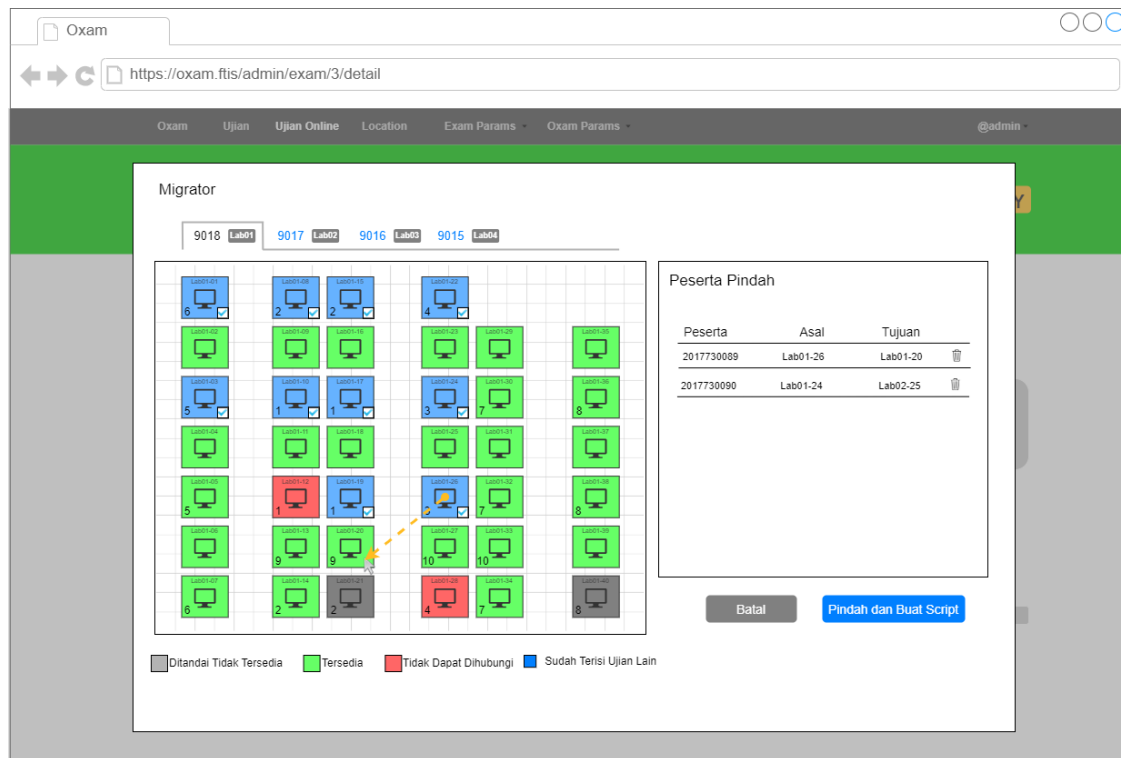
Gambar 27: Komputer yang terpilih untuk digunakan sebagai posisi ujian memiliki tanda *checkboxlist*.

Komputer dengan nilai prioritas 9 dan 10 pada Gambar 27 merupakan komputer yang diberikan sekat pada sisi kiri dan kanannya. Komputer yang diberikan sekat hanya akan digunakan apabila komputer dengan posisi berselang satu sudah penuh terisi. Pengaturan posisi dengan selang satu ini nantinya akan ditentukan Tim Admin melalui sistem manajemen ruang dan komputer dengan cara memberikan nilai prioritas pada komputer. Tiap lab memiliki pengaturan yang berbeda, hal ini dipengaruhi oleh jumlah komputer yang ada pada ruangan lab dan juga layout dari ruangan lab.

Jika diperhatikan lebih lanjut pada Gambar 27, komputer nomor 12 (Lab01-12) memiliki nilai prioritas satu (prioritas tertinggi). Namun komputer ini tidak dipilih, karena tidak memberikan respon balik pada proses *ping*. Begitu pula dengan komputer bernomor 21 (Lab01-21) yang tidak dipilih walaupun memiliki nilai prioritas 2 (prioritas tertinggi kedua), karena ditandai oleh Tim Admin sebagai komputer yang rusak atau komputer yang tidak tersedia secara fisik.

– Pemindahan Posisi Ujian Peserta (Migrator)

Pada saat ujian berlangsung, tidak jarang terjadi kendala seperti komputer yang mendadak bermasalah. Hal ini mengharuskan peserta ujian untuk pindah ke komputer lain agar dapat melanjutkan ujian. Pada OXAM v.5.0, terdapat *migrator* yang digunakan untuk memindahkan posisi ujian peserta pada saat ujian sedang berlangsung. Pada saat akan melakukan pemindahan posisi ujian peserta, Tim Admin akan menginput nomor NPM peserta yang akan dipindahkan dan komputer tujuan pemindahan.



Gambar 28: *Migrator* yang digunakan untuk memindahkan posisi ujian peserta pada saat ujian sedang berlangsung.

Pada fitur yang baru, pemindahan posisi ujian menggunakan migrator di ruangan yang sama dapat menggunakan *drag* dari posisi asal dan *drop* di posisi tujuan seperti yang terlihat pada Gambar 28. Perpindahan posisi ke ruang lab yang berbeda dapat dilakukan dengan cara klik posisi asal, kemudian klik posisi tujuan di ruangan lab lainnya. Hasil perpindahan akan tercatat di kolom *list* yang ada di sebelah kanan. Setelah hasil perpindahan dikonfirmasi dan tombol “Pindah dan Buat Script” ditekan, maka script untuk memindahkan posisi ujian peserta akan dibuat kemudian dapat dijalankan.

– Mode Ujian dengan Otentikasi Berbasis Token

Pada mode ujian dengan otentikasi berbasis ujian dapat dilakukan dari luar ruangan lab. Ujian dengan mode ini dapat dipersiapkan oleh Tim Admin maupun Tim Dosen dan nantinya akan diikuti oleh peserta dengan mengakses halaman exam online melalui laptop atau komputer yang dimiliki peserta.

Tampilan Antarmuka untuk Peserta

Peserta dapat melakukan akses ujian dengan mengunjungi alamat <https://oxam.labfti.net/exam-online>. Ketika mengunjungi halaman ini, peserta ujian diharuskan untuk melakukan *login* terlebih dahulu.

Oxam Exam Login

Username

Password

Have some difficulties, try our [Forgot Password](#) tool here

Login

Oxam v6.0: Radiant Mythology | Built on Senin 30 Februari 2022 pukul 25.61, Commit: ffpubgml

Gambar 29: Halaman *login* untuk peserta ujian.

Setelah mengisi *form login* seperti yang terlihat pada Gambar 29, data *login* peserta akan dicocokkan dengan data peserta yang ada pada *Active Directory* (AD). Jika pasangan *username* dan *password* sesuai, maka peserta akan diberikan token yang digunakan untuk mengakses ujian.

Oxam Exam Online

Ujian Akhir Semester

Hai, i17090 berikut detail sesi ujianmu:

Mata Kuliah: AIF183236 Sertifikasi Administrasi Jaringan Komputer 2

Durasi: 105 Menit

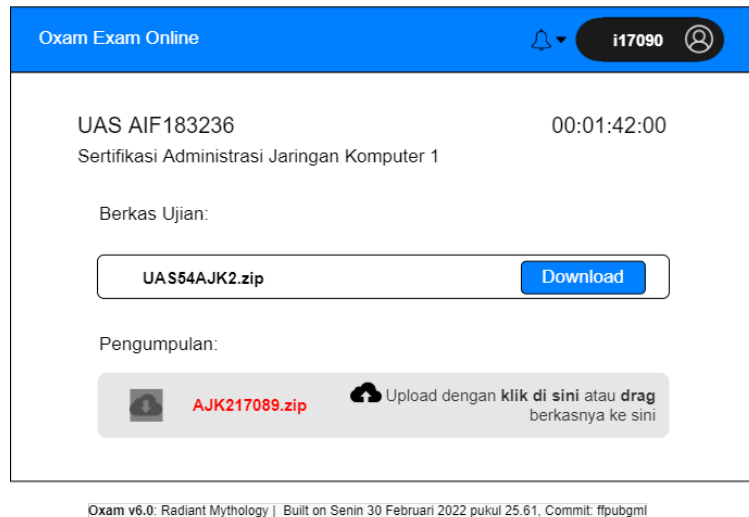
Berkas Ujian: UAS54AJK2.zip Download

Ujian dapat dimulai saat dosen pengawas telah menekan tombol mulai pada komputer dosen. Kontak dosen pengawas jika terdapat masalah atau kesalahan informasi ujian.

Oxam v6.0: Radiant Mythology | Built on Senin 30 Februari 2022 pukul 25.61, Commit: ffpubgml

Gambar 30: Halaman yang akan tampil setelah peserta melakukan *login* pada saat ujian akan berlangsung.

Pada ujian yang berlangsung di ruangan lab, berkas ujian didistribusikan ke komputer menggunakan *script*. Berbeda dengan mode ujian dengan otentikasi berbasis token, berkas ujian akan disertakan dalam bentuk *downloadable file*. Pada Gambar 30 dapat dilihat pada baris “Berkas Ujian” terdapat tombol *Download* yang dapat digunakan untuk melakukan unduh berkas ujian yang disertakan oleh Tim Admin atau Dosen pada saat membuat ujian dengan mode berbasis token.



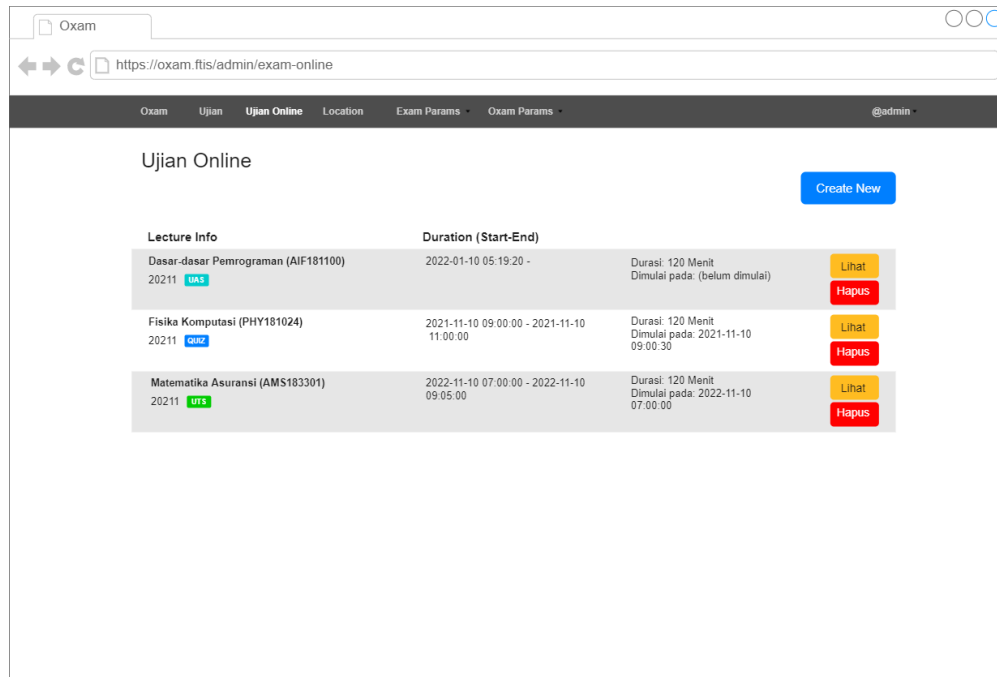
Gambar 31: Halaman yang akan tampil setelah peserta melakukan *login* pada saat ujian sedang berlangsung.

Gambar 31 merupakan tampilan ketika peserta berhasil melakukan *login* dan ujian sudah dimulai oleh Dosen. Berkas ujian tetap disertakan, sehingga dapat diunduh kembali ketika dibutuhkan.

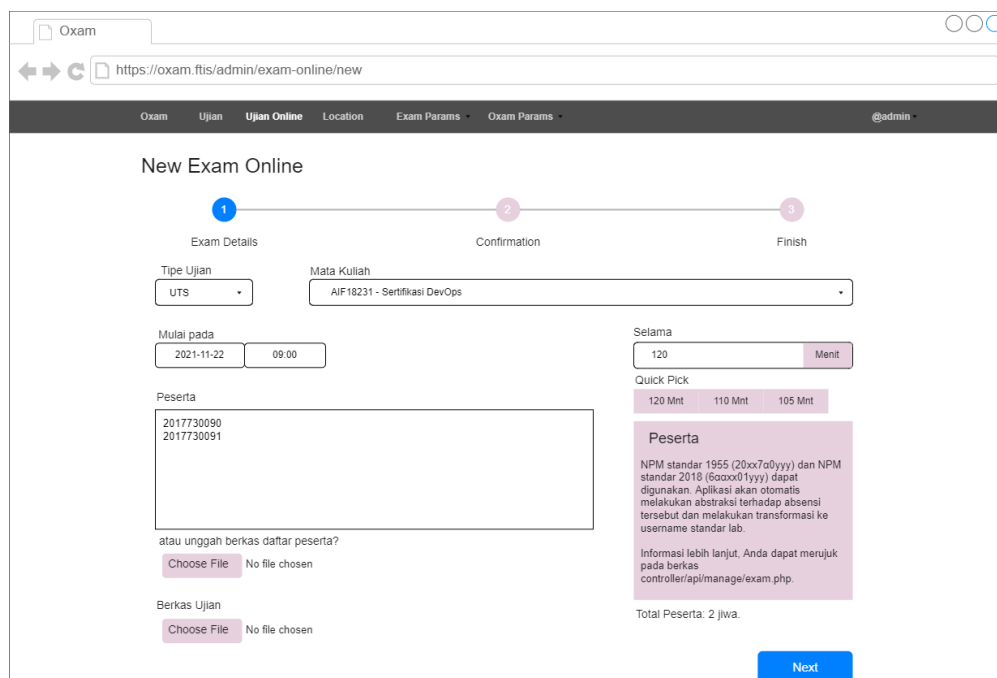
Tampilan Antarmuka untuk Dosen dan Tim Admin

Tim Admin dan juga Dosen dapat membuat entri baru untuk ujian dengan mode berbasis token. Sebelumnya dapat menambahkan entri baru, baik Tim Admin maupun Dosen harus melakukan *login* terlebih dahulu melalui halaman otentikasi yang ada pada alamat <https://oxam.labftis.net/admin>, kemudian mengakses menu utama “Ujian Online” untuk mulai menambahkan entri ujian.

Gambar 32 merupakan tampilan untuk menu utama “Ujian Online”. Tim Admin maupun Dosen dapat menambahkan entri baru ujian dengan menekan tombol “*Create New*”. Terdapat 3 buah label berbeda untuk menandai jenis entri ujian, yaitu label dengan warna biru muda untuk UTS, label hijau untuk UAS, dan label dengan warna biru tua untuk Quiz.



Gambar 32: Halaman utama untuk menu utama “Ujian Online” .



Gambar 33: Langkah pertama dalam penambahan entri ujian dengan mode otentikasi berbasis token.

Penambahan entri ujian baru pada mode ujian dengan otentikasi berbasis token terdiri dari tiga (3) tahap, yaitu tahap pengisian detail ujian (*exam detail*), tahap konfirmasi (*confirmation*), dan tahap penyelesaian (*finish*). Gambar 33 merupakan tahap pertama, yaitu tahap pengisian detail dari ujian. Pada tahap ini, Tim Admin maupun Dosen yang menambahkan entri ujian dapat memilih salah satu dari tiga jenis ujian yang ada, yaitu Quiz, UTS, dan UAS. Pada bagian paling bawah (bagian “Berkas Ujian”), Tim Admin maupun Dosen dapat menambahkan berkas ujian yang akan digunakan pada saat ujian berlangsung.

New Exam Online



Konfirmasi

QUIZ Sertifikasi DevOps

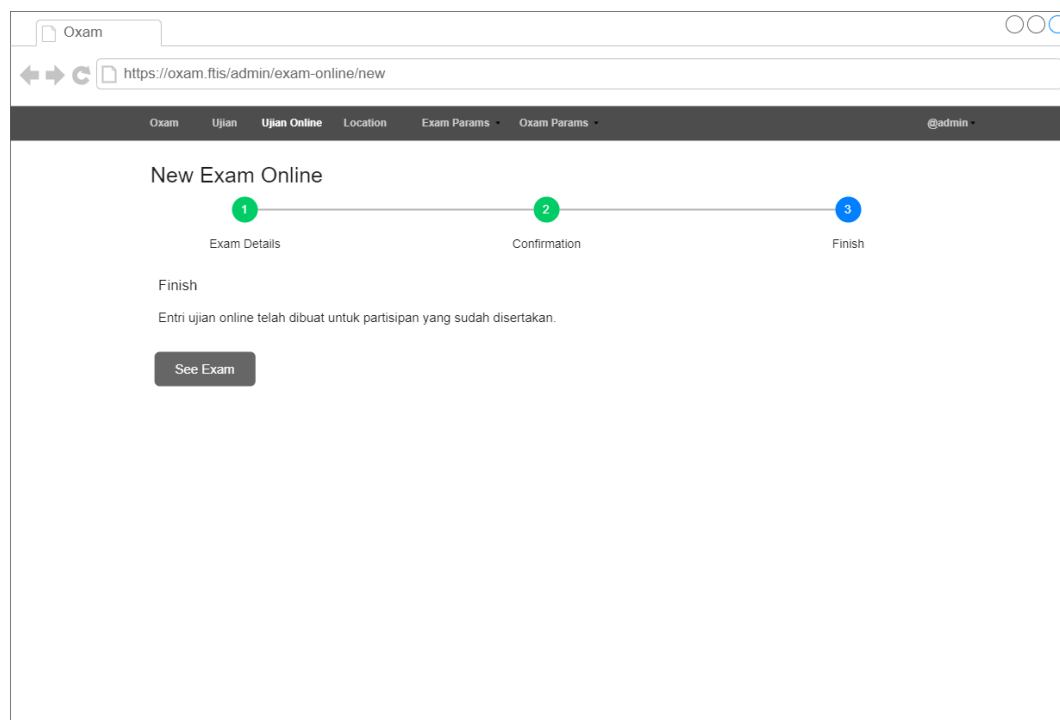
Jumlah Peserta: 2 Peserta

Dilaksanakan secara online. Akan dimulai pada **Senin, 22 November 2021** pukul **09.00** selama **1.75 jam**.

Untuk konfirmasi pembuatan ujian, tekan tombol di bawah ini.

Create Exam

Gambar 34: Langkah terakhir dalam penambahan entri ujian dengan mode otentikasi berbasis token.



Gambar 35: Langkah kedua dalam penambahan entri ujian dengan mode otentikasi berbasis token.

Gambar 34 merupakan tahap kedua atau tahap konfirmasi. Setelah konfirmasi selesai dilakukan dan tombol “*Create Exam*” ditekan, maka entri ujian akan ditambahkan ke basis data dan akan menampilkan halaman penyelesaian yang dapat dilihat pada Gambar 34.

6. Melaporkan hasil penelitian dalam bentuk dokumen skripsi.

Status : Ada sejak rencana kerja skripsi.

Hasil : -

6 Pencapaian Rencana Kerja

Langkah-langkah kerja yang berhasil diselesaikan dalam Skripsi 1 ini adalah sebagai berikut:

1. Mempelajari cara kerja OXAM dan skripsi dari Gunawan Christianto (2016730011).
2. Mempelajari React.js sebagai *library front-end*.
3. Mempelajari FatFree sebagai *framework* yang digunakan untuk *back-end*.

Bandung, 03/01/2022

Christian S. G. Patrick Nasira

Menyetujui,

Nama: Raymond Chandra Putra, S.T., M.T.
Pembimbing Tunggal