BAB IV

PEMBAHASAN

4.1 Definisi Aktor

Nazli R (2018) aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan – pekerjaan tertentu [10]. Singkatnya aktor adalah pengguna yang terlibat dalam sistem ini, yaitu:

1. Admin

Dalam sistem ini admin memiliki hak akses antara lain :

a. Mengolah data user

Hak akses yang dimiliki oleh admin pada data *user* adalah admin berhak melakukan penambahan *user* baru, melakukan perubahan data *user* yang telah disimpan, melihat dan mencari data *user* serta menghapus data *user*.

b. Mengolah data lowongan program magang kerja

Hak akses yang dimiliki oleh admin pada data lowongan magang kerja adalah admin berhak melakukan penambahan, pengeditan, penghapusan serta mencari dan menampilkan data lowongan program magang kerja.

c. Mengolah data informasi

Hak akses yang dimiliki oleh admin pada data informasi meliputi post berita, blog, menu *about us*, menu *term of use* dan data peserta program magang kerja adalah admin dapat melakukan

penambahan, pengeditan penghapusan serta mencari dan menampilkan data informasi.

2. Perusahaan

Dalam sistem ini Perusahaan memiliki hak akses antara lain :

a. Mengolah Profil

Hak akses yang dimiliki oleh perusahaan adalah perusahaan dapat mengganti informasi profil.

b. Membuat Lowongan Magang Kerja

Hak akses yang dimiliki oleh perusahaan adalah perusahaan dapat membuat lowongan program magang kerja yang nantinya akan ditinjau dan disetujui oleh admin.

c. Menerima Peserta Magang Kerja

Hak akses berikutnya yang dimiliki oleh perusahaan adalah dapat menolak maupun menerima pelamar program magang kerja pada lowongan magang kerja yang telah dibuat, sesuai kebutuhan perusahaan dengan menggunakan fitur filter sebagai cara mencari Mahasiswa peserta magang kerja yang tepat.

d. Mendapatkan akses data Transkrip dan SKPI

Perusahaan akan mendapatkan akses untuk membaca transkrip nilai serta Surat Keterangan Pendamping Ijazah (SKPI) dari Mahasiswa peserta program magang kerja yang terkait.

e. Membuat Laporan Hasil Program Magang Kerja

Kemudian hak akses yang dimiliki oleh perusahaan adalah mengolah dan membuat laporan hasil program magang kerja yang dapat langsung dicetak kepada Mahasiswa peserta program magang kerja.

3. Mahasiswa

Dalam sistem ini Perusahaan memiliki hak akses antara lain :

a. Mengolah Profil

Hak akses yang dimiliki oleh Mahasiswa adalah Mahasiswa dapat mengganti informasi profil, dari kontak serta foto terakhir.

b. Mencari Lowongan Magang Kerja

Mahasiswa dapat mencari lowongan magang kerja yang tersedia dengan fitur filter sesuai keinginan Mahasiswa.

c. Melamar

Mahasiswa mendapat akses untuk mendaftar Program Magang kerja yang telah dibuat oleh perusahaan, dengan satu tombol tanpa harus melampirkan Transkrip dan Surat Keterangan Pendamping Ijazah (SKPI).

d. Mendapat Laporan Hasil Program Magang Kerja

Setelah Program magang kerja telah selesai diikuti, dan perusahaan sudah membuat laporan hasil Magang kerja. Mahasiswa mendapat akses untuk melihat Hasil program magang kerja yang telah dibuat oleh perusahaan berupa nilai.

4. Fakultas

Dalam sistem ini Perusahaan memiliki hak akses antara lain :

a. Mengolah Profil

Hak akses yang dimiliki oleh fakultas adalah fakultas dapat mengganti informasi Profil, dari kontak serta foto terakhir.

b. Mengolah data user

Hak akses yang dimiliki oleh fakultas pada data *user* adalah fakultas dapat melakukan penambahan *user* baru, melakukan perubahan data *user* yang telah disimpan, melihat dan mencari data *user*.

c. Mendapat Laporan Hasil Program Magang Kerja

Setelah Program magang kerja telah selesai diikuti, dan perusahaan sudah membuat laporan hasil Magang kerja. Fakultas dapat mengakses untuk melihat Hasil program magang kerja yang telah dibuat oleh perusahaan berupa nilai.

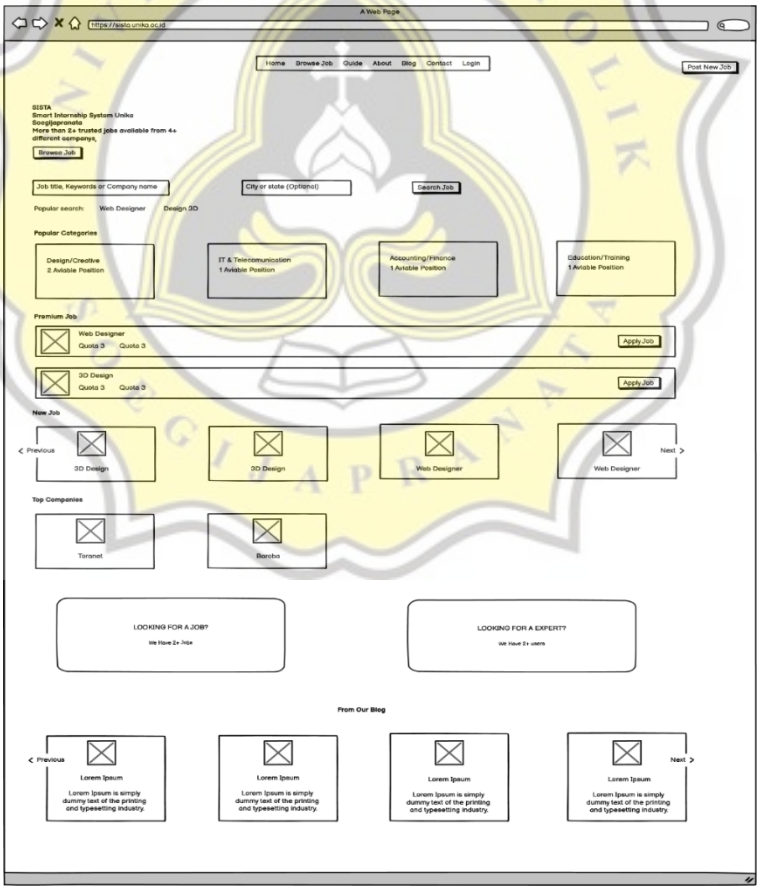
4.2 Perancangan Aplikasi

Aplikasi Smart Internship System Unika Soegijapranata (SISTA) Berbasis Pwa merupakan suatu aplikasi yang dirancang untuk dapat menampung semua kegiatan program kerja magang dari membuka dan mencari lowongan program magang kerja, melakukan persetujuan dan kerjasama program magang kerja serta memberikan nilai atau hasil program magang kerja yang terintegrasi dengan informasi transkrip nilai dan Surat Keterangan Pendamping Ijazah (SKPI) Mahasiswa Universitas Katolik Soegijapranata.

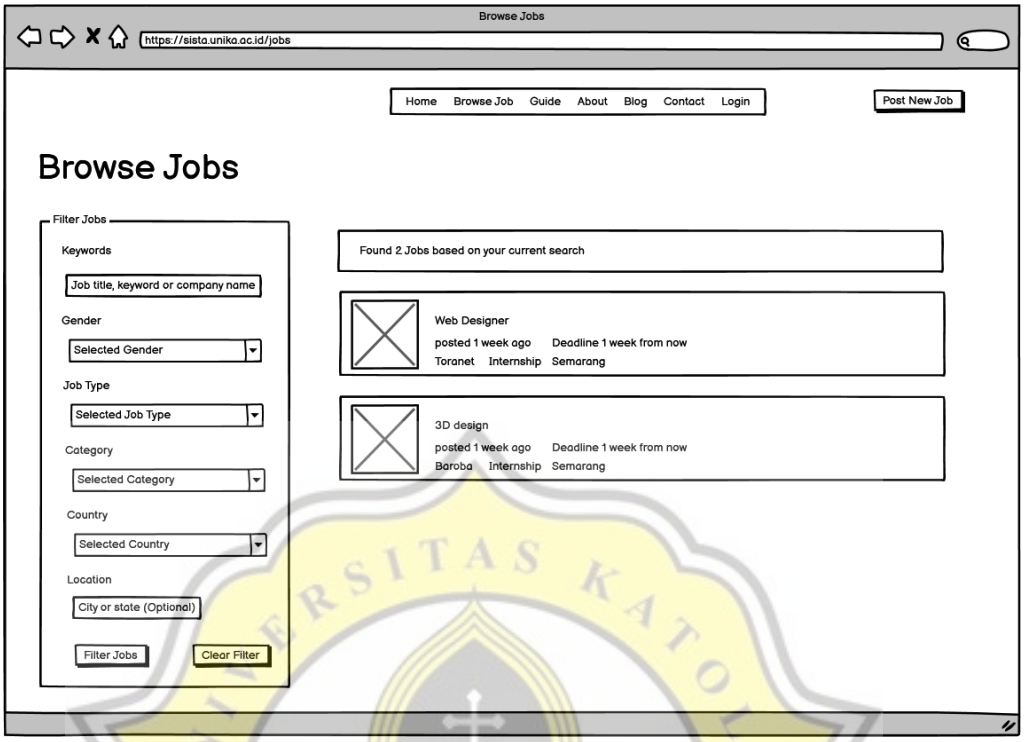
Aplikasi ini akan dirancang menggunakan aplikasi web serta penggunaan *progressive web app* untuk dapat digunakan di segala jenis platform.

4.2.1 Perancangan Desain Aplikasi

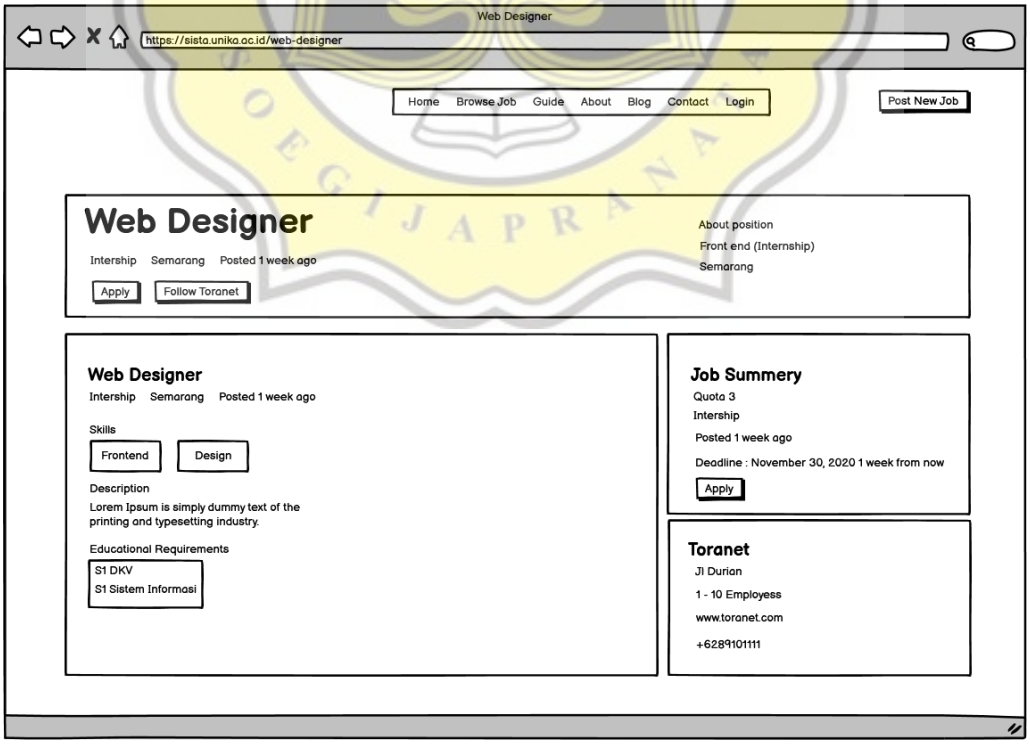
Sebagai mockup awal, aplikasi di desain menggunakan balsamiq. Pembuatan Mockup ini diperlukan karena mockup akan menjadi pedoman teknis dalam merancang halaman web. Berikut merupakan tampilan mockup Aplikasi Smart Internship System Unika Soegijapranata (SISTA) Berbasis PWA:



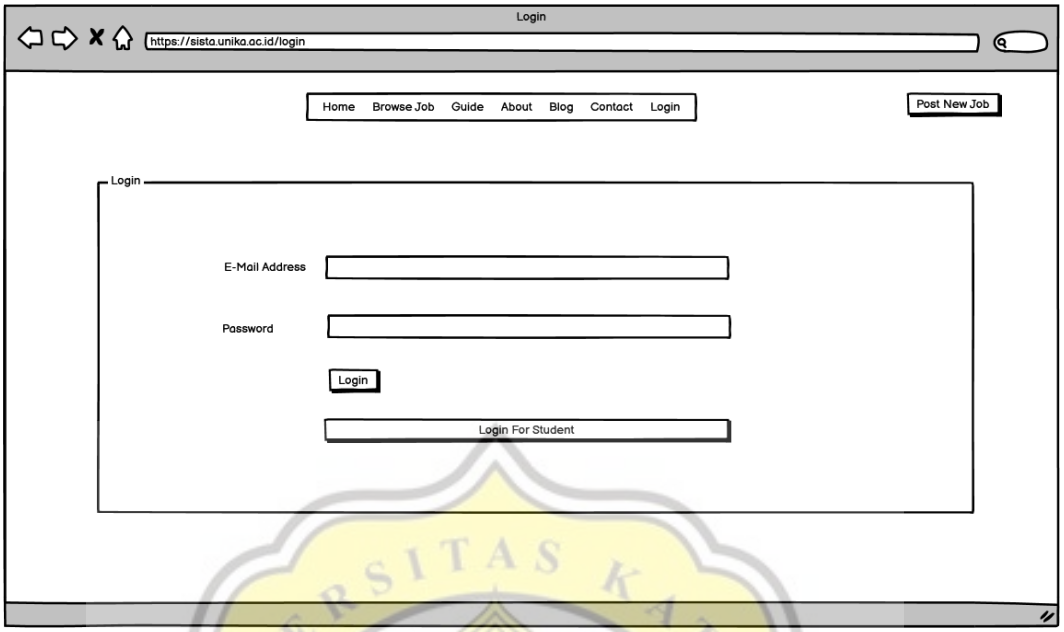
Gambar 4.2.1. Rancangan tampilan awal



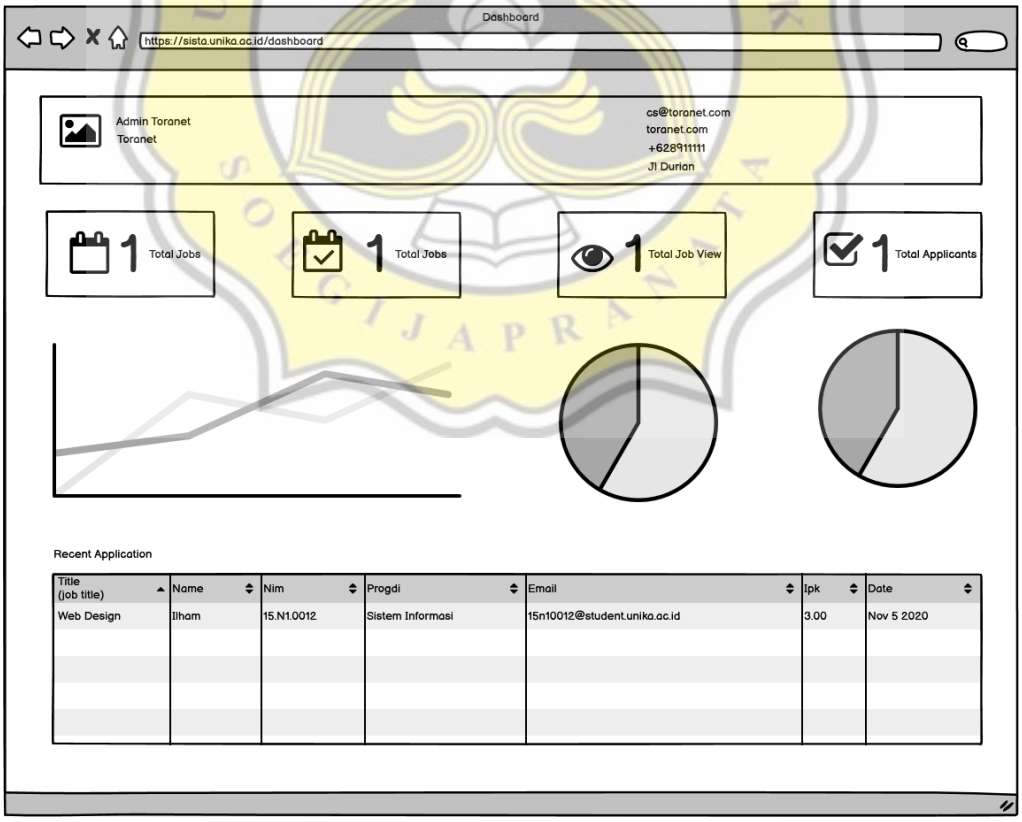
Gambar 4.2.2. Rancangan tampilan browse jobs



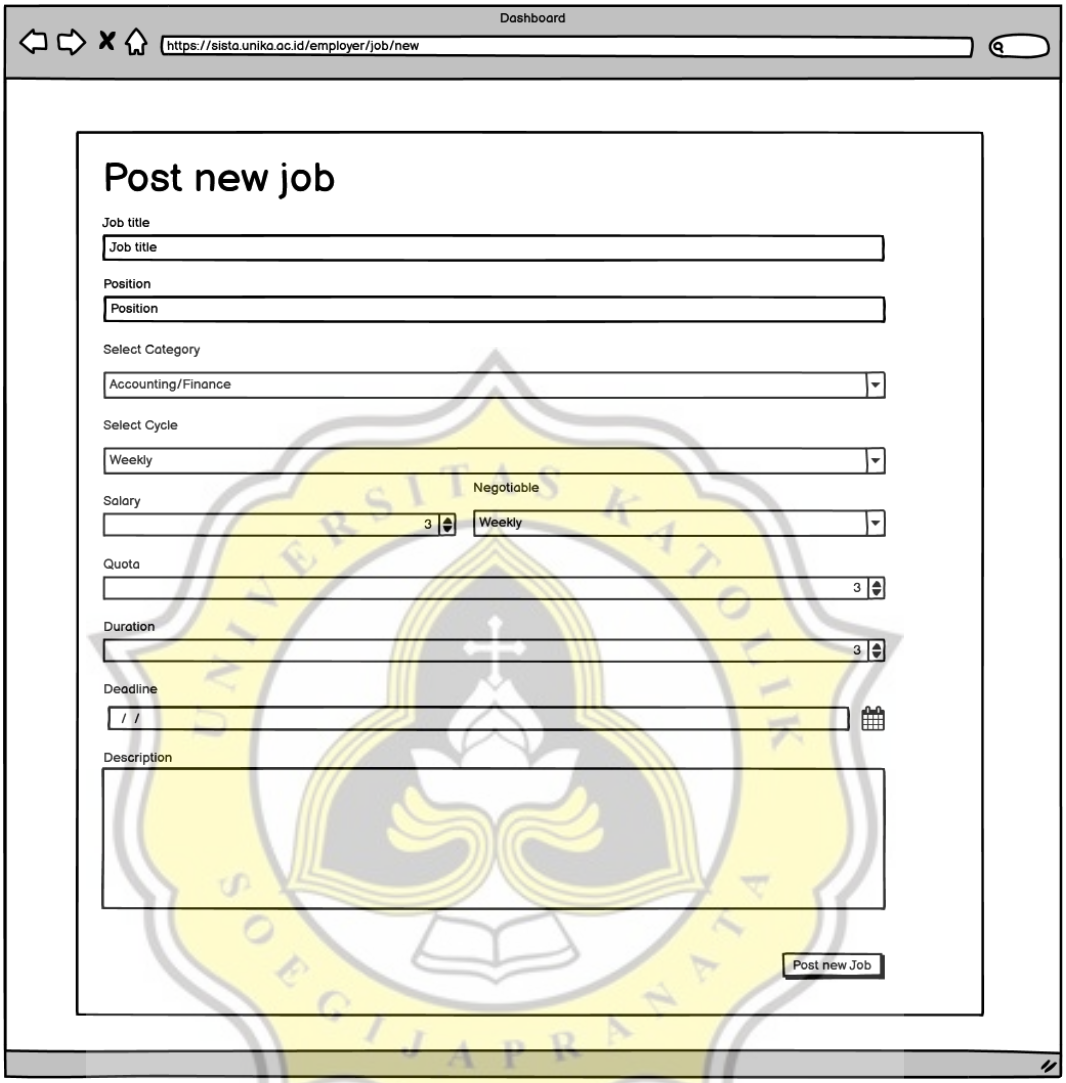
Gambar 4.2.3. Rancangan tampilan awal



Gambar 4.2.4. Rancangan tampilan login



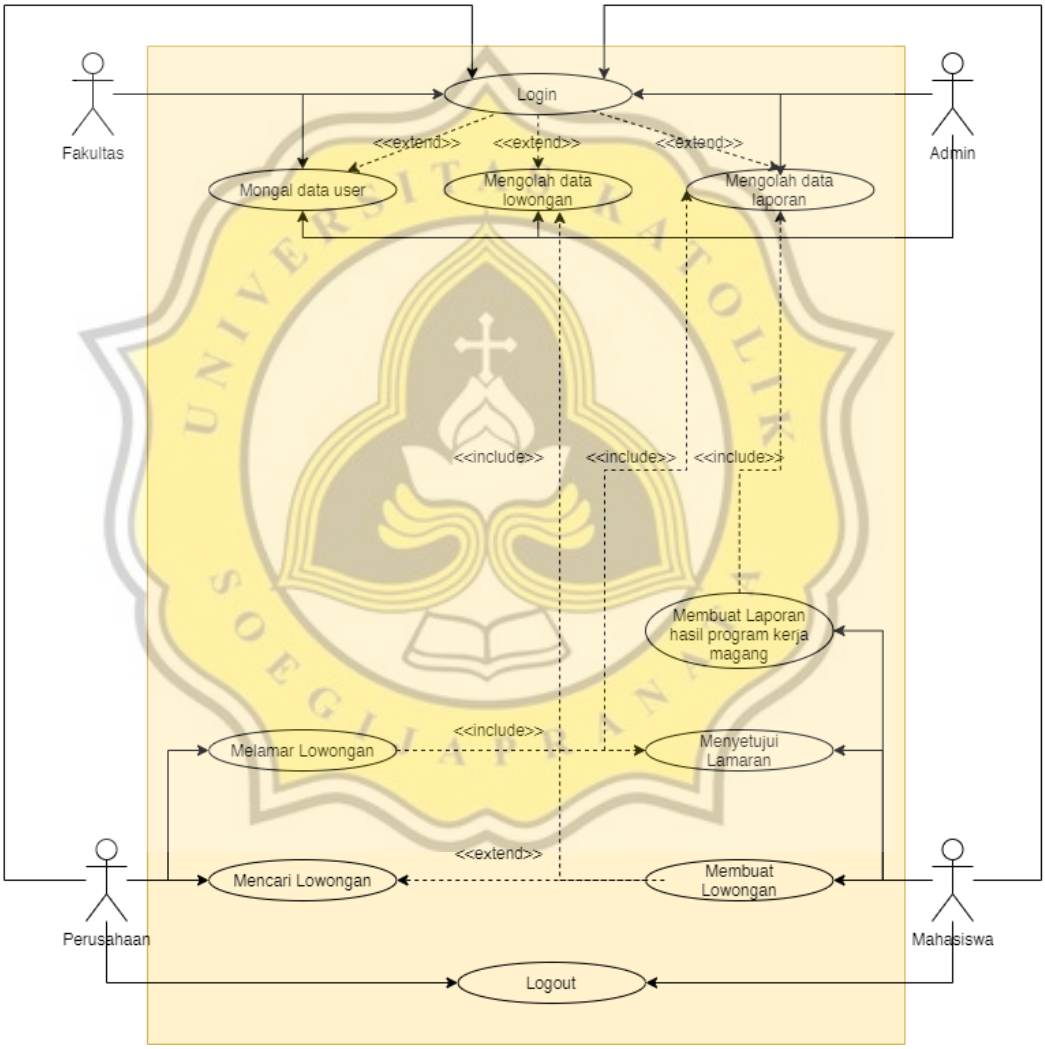
Gambar 4.2.5. Rancangan tampilan dashboard



Gambar 4.2.6. Rancangan tampilan membuat lowongan baru

4.2.2 Perancangan Use Case

Perancangan *use case* diperlukan untuk memahami fungsi apa saja yang ada pada sebuah sistem serta siapa saja yang dapat menggunakan fungsi – fungsi tersebut. Berikut adalah *use case* Smart Internship System Unika Soegijapranata (SISTA) Berbasis PWA (Progressive Web App):



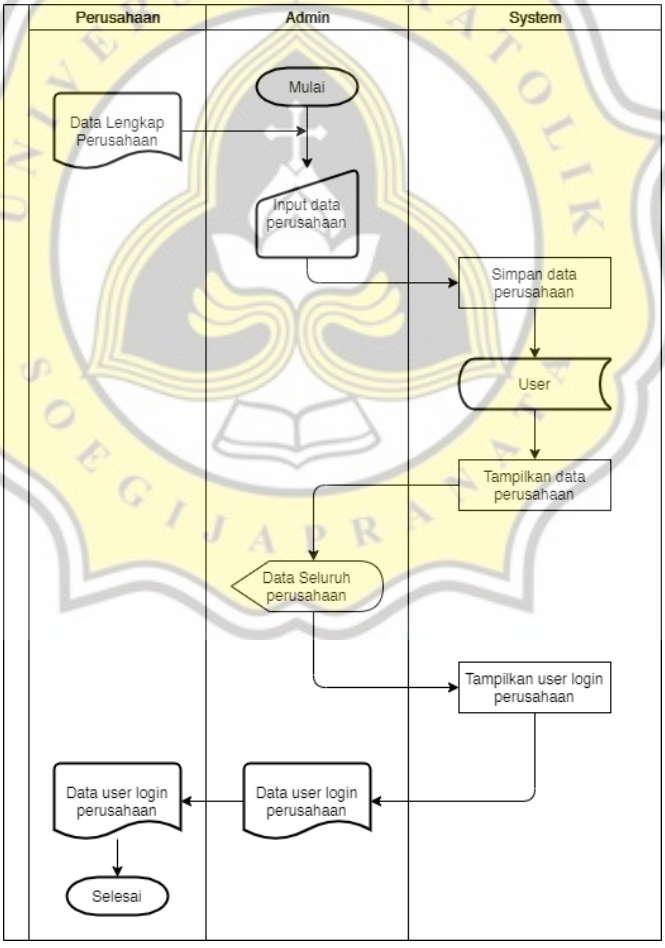
Gambar 4.2.7. Diagram use-case

4.2.3 System Flow

System flow atau sering disebut alur sistem adalah suatu jenis diagram yang mewakili atau menggambarkan suatu alur kerja atau proses suatu program menjadi lebih sederhana agar program tersebut lebih dapat dimengerti. Berikutnya adalah system flow Smart Internship System Unika Soegijapranata (SISTA) Berbasis PWA (Progressive Web App):

A. System Flow Penambahan data Perusahaan

System flow penambahan data perusahaan dapat dilihat pada Gambar 4.8.

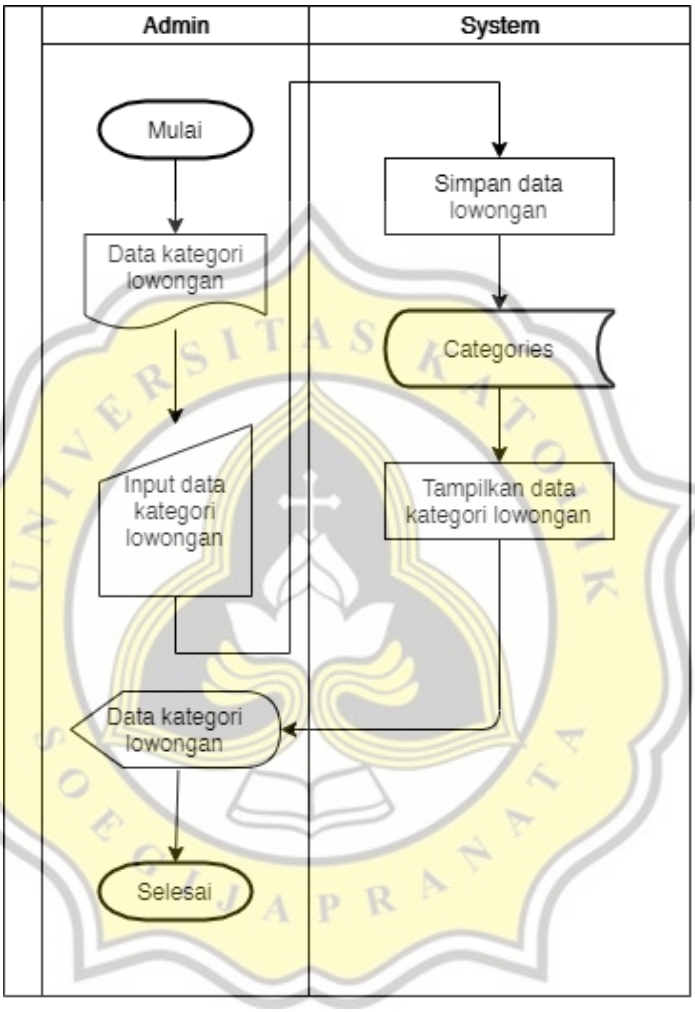


Gambar 4.2.8. System flow penambahan data perusahaan

System flow penambahan data perusahaan ini diawali dari perusahaan memberikan data lengkap mengenai perusahaan tersebut, kemudian pihak admin akan melakukan pengisian data perusahaan kemudian data disimpan pada tabel *user*, selanjutnya sistem akan memproses lalu menampilkan user baru. User baru ini nantinya akan digunakan untuk login pada saat perusahaan akan masuk ke aplikasi untuk mengolah serta membuat lowongan magang.

B. System Flow Mengolah Data Kategori

System flow mengolah data kategori dapat dilihat pada Gambar 4.9.

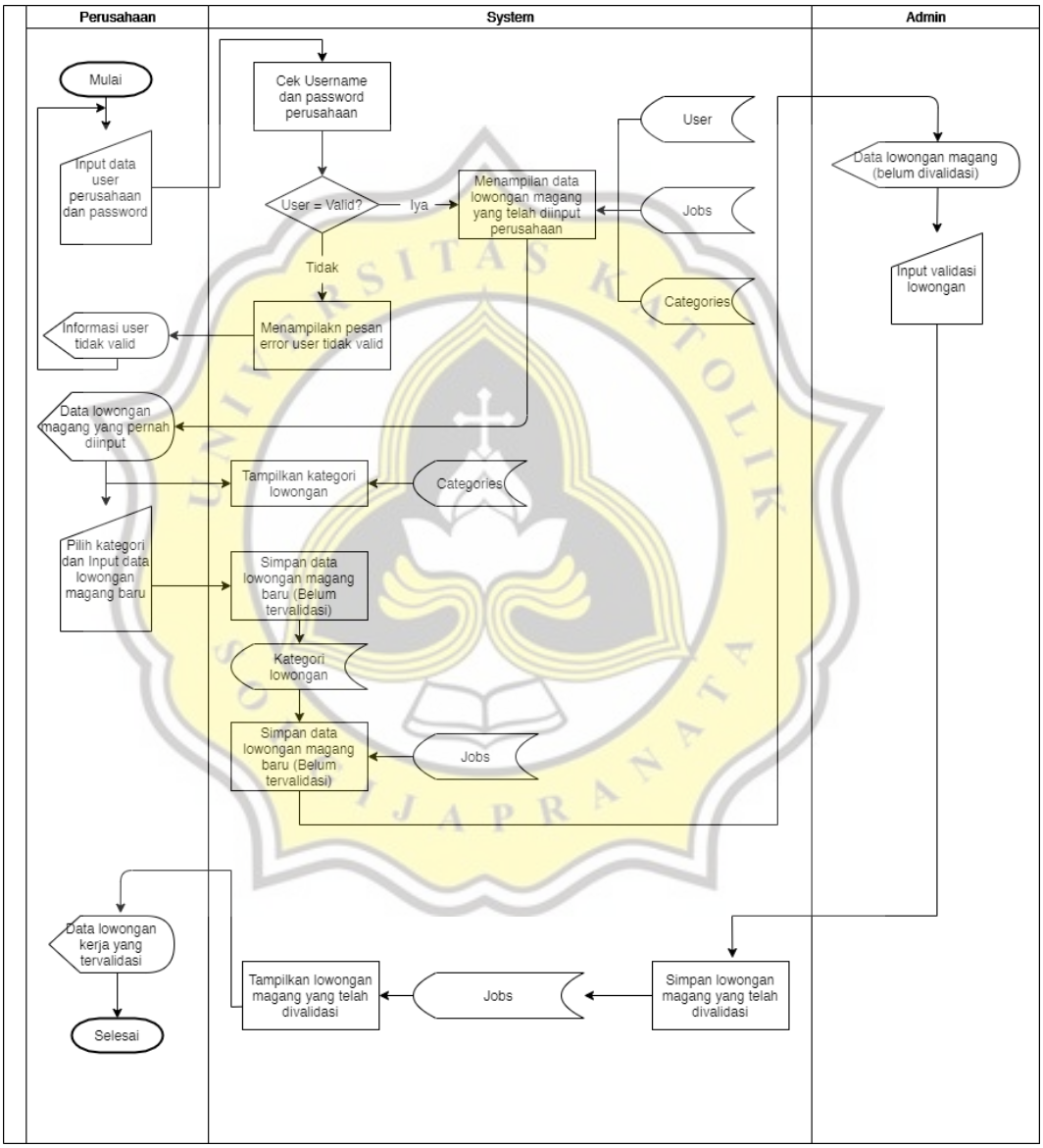


Gambar 4.2.9. System flow mengolah data kategori

System flow mengolah data kategori ini dimulai dari Admin mendapatkan daftar kategori lowongan magang kerja kemudian akan diolah dan dilakukan penginputan data kategori lowongan magang kerja berdasarkan dokumen tersebut kemudian disimpan pada tabel *categories*.

C. System Flow Proses Input Lowongan Magang Kerja

System flow proses input lowongan magang kerja dapat dilihat pada Gambar 4.10.



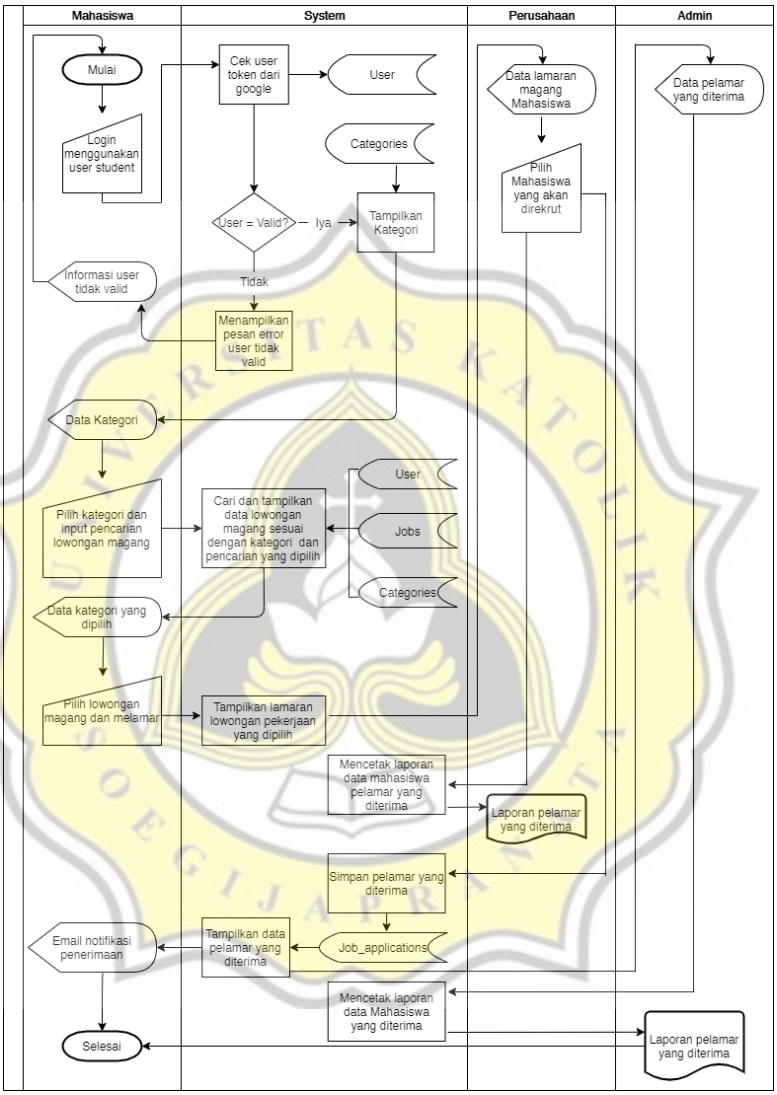
Gambar 4.2.10. System flow proses input lowongan magang kerja

System flow proses input lowongan magang kerja ini dimulai dari perusahaan melakukan login terlebih dahulu ke halaman dashboard dengan

memasukan username berupa email dan password, kemudian sistem akan melakukan validasi user, jika validasi gagal sistem akan menampilan pemberitahuan bahwa user tersebut tidak valid, jika validasi berhasil maka perusahaan akan dialihkan ke halaman dashboard. Kemudian perusahaan menginput data lowongan magang kerja yang akan dibuat, setelah data lowongan magang kerja sudah dibuat, akan disimpan di tabel *jobs*, admin akan melakukan persetujuan maka sistem akan menampilkan data lowongan magang kerja yang sudah tervalidasi pada Smart Internship System Unika Soegijapranata (SISTA) Berbasis PWA (Progressive Web App) dan dapat dilihat oleh para Mahasiswa.

D. System Flow Proses Mencari Dan Melamar Lowongan Magang Kerja

System flow proses mencari dan melamar lowongan magang kerja dapat dilihat pada Gambar 4.11.



Gambar 4.2.11. System flow proses mencari dan melamar lowongan magang kerja

System flow mencari dan melamar lowongan magang kerja ini dimulai dari pencarian lowongan magang kerja, Mahasiswa diharuskan login menggunakan alamat email student Unika Soegijapranata terlebih dahulu, kemudian google akan

mengirim token validasi dan sistem akan melakukan pengecekan apakah user tersebut benar merupakan Mahasiswa Unika Soegijapranata yang masih aktif, jika validasi tidak berhasil akan diberi pemberitahuan berupa *text* bahwa user ini tidak valid, jika berhasil melewati validasi maka user akan dialihkan ke halaman dashboard, kemudian Mahasiswa dapat mencari lowongan magang kerja berdasarkan kategori maupun kolom pencarian sesuai dengan keinginan Mahasiswa, sistem akan menampilkan lowongan magang kerja yang tersedia, selanjutnya Mahasiswa dapat melakukan proses melamar magang kerja, data lamaran magang kerja akan disimpan pada tabel *job applications*, setelah Mahasiswa melakukan proses pelamaran magang kerja, perusahaan akan melakukan pemilihan kepada Mahasiswa yang diterima pada program magang kerja tersebut, Mahasiswa akan mendapatkan email pemberitahuan jika diterima, pihak admin juga mendapatkan laporan data Mahasiswa yang hanya melamar maupun yang telah diterima oleh perusahaan terkait.

E. Perancangan ERD

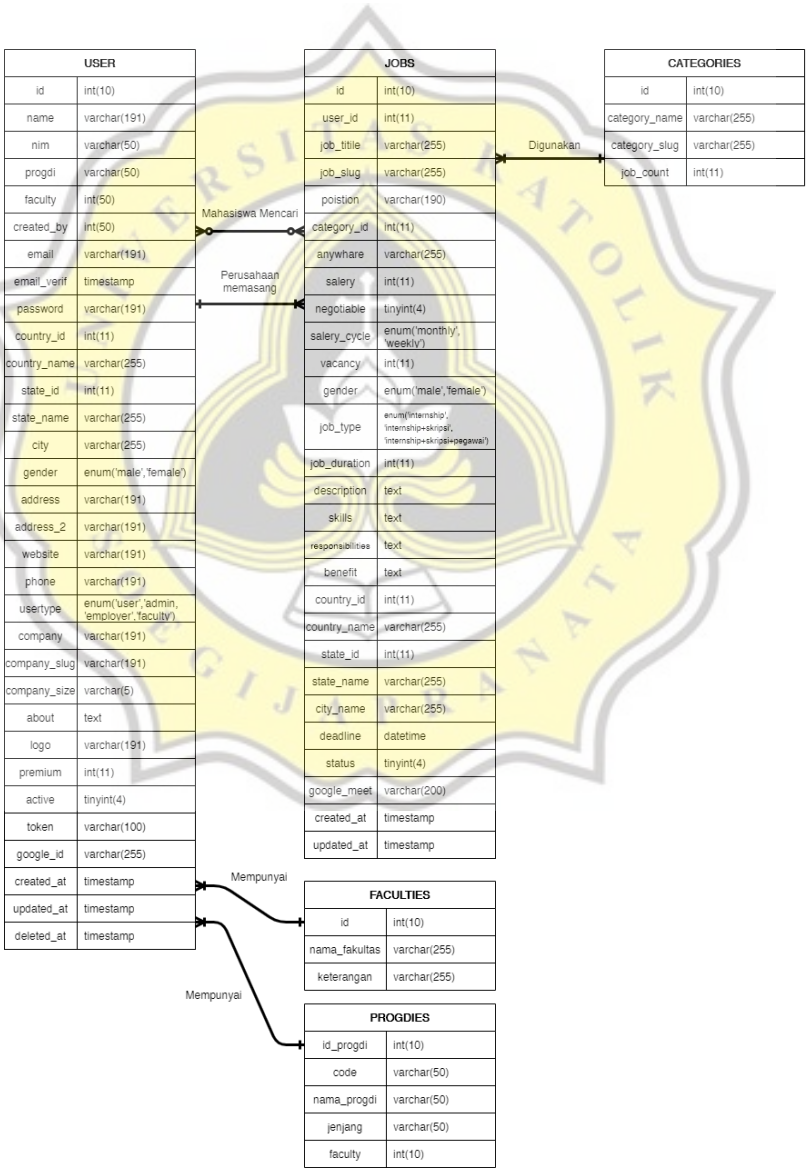
ERD atau biasa disebut diagram relasi entitas adalah suatu desain sistem untuk merepresentasikan model data yang ada pada sistem dan didalamnya terdapat *entity* dan *relationship*. ERD ini menggambarkan tabel – tabel dan relasinya yang ada pada Smart Internship System Unika Soegijapranata (SISTA) Berbasis PWA (Progressive Web App). Dalam pembentukan ERD terdapat 3 komponen yang akan digunakan, yaitu:

1. Entitas adalah tabel dari database.
2. Relasi adalah relasi yang menghubungkan antara entitas satu dan yang lainnya.
3. Atribut adalah file atau kolom dalam tabel.

ERD dibagi menjadi dua, yaitu *Conceptual Data Model* (CDM) dan *Physical Data Model* (PDM).

1. *Conceptual Data Model* (CDM)

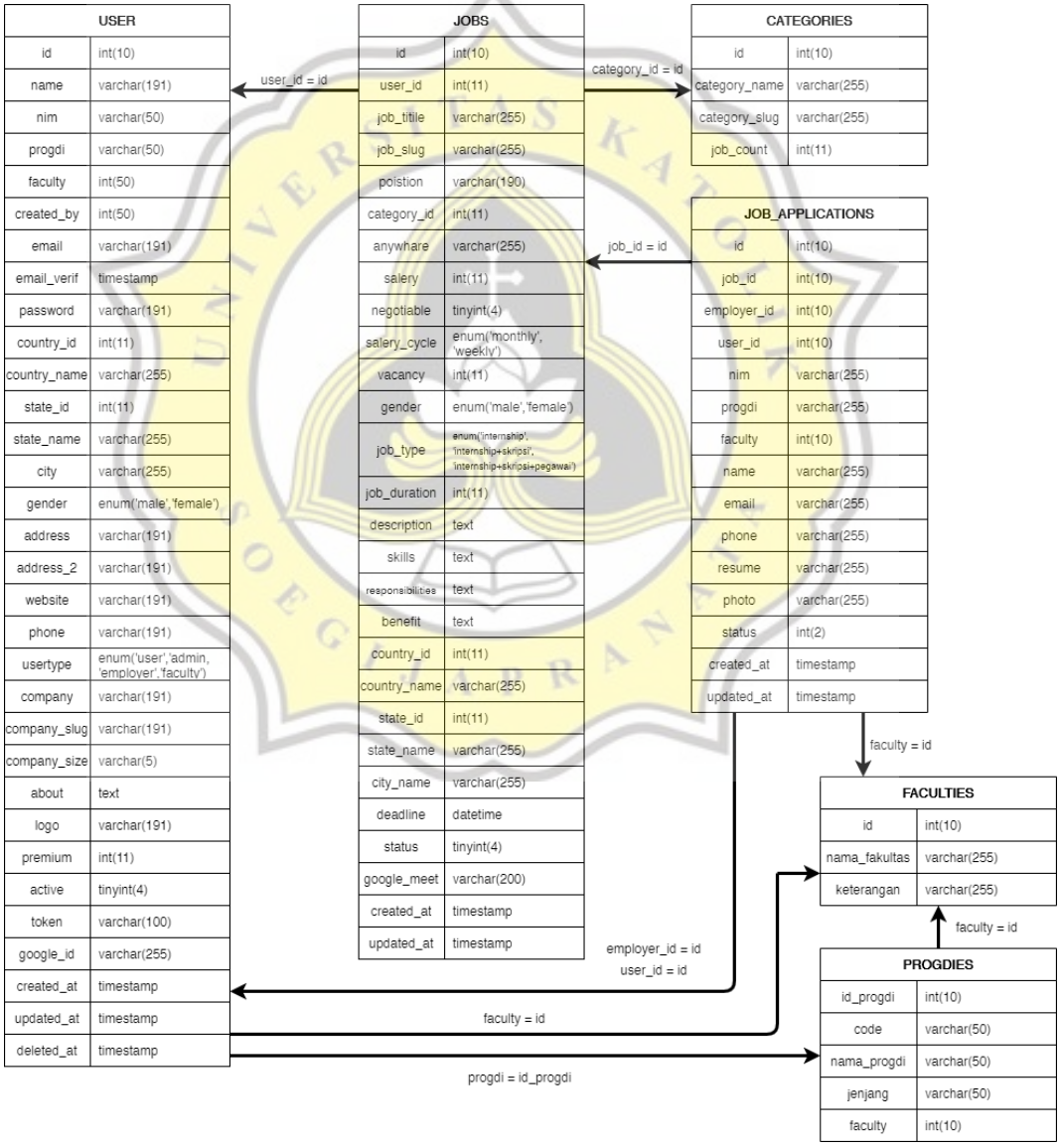
   CDM menggambarkan secara keseluruhan konsep terstruktur basis data yang dirancang untuk suatu program atau aplikasi. CDM dapat dilihat pada Gambar 4.12.



Gambar 4.2.12. CDM Smart Internship System Unika Soegijapranata (SISTA) Berbasis PWA (Progressive Web APP)

2. Physical Data Model (PDM)

PDM menggambarkan secara detail konsep rancangan basis data yang dirancang untuk suatu program aplikasi. PDM Smart Internship System Unika Soegijapranata (SISTA) Berbasis PWA (Progressive Web App) dapat dilihat pada Gambar 4.13.



Gambar 4.2.13. PDM Smart Internship System Unika Soegijapranata (SISTA) Berbasis PWA (Progressive Web App)

4.3 Pembuatan Aplikasi

Smart Internship System Unika Soegijapranata (SISTA) Berbasis PWA (Progressive Web App) merupakan aplikasi berbasis web yang dapat dibuka di berbagai *device.* Pembuatan aplikasi ini menggunakan *framework* bernama Laravel, Laravel sendiri dibangun menggunakan konsep MVC (*Model, View, Controller*), *Model* mewakili struktur data. Biasanya model berisi fungsi yang membantu pengembang mengolah basis data seperti membuat, menghapus, memperbaharui dan menampilkan data. View adalah bagian yang mengatur tampilan ke pengguna, sedangkan Controller merupakan bagian yang menjembatani Model dengan View, Bahasa yang digunakan mayoritas menggunakan bahasa pemrograman PHP dan HTML.

4.3.1 Database

Sebelum memulai untuk membuat fungsi aplikasi, terlebih dahulu harus membuat database, pembuatan database ini menggunakan MYSQL. Database ini terdiri dari berbagai tabel yang akan berguna dalam pembuatan Smart Internship System Unika Soegijapranata (SISTA) Berbasis PWA (Progressive Web App).

Pertama adalah tabel *categories* tabel ini berisi kategori lowongan program magang kerja, setiap perusahaan melakukan input data lowongan magang kerja dan memilih kategori tertentu, jumlah kategori dan lowongan magang kerja akan disimpan pada tabel tersebut, tabel *categories* berisi id, *category_name, category_slug,* dan *job_count.* Tampilan tabel *categories* dapat dilihat pada tabel 4.1.

Tabel 4.3.1. Categories

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(10)		UNSIGNED	No	*None*		AUTO_INCREMENT
2	categor name	Varchar (255)	utf8mb code_ci		Yes	*NULL*		
3	categor slug	Varchar (255)	utf8mb code_ci		Yes	*NULL*		
4	job_cou nt	int(11)			Yes	0		

Kedua adalah tabel *faculties* tabel ini berisi mengenai data nama fakultas. Tampilan tabel *faculties* dapat dilihat pada tabel 4.2.

Tabel 4.3.2. Faculties

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(50)			No	*None*		AUTO_INCREMENT
2	nama_fakultas	varchar(50)	latin1_swedish_ci		Yes	*NULL*		
3	Keterangan	varchar(50)	latin1_swedish_ci		No	-		

Ketiga adalah tabel prodi tabel ini berisi mengenai data nama program studi, sedangkan data *faculty* berkorelasi dengan id pada tabel *faculties*. Tampilan tabel prodi dapat dilihat pada tabel 4.3.

Tabel 4.3.3. Prodi

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id_progdi	int(11)			No	*None*		AUTO_INCREMENT
2	code	varchar(50)	latin1_swedish_ci		Yes	*NULL*		
3	nama_progdi	varchar(100)	latin1_swedish_ci		Yes	*NULL*		
4	jenjang	varchar(50)	latin1_swedish_ci		Yes	*NULL*		
5	faculty	int(50)			Yes	*NULL*		
6	keterangan	text	latin1_swedish_ci		Yes			

Berikutnya tabel yang paling penting adalah tabel yang berisi data user, tabel ini berisi semua informasi user dari admin, perusahaan, mahasiswa dan fakultas, tabel ini terdiri dari id, name, nim, prodi, faculty, created_by, email, email_verified_at, password,country_id, country_name, state_id, state_name, city, gender, address, address_2, website, phone, photo, user_type, company, company_slug, company_size, about_company, logo, premium_jobs_balance, active_status, remember_token, google_id, created_at, updated_at, deleted_at. Data prodi berkorelasi dengan id_progi pada tabel prodi dan *faculty* berkorelasi dengan id pada tabel *faculties* Tampilan tabel user dapat dilihat pada tabel 4.4.

Tabel 4.3.4. User

#	Name	Type	Collation	Attributes	Null	Default	Comments	**Extra**
1	id	int(10)		UNSIGND	No	*None*		AUTO_IMENT
2	name	varchar(11)	utf8mb4_de_ci		Yes	*NULL*		
3	nim	varchar(50)	utf8mb4_de_ci		Yes	*NULL*		
4	progdi	varchar(50)	utf8mb4_de_ci		Yes	*NULL*		
5	faculty	int(50)			Yes	*NULL*		
6	created_by	int(50)			Yes	0		
7	email	varchar(11)	utf8mb4_de_ci		No	*None*		

#	Name	Type	Collation	Attributes	Null	Default	Comments	**Extra**
8	email_verified_at	timestamp			Yes	*NULL*		
9	password	varchar(11)	utf8mb4_de_ci		No	*None*		
10	country_id	int(11)			Yes	*NULL*		
11	country_name	varchar(25)	utf8mb4_de_ci		Yes	*NULL*		
12	state_id	int(11)			Yes	*NULL*		
13	state_name	varchar(25)	utf8mb4_de_ci		Yes	*NULL*		
14	city	varchar(11)	utf8mb4_de_ci		Yes	*NULL*		
15	gender	enum('male'	utf8mb4_de_ci		Yes	*NULL*		
16	address	varchar(11)	utf8mb4_de_ci		Yes	*NULL*		
17	address_2	varchar(11)	utf8mb4_de_ci		Yes	*NULL*		
18	website	varchar(11)	utf8mb4_de_ci		Yes	*NULL*		
19	phone	varchar(11)	utf8mb4_de_ci		Yes	*NULL*		
20	photo	varchar(11)	utf8mb4_de_ci		Yes	*NULL*		
21	user_type	enum('usployer', 'admin', ult...	utf8mb4_de_ci		No	*None*		

#	Name	Type	Collation	Attributes	Null	Default	Comments	**Extra**
22	company	varchar(11)	utf8mb4_de_ci		Yes	*NULL*		
23	company_slug	varchar(11)	utf8mb4_de_ci		Yes	*NULL*		
24	company_size	varchar(50)	utf8mb4_de_ci		Yes	*NULL*		
25	about_company	text	utf8mb4_de_ci		Yes			
26	logo	varchar(11)	utf8mb4_de_ci		Yes	company.png		
27	premium_balance	int(11)			Yes	0		
28	active_status	tinyint(4)			No	0		
29	remember_token	varchar(10)	utf8mb4_de_ci		Yes	*NULL*		
30	google_id	varchar(25)	utf8mb4_de_ci		Yes	*NULL*		
31	created_at	timestamp			Yes	*NULL*		
32	updated_at	timestamp			Yes	*NULL*		
33	deleted_at	timestamp			Yes	*NULL*		

Tabel berikutnya yang dibuat adalah tabel *jobs*, tabel ini berisi semua data lowongan magang kerja yang dibuat oleh perusahaan. Isi tabel *jobs* adalah user_id

yang berkorelasi dengan kolom id pada tabel user. Tampilan tabel *jobs* dapat dilihat pada tabel 4.5. LAMPIRAN

Tabel 4.3.5. Jobs

#	Name	Type	Collatio	Attribut	Null	Default	Comme	Extra
1	id	int(10)		UNSIG	No	*None*		AUTO_ MENT
2	user_id	int(11)			Yes	*NULL*		
3	job_title	varchar(	utf8mb4 e_ci		Yes	*NULL*		
4	job_slug	varchar(	utf8mb4 e_ci		Yes	*NULL*		
5	position	varchar(	utf8mb4 e_ci		Yes	*NULL*		
6	category	int(11)			Yes	*NULL*		
7	is_any_	varchar(	utf8mb4 e_ci		Yes	*NULL*		
8	salary	int(11)			Yes	0		
9	salary_u	int(11)			Yes	0		
10	is_negot	tinyint(4			Yes	0		
11	salary_c	enum(' ', 'weekly', nour...	utf8mb4 e_ci		Yes	*NULL*		
12	salary_c	varchar(	utf8mb4 e_ci		Yes	*NULL*		
13	vacancy	int(11)			Yes	*NULL*		

14	gender	enum(' 'female', nder',	utf8mb4 e_ci		No	*None*		
15	job_typ	enum('f ', ne', t', ..	utf8mb4 e_ci		Yes	Internsh		
16	job_dur	int(11)			Yes	*NULL*		
17	exp_lev	enum(' 'entry',	utf8mb4 e_ci		Yes	*NULL*		
18	Descript	Text	utf8mb4 e_ci		Yes			
19	Skills	Text	utf8mb4 e_ci		Yes			
20	Respons	Text	utf8mb4 e_ci		Yes			
21	educatio uirements	Text	utf8mb4 e_ci		Yes			
22	experien irements	Text	utf8mb4 e_ci		Yes			
23	addition rements	Text	utf8mb4 e_ci		Yes			
24	benefits	Text	utf8mb4 e_ci		Yes			
25	apply_i on	Text	utf8mb4 e_ci		Yes			
26	country	int(11)			Yes	*NULL*		

27	country	varchar(	utf8mb4 e_ci		Yes	*NULL*		
28	state_id	int(11)			Yes	*NULL*		
29	state_na	varchar(	utf8mb4 e_ci		Yes	*NULL*		
30	city_na	varchar(	utf8mb4 e_ci		Yes	*NULL*		
31	experien ired_year	tinyint(4			Yes	0		
32	experien	tinyint(4			Yes	0		
33	views	int(11)			Yes	0		
34	applyed	int(11)			Yes	0		
35	approve	Datetim			Yes	*NULL*		
36	deadline	Datetim			Yes	*NULL*		
37	status	tinyint(4			Yes	0		
38	job_id	varchar(	utf8mb4 e_ci		Yes	*NULL*		
39	is_prem	tinyint(4			Yes	*NULL*		
40	google om	varchar(	utf8mb4 e_ci		Yes	*NULL*		
41	created_	Timesta			Yes	*NULL*		
42	updated	Timesta			Yes	*NULL*		

Selanjutnya tabel *job_applications*, tabel ini memuat data user Mahasiswa yang melamar pada lowongan program magang kerja terkait, user_id dan employer_id berkorelasi dengan id pada tabel user, sedangkan job_id berkorelasi dengan id pada tabel *jobs.* Tampilan tabel *job_applications* dapat dilihat pada tabel 4.6.

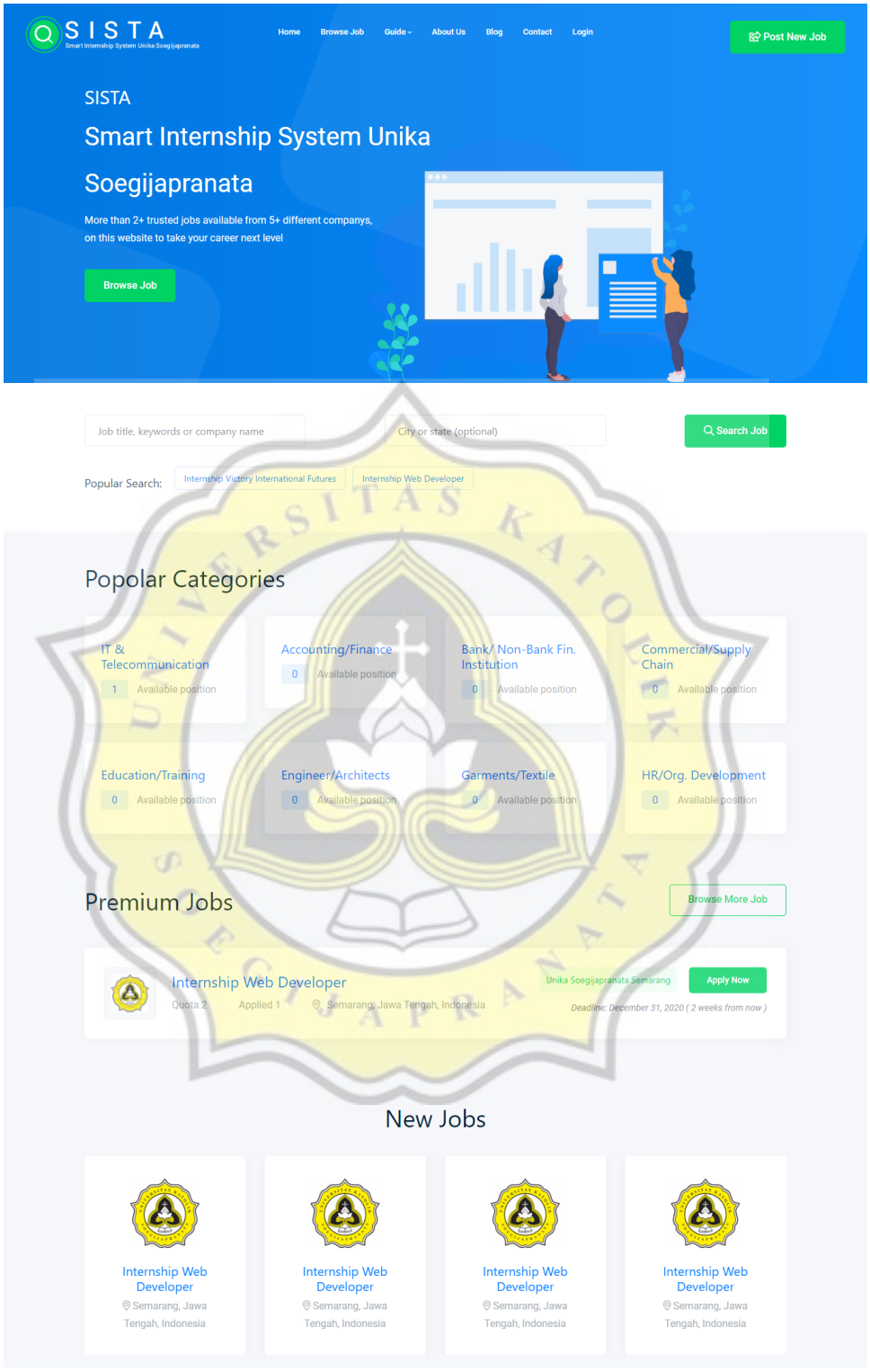
Tabel 4.3.6. Job applications

#	Name	Type	Collatio	Attribut	Null	Default	Comme	Extra
1	id	int(10)		UNSIG	No	*None*		AUTO_ MENT
2	job_id	int(11)			Yes	*NULL*		
3	employ	int(11)			Yes	*NULL*		
4	user_id	int(11)			Yes	*NULL*		
5	nim	varchar(	utf8mb de_ci		Yes	*NULL*		
6	progdi	varchar(	utf8mb de_ci		Yes	*NULL*		
7	faculty	int(11)			Yes	*NULL*		
8	name	varchar(	utf8mb de_ci		Yes	*NULL*		
9	email	varchar(	utf8mb de_ci		Yes	*NULL*		
10	ipk	varchar(	utf8mb de_ci		Yes	*NULL*		
11	phone_	varchar(	utf8mb de_ci		Yes	*NULL*		
12	messag	Text	utf8mb de_ci		Yes			

13	resume	varchar(	utf8mb de_ci		Yes	*NULL*		
14	photo	varchar(	utf8mb de_ci		Yes	*NULL*		
15	transkri	varchar(	utf8mb de_ci		Yes	*NULL*		
16	skpi	varchar(	utf8mb de_ci		Yes	*NULL*		
17	is_short	int(2)			Yes	0		
18	status	int(2)			No	0		
19	user_sta	int(2)			Yes	0		
20	created	Timesta			Yes	*NULL*		
21	updated	Timesta			Yes	*NULL*		

4.3.2 Halaman Utama

Ketika user pertama kali membuka halaman web, user langsung diarahkan ke halaman utama atau *home*, pada halaman utama memuat informasi kategoris, lowongan magang kerja yang tersedia, dan perusahaan yang telah melakukan registrasi, terdapat beberapa menu, jika user menekan menu tersebut akan dialihkan ke halaman menu terkait, menu tersebut yaitu, *browse job*, *guide*, *about us*, *blog*, *contact* dan *login.* Tampilan halaman utama dapat dilihat pada Gambar 4.14.



Gambar 4.3.1. Tampilan halaman utama

Karena aplikasi ini dikembangkan menggunakan framework Laravel yang menggunakan konsep MVC (*Model*, *View*, *Controller*) maka *script* yang digunakan untuk membuat tampilan halaman utama adalah sebagai berikut.

```
      public function index(){
          $categorie_tops  =  Category::orderBy('job_count',  'desc')-
>take(8)->get();
          $company_tops            =            Job::select('user_id')-
>selectRaw('sum(applyed)        as        jumlah')->groupBy('user_id')-
>orderBy('jumlah', 'desc')->with('employer')->get();
          $categories  =  Category::orderBy('category_name',  'asc')-
>get();
          $premium_jobs    =    Job::active()->premium()->orderBy('id',
'desc')->with('employer')->get();
          $regular_jobs    =    Job::active()->orderBy('id',    'desc')-
>with('employer')->take(15)->get();
          $total_jobs = Job::where('status', '=', '1')->count();
          $blog_posts     =     Post::whereType('post')->with('author')-
>orderBy('id', 'desc')->take(3)->get();
          $total_users  =  User::where('active_status',  '=',  '1')-
>count();
          $total_user_googles = User::where('user_type', '=', 'user')-
>count();
          $total_companys = User::whereNotNull('company')->count();
          $data['count'] = User::whereNotNull('company')->count();
          $views = Job::whereNotNull('views')->orderBy('views', 'desc')-
>take(7)->get();
          $total_applys = JobApplication::count();
          $packages = Pricing::all();
          return     view('home',     $data,     compact('categories',
'premium_jobs', 'regular_jobs', 'packages', 'blog_posts', 'total_users',
'total_companys',      'total_jobs',      'total_applys',      'views',
'categorie_tops', 'company_tops', 'total_user_googles' ));}
```

Gambar 4.3.2. Script untuk home controller

Gambar 4.15 merupakan *script* untuk menjalankan fungsi mengambil data pada model atau tabel terkait yang nantinya akan ditampilkan pada *script* tampilan halaman utama. Sedangkan *script* tampilan halaman utama terbagi menjadi beberapa bagian, akan ditampilkan dalam beberapa Gambar sebagai berikut.

```
<?php
$header_menu_pages = config('header_menu_pages');
?>
@if($header_menu_pages->count() > 0)
@foreach($header_menu_pages as $page)
<li><a href="{{ route('single_page', $page->slug) }}">{{ $page->title
}}</a></li>
@endforeach
@endif
<li><a href="{{route('blog_index')}}">Blog</a></li>

<li><a href="{{route('contact_us')}}">Contact</a></li>
<li> <a href="{{ route('login') }}">{{ __('app.login') }}</a></li>
</ul>
</nav>
```

Gambar 4.3.3. Script untuk header home.php

Gambar 4.16 merupakan *script* untuk bagian *header* pada halaman utama, *script* header sendiri dibuat terpisah agar saat membuat halaman baru *script* ini dapat dipanggil kembali tanpa harus menulis ulang *script* tersebut atau sering juga disebut sebagai fungsi *include.*

```
        <div class="catagory_area">
                <div class="container">
                    <form action="{{route('jobs_listing')}}" method="get">
                        <div class="row cat_search">
                            <div class="col-lg-20 col-md-4">
                                <div class="single_input">
                                    <input      type="text"      name="q"
placeholder="@lang('app.job_title_placeholder')">
                                </div>
                            </div>
                            <div class="col-lg-20 col-md-4">
                                <div class="single_input">
                                    <input   type="text"   name="location"
placeholder="@lang('app.job_location_placeholder')">
                                </div>
                            </div>
                            <div class="col-lg-auto col-md-12">
                                <div class="job_btn">
                                    <button  type="submit"  class="boxed-
btn3"><i       class="far       fa-search"></i>       @lang('app.search')
@lang('app.job')</button>
                                </div>
                            </div>
                        </div>
                    </form>
                    <div class="row">
                        <div class="col-lg-12">
                            <div  class="popular_search  d-flex  align-items-
center">
                                <span>Popular Search:</span>
                                <ul>
                                    @foreach($views as $view)
                                    <li><a      href="{{route('job_view',
$view->job_slug)}}">{{ $view->job_title }}</a></li>
                                    @endforeach
                                </ul>
                            </div>
                        </div>
                    </div>
                </div>
        </div>
```

Gambar 4.3.4. Script untuk category home.php

Pada Gambar *4.17 script* untuk menampilan area category pada halaman utama, setiap kategori yang diambil oleh controller akan ditampilkan pada halaman utama. Kategori diambil oleh *controller* dari model atau tabel *categories* dengan nama variabel *views.*

```
<a href="{{route('job_view', $job->job_slug)}}">
<h4>{!! $job->job_title !!}</h4></a>
<div class="links_locat d-flex align-items-center">
@if($job->vacancy)
<div class="location">
<p> @lang('app.job_vacancy') {!! $job->vacancy !!} </p>
</div>
@endif
@if($job->applyed)
<div class="location">
<p> @lang('app.job_applied') {!! $job->applyed !!} </p>
</div>
@endif
@if ($job->is_any_where == 1)
<div class="location">
<p> <i class="fal fa-map-marker-alt"></i> @lang('app.anywhere') </p>
</div>
@else
<div class="location">
<p> <i class="fal fa-map-marker-alt"></i>
@if($job->city_name)
{!! $job->city_name !!},
@endif
@if($job->state_name)
{!! $job->state_name !!},
@endif
@if($job->state_name)
{!! $job->country_name !!}
@endif
</p>
</div>
@endif
```

Gambar 4.3.5. Script untuk premium job home.php

Gambar 4.18 merupakan *script* untuk yang berfungsi untuk daftar premium job pada halaman utama, data diambil controller dari model atau tabel *jobs*, setelah itu di tampilkan pada halaman utama yang dipanggil menggunakan nama variabel job.

```
@if($regular_jobs->count())
<div class="featured_candidates_area">
<div class="container">
<div class="row">
<div class="col-lg-12">
<div class="section_title text-center mb-40">
<h3>@lang('app.new_jobs')</h3>
</div>
</div>
</div>
<div class="row">
<div class="col-lg-12">
<div class="candidate_active owl-carousel">
@foreach($regular_jobs as $regular_job)
<div class="single_candidates text-center">
<div class="thumb">
<img src="{{ $regular_job->employer->logo_url }}" alt="{!! $regular_job-
>job_title !!}">
</div>
<a href="{{route('job_view', $regular_job->job_slug)}}"><h4>{!!
$regular_job->job_title !!}</h4></a>
@if ($regular_job->is_any_where == 1)
<div class="location">
<p> <i class="fal fa-map-marker-alt"></i> @lang('app.anywhere') </p>
</div>
@else
<div class="location">
<p> <i class="fal fa-map-marker-alt"></i>
@if($regular_job->city_name)
{!! $regular_job->city_name !!},
@endif
@if($regular_job->state_name)
{!! $regular_job->state_name !!},
@endif
@if($regular_job->state_name)
{!! $regular_job->country_name !!}
@endif
```

Gambar 4.3.6. Script untuk reguler job home.php

Gambar *4.*19 *script* yang berfungsi untuk menampilkan daftar regular job pada halaman utama, data diambil controller dari model atau tabel *jobs*, setelah itu di tampilkan pada halaman utama yang dipanggil menggunakan nama variabel regular_job.

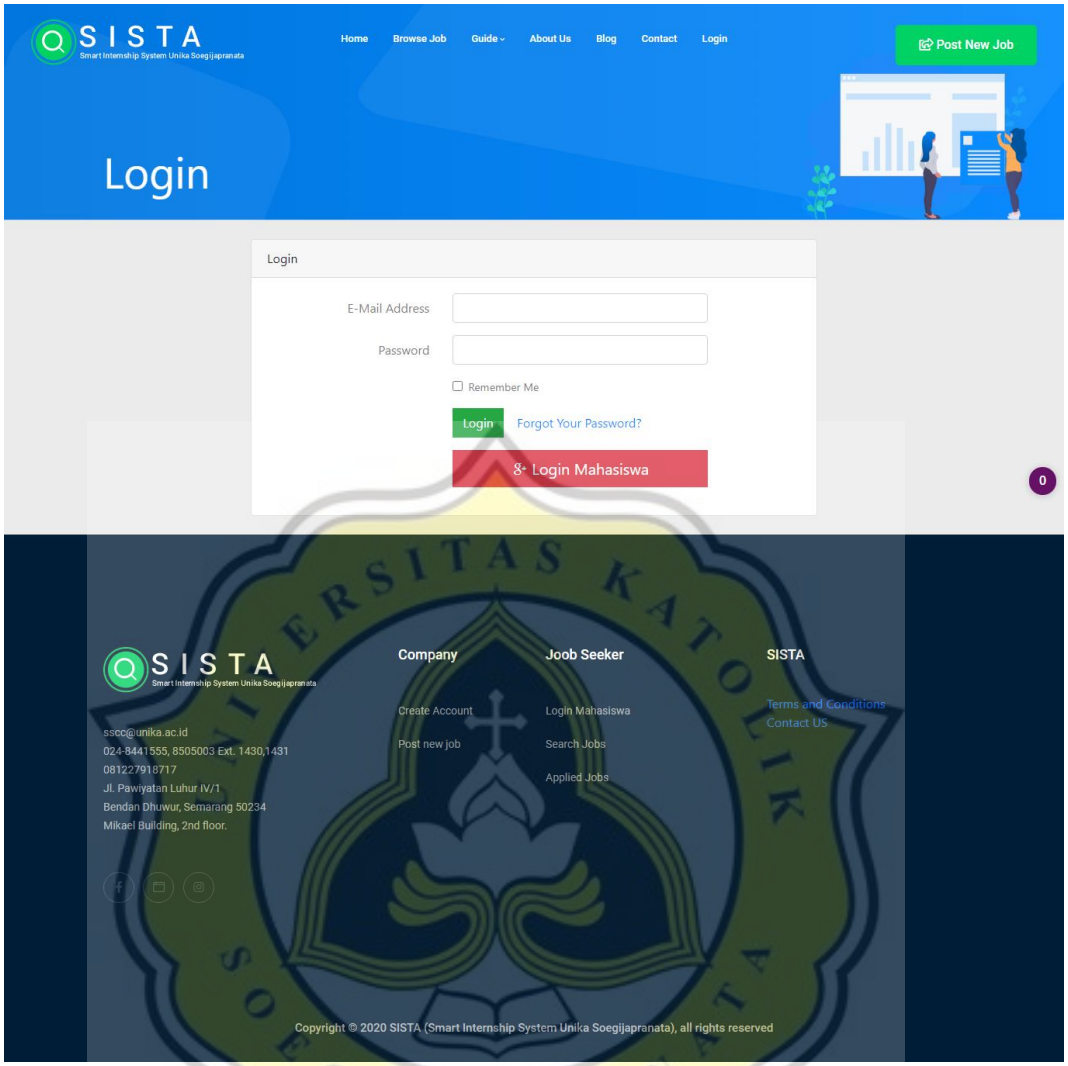
```
<div class="top_companies_area">
<div class="container">
<div class="row align-items-center mb-40">
<div class="col-lg-6 col-md-6">
<div class="section_title">
<h3>Top Companies</h3>
</div>
</div>
{{-- <div class="col-lg-6 col-md-6">
<div class="brouse_job text-right">
<a href="jobs.html" class="boxed-btn4">Browse More Job</a>
</div>
</div> --}}
</div>
<div class="row">
@foreach($company_tops as $company_top)
<div class="col-lg-4 col-xl-3 col-md-6">
<div class="single_company">
<div class="thumb">
<img src="{{ $company_top->employer->logo_url }}" alt="{{ $company_top-
>company }}">
</div>
<a href="{{route('jobs_by_employer', $company_top->employer-
>company_slug)}}"><h3>{{ $company_top->employer->company }}</h3></a>
<p> <span>{{ $company_top->jumlah }}</span> Applyed position</p>
</div>
</div>
@endforeach
</div>
</div>
</div>
```

Gambar 4.3.7. Script untuk company home.php

Gambar 4.20 merupakan *script* yang berfungsi untuk menampilan daftar perusahaan yang telah melakukan registrasi pada halaman utama, data diambil controller dari model atau tabel *user*, setelah itu di tampilkan pada halaman utama yang dipanggil menggunakan nama variabel company_tops.

4.3.3 Halaman Utama

Dari halaman utama user dapat melihat tombol login, yang dimana jika tombol itu diklik akan mengarahkan user ke halaman login. Pada halaman ini user harus mengisi *username* dan *password*, dan untuk Mahasiswa menggunakan *social login* akun gmail student Unika Soegijapranata.



Gambar 4.3.8. Halaman login user

Halaman login dibagi menjadi dua *controller*, *controller* pertama mengatur login menggunakan username dan password sedangkan *controller* kedua mengatur login menggunakan alamat email student Unika Soegijapranata untuk Mahasiswa.

```php
public function loginView()
{
    return view('login/main', [
        'layout' => 'login'
    ]);
}

/**
 * Authenticate login user.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function login(LoginRequest $request)
{
    if (!\Auth::attempt([
        'email' => $request->email,
        'password' => $request->password
    ])) {
        throw new \Exception('Wrong email or password.');
    }
}

/**
 * Logout user.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function logout()
{
    \Auth::logout();
    return redirect('login');
}
```

Gambar 4.3.9. Script untuk auth login controller

Gambar 4.22 *script* untuk mengatur user login menggunakan username dan password, fungsinya cukup sederhana dimana, jika username dan password yang dimasukan kedalam kolom text sama dengan yang ada di database user maka sistem akan menganggap user tersebut valid dan nantinya sistem akan mengirim data yang dibutuhkan untuk selanjutnya diarahkan ke halaman berikutnya jika tidak valid sistem akan menampilkan pesan error *"Wrong email or password"* dan tidak mengirimkan data apapun.

```
class GoogleController extends Controller
{
public function redirectToGoogle()
{
        return Socialite::driver('google')->redirect();
}
        public function handleGoogleCallback()
{
try {
        $user = Socialite::driver('google')->user();
        $finduser = User::where('google_id', $user->id)->first();
if (str_contains($user->email,'@student.unika.ac.id')) {
       if($finduser){
         Auth::login($finduser);
         return redirect('/dashboard');
       }else{
         $nim_raw = explode("@", $user->email);
         $code_progdi = substr($nim_raw[0], 2, -4);
         $progdi = Progdi::select('*')->where('code', '=', $code_progdi)-
         >first();
         $codetahun = substr($nim_raw[0], 0, -6);
         $codenim = substr($nim_raw[0], 4);
         $nim = $codetahun.".".strtoupper($code_progdi).".".$codenim;
         $newUser = User::create([
               'name'              => $user->name,
               'nim'               => $nim,
               'progdi'            => $progdi->nama_progdi,
               'faculty'           => $progdi->faculty,
               'photo'             => $user->avatar,
               'email'             => $user->email,
               'google_id'         => $user->id,
               'email_verified_at' => now(),
               'password'          => Hash::make('ilham211')
              ]);
         Auth::login($newUser);
         return redirect('/dashboard');
         }
}
else{
return
        redirect(route('login'))->with('error',
__('app.access_restricted_login'));
}
} catch (Exception $e) {
dd($e->getMessage());
}}}
```

Gambar 4.3.10. Script untuk auth login google controller

Gambar 4.23 merupakan *script controller* untuk mengatur user login menggunakan alamat email student Unika Soegijapranata. Fungsinya *controller* adalah mendapatkan respon dari *google+ api,* dimana respon didapatkan jika user

login ke web gmail yang nantinya respon tersebut berisi token dan email, token didapatkan jika user berhasil login ke web gmail, kemudian data tersebut diolah oleh sistem untuk dimasukan ke database user, fungsinya untuk memberikan akses kepada user terkait dengan token yang sudah diberikan oleh api google, namun jika user gagal login ke web gmail maka sistem tidak akan memperoleh data token yang nantinya sistem tidak akan menyimpan data apapun, halaman login akan menampilkan pesan *access restricted* dan user tidak akan bisa login.

```
        <div class="card-body">
        <form method="POST" action="{{ route('login') }}">
        @csrf

        <div class="form-group row">
        <label for="email" class="col-sm-4 col-form-label text-md-
right">{{ __('E-Mail Address') }}</label>

        <div class="col-md-6">
        <input id="email" type="email" class="form-control{{ $errors-
>has('email') ? ' is-invalid' : '' }}" name="email" value="{{
old('email') }}" required autofocus>

        @if ($errors->has('email'))
        <span class="invalid-feedback" role="alert">
        <strong>{{ $errors->first('email') }}</strong>
        </span>
        @endif
        </div>
        </div>

        <div class="form-group row">
        <label for="password" class="col-md-4 col-form-label text-md-
right">{{ __('Password') }}</label>

        <div class="col-md-6">
        <input id="password" type="password" class="form-control{{
$errors->has('password') ? ' is-invalid' : '' }}" name="password"
required>
```

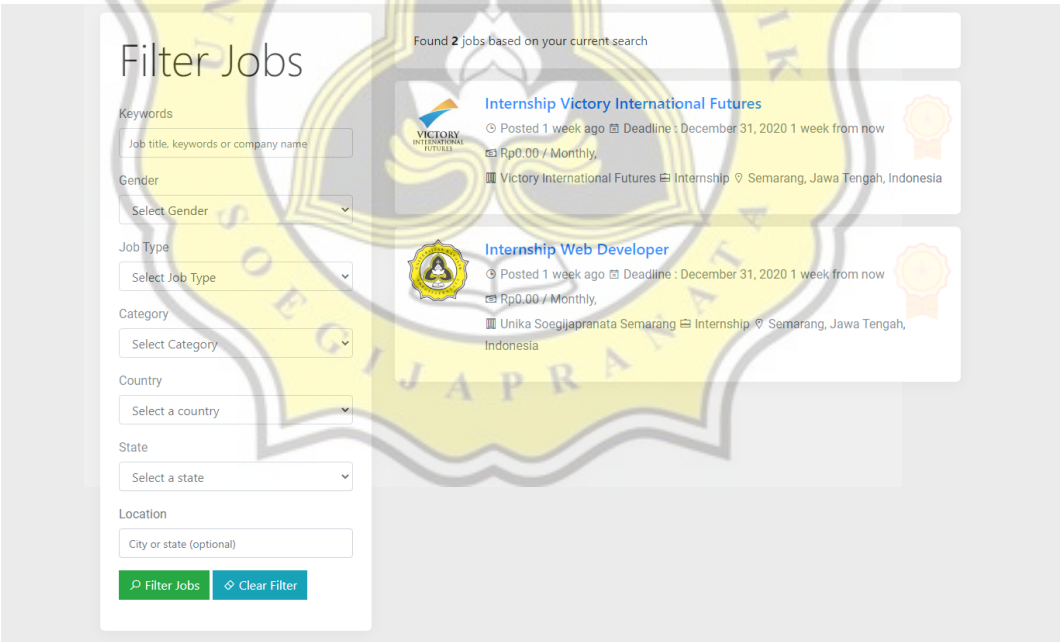
Gambar 4.3.11. Script untuk halaman login.php

Gambar 4.24 merupakan *script* untuk tampilan login, terdapat kolom username dan password serta dua tombol login yaitu tombol login dan login

Mahasiswa. Tombol login sendiri jika data dikolom username dan password benar maka akan dialihkan ke halaman dashboard sesuai user, sedangkan login Mahasiswa jika diklik maka akan dialihkan ke halaman login gmail, jika login berhasil user baru dipindahkan ke web sista kembali lalu otomatis membuka dashboard user.

4.3.4 Halaman *Browse Jobs*

Halaman *browse jobs* digunakan untuk user mencari lowongan program magang kerja sesuai filter yang diinginkan, filter yang tersedia yaitu *keyword*, *gender*, *job type*, *category*, *country*, *state*, dan *location*.



Gambar 4.3.12. Halaman browse jobs

```php
public function jobsListing(Request $request){

    $title = "Browse Jobs";


    $categories = Category::orderBy('category_name', 'asc')->get();
    $countries = Country::where("country_code", "=", "ID")->get();
    $states = State::where("country_id", "=", "102")->get();
    $old_country = false;
    if (old('country')){
        $old_country = Country::find(old('country'));
    }

    $jobs = Job::active();
    $job_counts = Job::count();

    if ($request->q){
        $jobs = $jobs->where(function ($query) use($request){
            $query->where('job_title', 'like', "%{$request->q}%")
                ->orWhere('position', 'like', "%{$request->q}%")
                ->orWhere('description', 'like', "%{$request->q}%");
        });}

    if ($request->location){
        $jobs = $jobs->where('city_name', 'like', "%{$request->location}%");
    }

    if ($request->gender){
        $jobs = $jobs->whereGender($request->gender);
    }
    if ($request->exp_level){
        $jobs = $jobs->whereExpLevel($request->exp_level);
    }
    if ($request->job_type){
        $jobs = $jobs->whereJobType($request->job_type);
    }
    if ($request->country){
        $jobs = $jobs->whereCountryId($request->country);
    }
    if ($request->state){
        $jobs = $jobs->whereStateId($request->state);
    }
    if ($request->category){
        $jobs = $jobs->whereCategoryId($request->category);
    }

    $jobs = $jobs->orderBy('id', 'desc')->with('employer')->paginate(5);

    return  view('jobs',  compact('title',  'jobs','categories',  'countries',
'old_country', 'job_counts', 'states' ));}
```

Gambar 4.3.13. Script untuk job listing controller

Gambar 4.26 merupakan *script* controller untuk mengatur pencarian lowongan program magang kerja, dimana pada setiap kolom filter yang diisi oleh user akan dikirim ke controller ini, setelah itu controller akan mencari lowongan

program magang kerja yang sesuai dengan filter di database yang nantinya akan mengirim data tersebut ke tampilan halaman *browse jobs.*

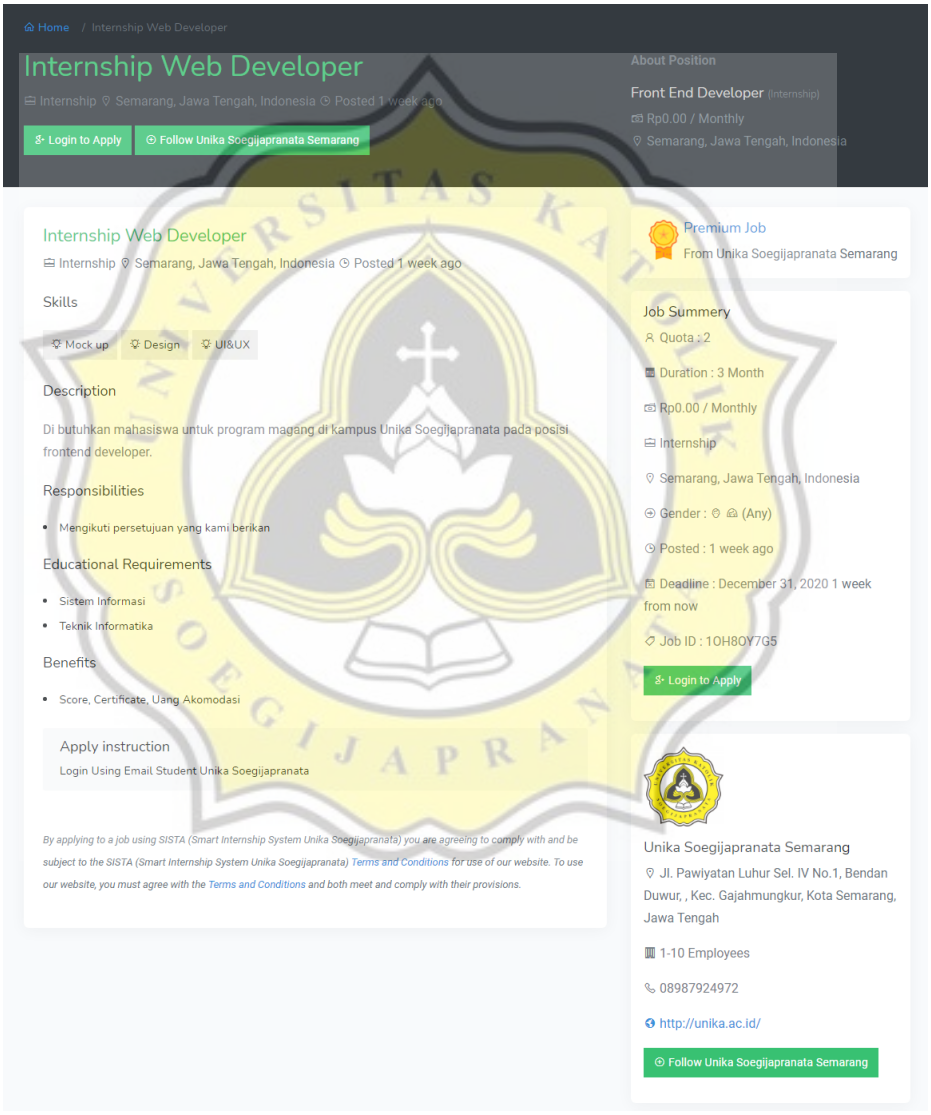
```
<div class="row">
       <div class="col-md-8">
       <h1 class="text-success">{!! $job->job_title !!}</h1>
       <p class="text-muted">
       <i class="la la-briefcase"></i> @lang('app.'.$job->job_type)

       <i class="la la-map-marker"></i>
       @if($job->city_name)
       {!! $job->city_name !!},
       @endif
       @if($job->state_name)
       {!! $job->state_name !!},
       @endif
       @if($job->country_name)
       {!! $job->country_name !!}
       @endif

       <i class="la la-clock-o"></i> @lang('app.posted') {{$job-
>created_at->diffForHumans()}}
       </p>
```

Gambar 4.3.14. Script untuk browser job.php

Gambar 4.27 *script* untuk tampilan *browse jobs* dimana *script* tersebut berfungsi untuk menampilkan kolom pencarian berdasarkan filter *keyword, gender, job type, category, country, state,* dan *location.* Serta menampilkan lowongan kerja yang tersedia yang didapat dari *JobListing Controller* yang disimpan dengan variabel *jobs*.

4.3.5 Halaman *Jobs*

Halaman *jobs* merupakan halaman untuk menampilkan detail lowongan program magang kerja yang tersedia, dan tombol untuk melamar ke lowongan terkait.



Gambar 4.3.15. Halaman jobs

```
public function view($slug = null){
    $job = Job::whereJobSlug($slug)->first();
    if ( ! $slug || ! $job || (! $job->is_active() && ! $job->can_edit()) ){
        abort(404);}
    $job->views++;
    $job->save();
    $title = $job->job_title;
    try{
        $user_id = Auth::user()->id;
        $applications  =  JobApplication::whereUserId($user_id)->where('job_id',
'=', $job->id)->count();
        return view('job-view', compact('title', 'job', 'applications'));
    }catch (\Exception $e){
        return view('job-view', compact('title', 'job'));}
```

Gambar 4.3.16. Script untuk job view controller

Gambar 4.29 merupakan *script* controller yang berfungsi mengambil data lowongan program magang kerja tertentu dari database yang kemudian akan ditampilkan pada halaman Jobs, data yang dikirim merupakan data detail lowongan program magang kerja yaitu judul lowongan, deskripsi, durasi, kuota, tipe lowongan, penempatan *deadline* dan tanggal lowongan dibuat serta profil perusahaan dari nama perusahaan, alamat, ukuran perusahaan, nomor, email, dan web perusahaan.

```
public function applyJob(Request $request){
$rules = [
];
$validator = Validator::make($request->all(), $rules);
$user_id = 0;
if (Auth::check()){
$user_id = Auth::user()->id;

session()->flash('job_validation_fails', true);

if ($validator->fails()){
return redirect()->back()->withInput($request->input())->withErrors($validator);
}

if ($request->hasFile('resume')){
// Resume
$resume = $request->file('resume');
$valid_extensions = ['pdf','doc','docx'];
if ( ! in_array(strtolower($resume->getClientOriginalExtension()),
$valid_extensions) ){
session()->flash('job_validation_fails', true);
return redirect()->back()->withInput($request->input())->with('error',
trans('app.resume_file_type_allowed_msg') ) ;
}
```

```php
// Unika job fair costume
$application_data = [
'job_id'                => $request->job_id,
'employer_id'           => $job->user_id,
'user_id'               => $user_id,
'nim'                   => $nim,
'progdi'                => $progdi->nama_progdi,
'faculty'               => $progdi->faculty,
'name'                  => $user_name,
'email'                 => $user_email,
'ipk'                   => $ipk,
'phone_number'          => $request->phone_number,
'photo'                 => $image_name,
'resume'                => $resume_name,
'is_shortlisted'        => 0,
'status'                => 0,
];

JobApplication::create($application_data);

// Unika Job fair Costume
// Count Applyed jobs
$job->applyed++;
$job->save();
```

Gambar 4.3.17. Script untuk apply job controller

Gambar 4.30 merupakan *script* yang berfungsi untuk melakukan lamaran ke lowongan program magang kerja. Setelah user menekan tombol "apply" dan user mengisi kolom lamaran maka sistem akan melakukan validasi data, jika data yang dimasukan sesuai maka sistem akan mencari data ipk melalui api unika, api tersebut meminta nim untuk mendapatkan data ipk dengan nim terkait, lalu data file seperti resume dan foto akan disimpan pada folder asset dengan nama file *randomstring*-nama file asli. Setelah semua proses berjalan dan data yang dibutuhkan terkumpul, sistem akan menyimpan semua data tersebut pada tabel *job_applications*, selain itu sistem juga akan mengupdate jumlah data kolom applied ditabel job pada lowongan program magang kerja terkait.

```
@if(auth()->check() && auth()->user()->is_user())
                        @if ($applications > 0)
                            <button type="button" class="btn btn-warning"> <i
class="la la-clipboard"></i> @lang('app.already_applied') </button>
                        @else
                            <button type="button" class="btn btn-success" data-
toggle="modal" data-target="#applyJobModal" ><i class="la la-calendar-plus-
o"></i> @lang('app.apply_online') </button>
                        @endif
                    @elseif(auth()->check() && auth()->user()->is_employer())
                        @if($job->applyed)
                            <button type="button" class="btn btn-warning"> <i
class="la la-clipboard"></i> @lang('app.job_applied') {!! $job->applyed !!}
</button>
                        @else
                            <button type="button" class="btn btn-warning"> <i
class="la la-clipboard"></i> @lang('app.job_applied') 0 </button>
                        @endif
                    @else
                        <a href="{{route('auth_google')}}" class="btn btn-
success" ><i class="la la-google-plus"></i> @lang('app.apply_online_redirect')
</a>
                    @endif
            </div>
```

Gambar 4.3.18. Script untuk job view.php

Gambar 4.31 merupakan *script* pada bagian header untuk tampilan halaman job, pada *script* tersebut mengambil data melalu view controller berisi kolom judul lowongan dengan variabel job_titile, tipe lowongan dengan variabel job_type, lokasi lowongan dengan variabel city_name, state_name dan country_name, waktu pembuatan lowongan dengan nama variabel created_at, jumlah pelamar dengan variabel job_applyed, kemudian data pendukung seperti posisi lowongan dengan variabel job_position, gaji lowongan dengan variabel job_salary_currency. Selain itu ada tombol applied dimana jika tombol tersebut ditekan akan muncul *pop up* kolom data untuk diisi.

```
@if(session('error'))
<div class="alert alert-warning">{{session('error')}}</div>
@endif

<div class="form-group {{ $errors->has('phone_number')? 'has-error':'' }}">
<label for="phone_number" class="control-
label">@lang('app.phone_number'):</label>
<input type="text" class="form-control {{e_form_invalid_class('phone_number',
$errors)}}" id="phone_number" name="phone_number"
value="{{old('phone_number')}}" placeholder="@lang('app.phone_number')">
{!! e_form_error('phone_number', $errors) !!}
</div>

<div class="form-group {{ $errors->has('image')? 'has-error':'' }}">
<label for="image" class="control-label">@lang('app.new_photo'):</label>
<div class='file-input'>
<input type="file" accept=".jpg,.jpeg,.png" class="form-control
{{e_form_invalid_class('image', $errors)}}" id="image" name="image">
<span class='button'>Choose</span>
<span class='label' data-js-label>No file selected</label>
</div>
<p class="text-muted">@lang('app.image_file_types')</p>
{!! e_form_error('image', $errors) !!}
</div>
```

Gambar 4.3.19. Script untuk apply job modal.php

Gambar 4.32 adalah *script* tampilan pop up untuk kolom data lamaran, kolom tersebut berisi nomor kontak, file foto terbaru, file resume.

```
                @if($job->skills)
                    <div class="job-view-single-section my-4">
                        <h5 class="mb-4">@lang('app.skills')</h5>
                        @php
                            $skills = explode(',', $job->skills);
                        @endphp

                        @if(is_array($skills) && count($skills))
                            @foreach($skills as $skill)
                                <span class="single-skill"><i class="la la-
lightbulb-o"></i> {{$skill}}</span>
                            @endforeach
                        @endif
                    </div>
                @endif

                @if($job->description)
                    <div class="job-view-single-section my-4">
                        <h5 class="mb-4">@lang('app.description')</h5>
                        <p>
                            {!! nl2br($job->description) !!}
                        </p>
                    </div>
                @endif

                @if($job->responsibilities)
                    <div class="job-view-single-section my-4">
                        <h5 class="mb-4">@lang('app.responsibilities')</h5>
                        {!! $job->nl2ulformat($job->responsibilities) !!}
                    </div>
                @endif

                @if($job->educational_requirements)
                    <div class="job-view-single-section my-4">
                        <h5 class="mb-
4">@lang('app.educational_requirements')</h5>
                        {!! $job->nl2ulformat($job->educational_requirements)
!!}
                    </div>
                @endif

                @if($job->experience_requirements)
                    <div class="job-view-single-section my-4">
                        <h5 class="mb-
4">@lang('app.experience_requirements')</h5>
                        {!! $job->nl2ulformat($job->experience_requirements) !!}
                    </div>
                @endif
```

Gambar 4.3.20. Script untuk job deteail.php

Gambar 4.33 *script* untuk tampilan pada halaman jobs, script tersebut akan menampilkan detail lowongan magang kerja yang diambil melalui controller dan disimpan dengan nama variabel job.

4.3.6 Halaman *Dashboard*

Halaman *dashboard* memiliki berbagai fungsi, fungsi dibedakan dengan peran masing – masing user, untuk *dashboard* admin memiliki fungsi untuk mengatur sistem, *dashboard* perusahaan memiliki fungsi untuk mengolah data lowongan program magang kerja, sedangkan *dashboard* user atau Mahasiswa berfungsi untuk menampilan serta mengolah data lamaran yang telah dibuat pada setiap program magang kerja.

Gambar 4.3.21. Halaman dashboard

```
        public function dashboard(){
        $nama_job = $company_tops_jobs->pluck('employer')->pluck('company');
        $jumlah_job = $company_tops_jobs->pluck('jumlah');

        $nama_view = $company_tops_views->pluck('employer')->pluck('company');
        $jumlah_view = $company_tops_views->pluck('jumlah');

        $nama_applyed = $company_tops_applyeds->pluck('employer')-
>pluck('company');
        $jumlah_applyed = $company_tops_applyeds->pluck('jumlah');

        $totalJob = Job::Where('user_id', '=', $user->id)->count();
        $totalJobActive = Job::Where('user_id', '=', $user->id)->active()-
>count();
        $totalApplicants_employer = JobApplication::where('employer_id', '=',
$user->id)->count();
        $totalApplicants_user = JobApplication::where('user_id', '=', $user-
>id)->count();
        $totalJobView_employer = Job::Where('user_id', '=', $user->id)-
>sum('views');

        $title_job_employer = $jobs->pluck('job_title');
        $view_job_employer = $jobs->pluck('views');
        $applyed_job_employer = $jobs->pluck('applyed');

        $created_by = User::select('*')->where('id', '=', $user_id)->get();

        $data = [
        'usersCount' => User::count(),
        'totalPayments' => Payment::success()->sum('amount'),
        'totalJobview' => Job::sum('views'),
        'activeJobs' => Job::active()->count(),
        'totalJobs' => Job::count(),
        'employerCount' => User::employer()->count(),
        'agentCount' => User::agent()->count(),
        'totalApplicants' => JobApplication::count(),
        'totalBlog' =>  Post::count(),
        'totalBlogview' =>  Post::where('status', '=', '1')->sum('views'),
        ];

        return view('admin.dashboard', $data, compact('flagjobs',
'company_tops_applyeds', 'company_tops_views', 'jumlah_applyed', 'nama_applyed',
'nama_view', 'jumlah_view', 'nama_job', 'jumlah_job', 'company_tops_jobs',
'regular_jobs', 'user', 'is_user_id_view', 'totalJob', 'totalJobActive',
'totalApplicants_employer', 'totalApplicants_user', 'totalJobView_employer',
'jobs', 'applications', 'progdis', 'years', 'title_job_employer',
'view_job_employer', 'applyed_job_employer', 'created_by', 'job_id_facultys'));}
```

Gambar 4.3.22. Script untuk dashboard controller

Gambar 4.35 merupakan *script* controller untuk tampilan dashboard, controller tersebut mengambil data akun dari tabel *user* dengan nama variabel *user*, daftar lowongan magang kerja dari tabel jobs dengan nama variabel jobs dan daftar Mahasiswa yang melamar pada tabel job_applicant dengan nama variabel *applications.* Nantinya data tersebut akan dipanggil pada *script* tampilan dashboard.

```
@php
try {
$persenapplied = $totalJob/$totalJobs*100;
} catch (Exception $e) {
$persenapplied = 0;
}
@endphp
<div class="report-box__indicator bg-theme-9 tooltip cursor-pointer"
title="{{round($persenapplied)}}% From total jobs in this site">
{{round($persenapplied)}}% </div>
@php
try {
$persenapplied = $totalJobView_employer/$totalJobview*100;
} catch (Exception $e) {
$persenapplied = 0;
}
@endphp
<div class="report-box__indicator bg-theme-9 tooltip cursor-pointer"
title="{{round($persenapplied)}}% From total views in this site">
{{round($persenapplied)}}% </div>
@foreach($jobs as $job)
<option value="{{$job->id}}">{{$job->job_title}}</option>
@endforeach
@foreach ($jobs as $job)
<div class="flex items-center">
<span class="truncate">{{$job->job_title}}</span>
<div class="h-px flex-1 border border-r border-dashed border-gray-300 mx-3
xl:hidden"></div>
<span class="font-medium xl:ml-auto">{{$job->applyed}}</span>
</div>
@endforeach
@foreach ($jobs as $job)
<div class="flex items-center">
<span class="truncate">{{$job->job_title}}</span>
<div class="h-px flex-1 border border-r border-dashed border-gray-300 mx-3
xl:hidden"></div>
<span class="font-medium xl:ml-auto">{{$job->views}}</span>
</div>
```

Gambar 4.3.23. Script untuk dashboard card.php

Gambar 4.36 merupakan *script* untuk membuat tampilan card pada dashboard yang berisi mengenai data profil yaitu *total jobs*, *active jobs*, *total job view* dan *applicants.* Data tersebut diambil dari dashboard controller.

```
<!-- BEGIN: JOB CHART -->
<div class="col-span-12 lg:col-span-6 mt-8">
<div class="intro-y block sm:flex items-center h-10">
<h2 class="text-lg font-medium truncate mr-5">
@lang('app.applicant_chart')
</h2>
</div>
<div class="intro-y box p-5 mt-12 sm:mt-5">
<div class="flex flex-col xl:flex-row xl:items-center">
<div class="flex">
<div>
<div class="text-theme-20 dark:text-gray-300 text-lg xl:text-xl font-
bold">{{$totalJobActive}}</div>
<div class="text-gray-600 dark:text-gray-600">Your Active Jobs</div>
</div>
<div class="w-px h-12 border border-r border-dashed border-gray-300 dark:border-
dark-5 mx-4 xl:mx-6"></div>
<div>
<div class="text-gray-600 dark:text-gray-600 text-lg xl:text-xl font-
medium">{{$activeJobs}}</div>
<div class="text-gray-600 dark:text-gray-600">Total Active Jobs</div>
</div>
</div>

<div class="dropdown xl:ml-auto mt-5 xl:mt-0">
<select class="input border w-60" id="dd">
<option value="" selected="selected">Filter By Active Job</option>
@foreach($jobs as $job)
<option value="{{$job->id}}">{{$job->job_title}}</option>
@endforeach
</select>
</div>
</div>
<div class="report-chart">
<canvas id="chartContainer" height="160" class="mt-6"></canvas>
</div>
</div>
</div>
<!-- END: JOB CHART -->
```

Gambar 4.3.24. Script untuk graphic dashboard controller

Gambar 4.37 merupakan *script* untuk menampilkan data job dengan tampilan grafik, data diambil dari dashboard controller, data disimpan dengan nama variabel jobs lalu diolah dengan tampilan grafik menggunakan plugin bernama chart.js yang dapat dipanggil melalui script canvas, untuk nama canvas grafik batang adalah chartContainer sedangkan nama canvas grafik lingkaran atau sering disebut donat dan pie adalah jobs_by_employer_applyed dan jobs_by_employer_view.

4.3.7 Halaman *Premium Data*

Halaman *premium data* merupakan halaman khusus untuk perusahaan dimana isi dari halaman tersebut adalah jumlah semua lowongan magang kerja, jumlah semua lowongan magang kerja yang aktif, jumlah total pengunjung web, jumlah total pengunjung yang melamar, perusahaan dengan lowongan magang kerja terbanyak, perusahaan dengan jumlah pengunjung terbanyak, perusahaan dengan pelamar terbanyak, serta daftar mahasiswa dengan nim tertinggi per-angkatan dan daftar lulusan Unika Soegijapranata terbaru.

Gambar 4.3.25. Halaman premium data

```
        public function json_top_student($id, $angkatan){
            $code_progdi = ucfirst($id);
            $tahun_angkatan = substr($angkatan, -2);
            $response                                                    =
Http::get('http://API_URL/skripsi/api/magang_ipk.php?a='.$tahun_angkatan.'&p='.$
code_progdi)->json();
return $response;}
```

Gambar 4.3.26. Script untuk top student controller

```
<!-- BEGIN: Top Student List -->
<div class="col-span-12 mt-6">
<div class="intro-y block sm:flex items-center h-10">
<h2 class="text-lg font-medium truncate mr-5">
@lang('app.top_student_list')
</h2>

<div class="dropdown xl:ml-auto mt-5 xl:mt-0">
<select class="input border w-60" id="progdi_top_student_list">
<option value="" selected="selected">Filter By Progdi</option>
@foreach($progdis as $progdi)
<option value="{{$progdi->code}}">{{$progdi->nama_progdi}}</option>
@endforeach
</select>
</div>

<div class="dropdown xl:ml-auto mt-5 xl:mt-0">
<select class="input border w-60" id="class_top_student_list">
<option value="" selected="selected">Filter By Class</option>
@foreach($years as $year)
<option value="{{$year}}">{{$year}}</option>
@endforeach
</select>
</div>

</div>
<div class="intro-y datatable-wrapper box p-5 mt-5">
<!-- BEGIN: Datatable -->
<table class="table table-report table-report--bordered display datatable w-
full" id="top_student_list">
<thead>
<tr>
<th>@lang('app.name')</th>
<th>@lang('app.nim')</th>
<th>@lang('app.ipk')</th>
<th>@lang('app.email')</th>
</tr>
</thead>
</table>
<!-- END: Datatable -->
</div>
</div>
<!-- END: Top Student List  -->
```

Gambar 4.3.27. Script untuk top student dashboard.php

Gambar 4.39 merupakan *script* untuk mengambil data Mahasiswa Unika Soegijapranata yang diambil dari API, respon data berupa json yang berisi nama, nim, ipk serta email yang nantinya diolah dan di tampilkan di tabel. *Script* tampilan dapat dilihat pada *Gambar 4*.40.

```
      public function json_lulus($id){
          $bulan = now()->month;
          $tahun = now()->year;
          $code_progdi = ucfirst($id);
          $response                                                    =
Http::get('http://API_URL/skripsi/api/lulus_ipk.php?m='.$bulan.'&y='.$ta
hun.'&p='.$code_progdi)->json();
          return $response;}
```

Gambar 4.3.28. Script untuk lulusan controller

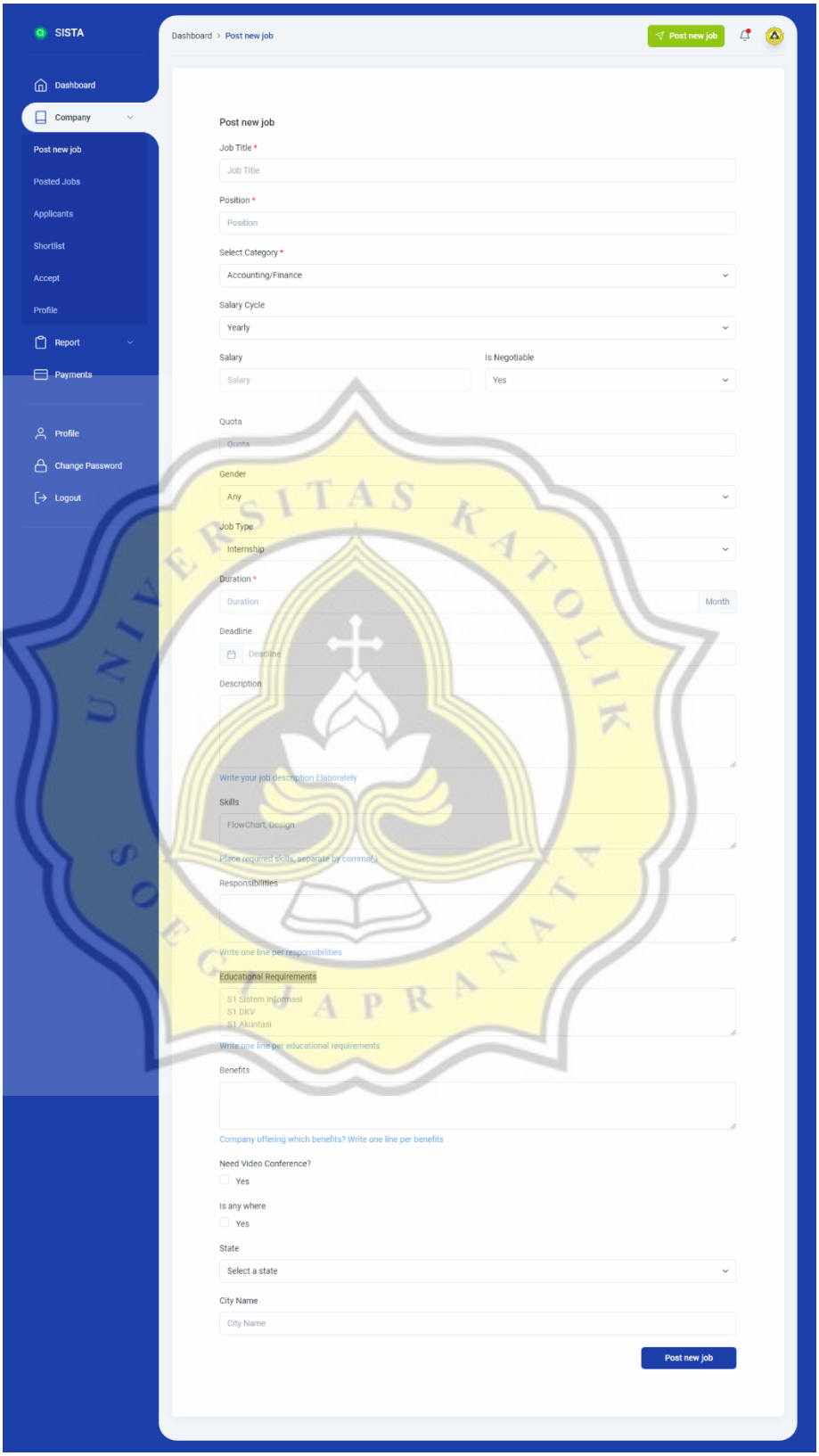
```
<!-- BEGIN: Latest Graduate List -->
<div class="col-span-12 mt-6">
    <div class="intro-y block sm:flex items-center h-10">
        <h2 class="text-lg font-medium truncate mr-5">
            @lang('app.latest_graduate_list')
        </h2>
        <div class="dropdown xl:ml-auto mt-5 xl:mt-0">
            <select class="input border w-60" id="progdi_latest_graduate_list">
                <option value="" selected="selected">Filter By Progdi</option>
                @foreach($progdis as $progdi)
                    <option value="{{$progdi->code}}">{{$progdi-
>nama_progdi}}</option>
                @endforeach
            </select>
        </div>
    </div>
    <div class="intro-y datatable-wrapper box p-5 mt-5">
    <!-- BEGIN: Datatable -->
        <table class="table table-report table-report--bordered display
datatable w-full" id="latest_graduate_list">
        <thead>
            <tr>
                <th>@lang('app.name')</th>
                <th>@lang('app.nim')</th>
                <th>@lang('app.ipk')</th>
                <th>@lang('app.thesis_title')</th>
                <th>@lang('app.graduate_date')</th>
            </tr>
        </thead>
        </table>
    <!-- END: Datatable -->
    </div>
</div>
<!-- END: Latest Graduate List -->
```

Gambar 4.3.29. Script untuk lulusan dashboard.php

Gambar 4.41 merupakan *script* untuk mengambil data lulusan Unika Soegijapranata yang diambil dari API, respon data berupa json yang berisi nama, nim, ipk, judul skripsi serta tanggal lulus yang nantinya diolah dan di tampilkan di tabel. Gambar 4.22 merupakan *script* untuk mengolah serta menampilkan data lulusan Unika Soegijapranata yang didapatkan dari API dalam bentuk tabel.

4.3.8 Halaman *Premium Data*

Halaman *post new job* merupakan halaman khusus untuk perusahaan dimana pada halaman ini perusahaan dapat membuat lowongan program magang kerja baru, data yang perlu diisi adalah judul lowongan program magang kerja, posisi, kategori, durasi pembayaran gaji, gaji, kuota, tipe lowongan, deadline, deskripsi, manfaat, alamat serta data untuk persyaratan pelamar seperti jenis kelamin, persyaratan jurusan dan kemampuan tertentu.



Gambar 4.3.30. Halaman post new job

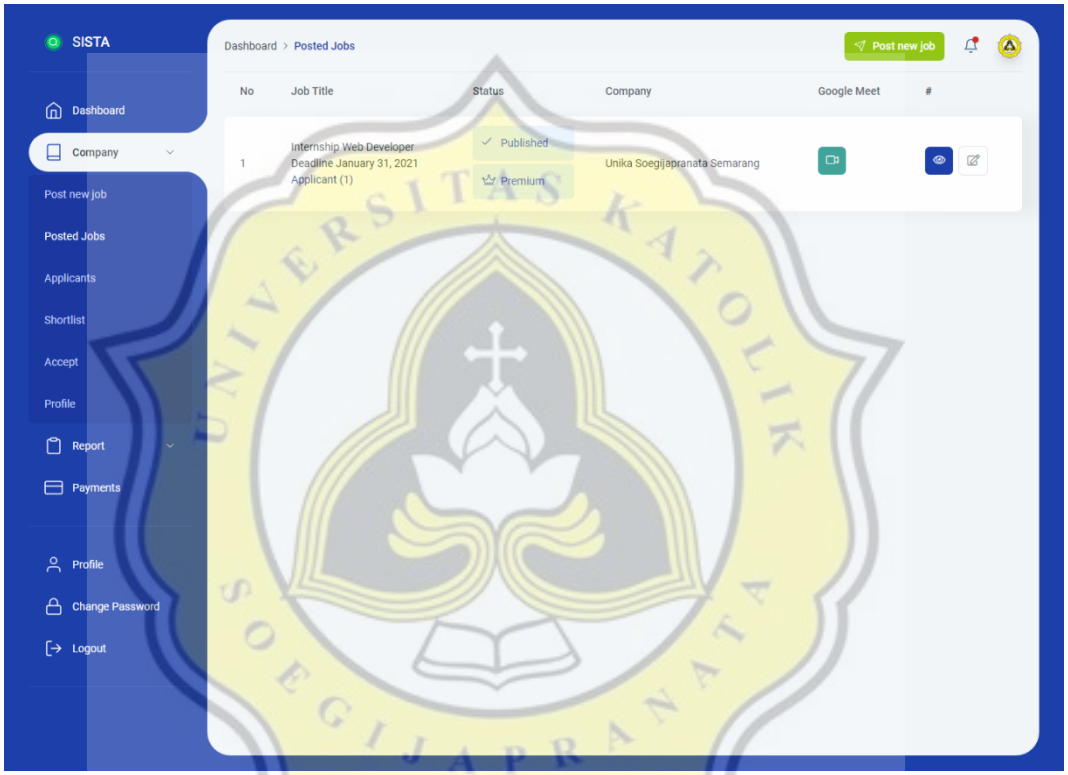
```php
public function newJobPost(Request $request){
$data = [
'user_id'                   => $user_id,
'job_title'                 => $job_title,
'job_slug'                  => $job_slug,
'position'                  => $request->position,
'category_id'               => $request->category,
'is_any_where'              => $request->is_any_where,
'salary'                    => $request->salary,
'salary_upto'               => $request->salary_upto,
'is_negotiable'             => $request->is_negotiable,
'salary_currency'           => "IDR",
'salary_cycle'              => $request->salary_cycle,
'vacancy'                   => $request->vacancy,
'gender'                    => $request->gender,
'exp_level'                 => $request->exp_level,
'job_type'                  => $request->job_type,
'job_duration'              => $request->job_duration,

'experience_required_years' => $request->experience_required_years,
'experience_plus'           => $request->experience_plus,
'description'               => $request->description,
'skills'                    => $request->skills,
'responsibilities'          => $request->responsibilities,
'educational_requirements'  => $request->educational_requirements,
'experience_requirements'   => $request->experience_requirements,
'additional_requirements'   => $request->additional_requirements,
'benefits'                  => $request->benefits,
'apply_instruction'         => "Login Using Email Student Unika Soegijapranata",
'country_id'                => "102",
'country_name'              => "Indonesia",
'state_id'                  => $request->state,
'state_name'                => $state_name,
'city_name'                 => $request->city_name,
'deadline'                  => $request->deadline,
'status'                    => 0,
'is_premium'                => 0,
'googlemeet_room'           => $room,
];
```

Gambar 4.3.31. Script untuk post new job controller

Gambar 4.44 merupakan *script* yang menjalankan fungsi untuk menyimpan data ke dalam database yang didapatkan dari perusahaan yang telah mengisi form untuk membuat lowongan program magang kerja baru.

4.3.9 Halaman *Premium Data*

Halaman *posted jobs* merupakan halaman khusus untuk perusahaan dimana pada halaman ini perusahaan dapat melihat daftar lowongan magang kerja yang telah dibuat.



Gambar 4.3.32. Halaman posted jobs

```php
public function postedJobs(){
    $title = __('app.posted_jobs');
    $user = Auth::user();
    $jobs = $user->jobs()->paginate(5);

    return view('admin.jobs', compact('title', 'jobs','user'));
}
```

Gambar 4.3.33. Script untuk list job controller

Gambar 4.46 merupakan *script* yang menjalankan fungsi untuk mengambil data lowongan pekerjaan berdasarkan perusahaan tertentu, yang kemudian akan diolah untuk ditampilkan.

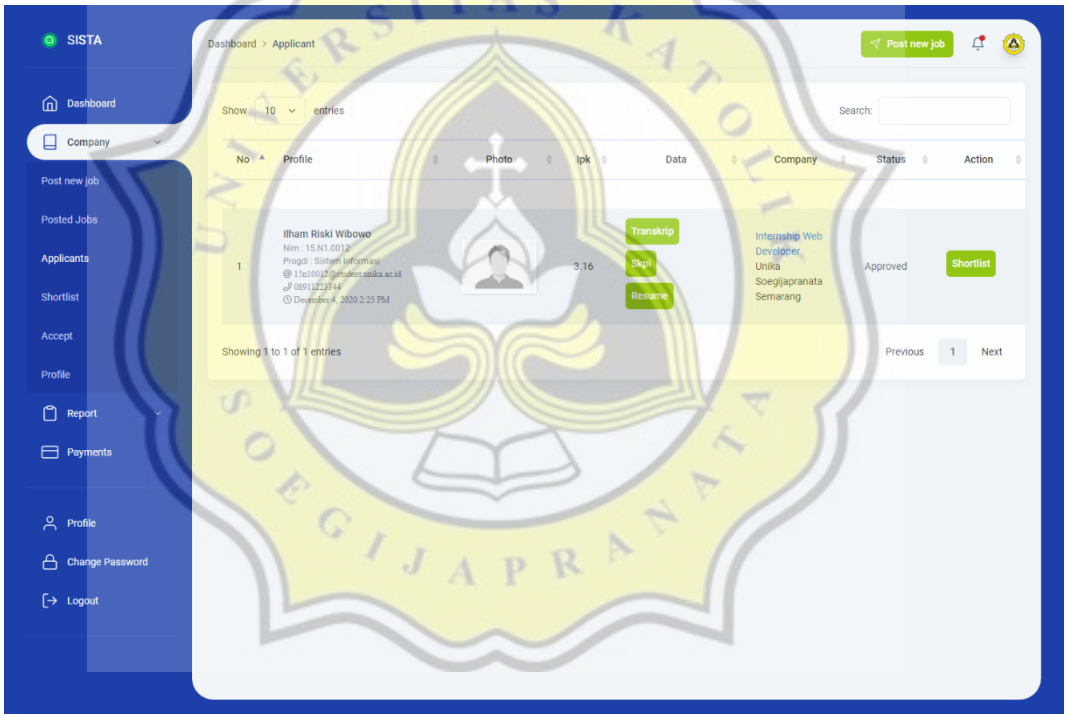
```
@foreach($jobs as $key => $job)
<tbody>
    <tr>
        <td>
            {{ $jobs->firstItem() + $key }}
        </td>
        <td>
            {{$job->job_title}}
            <p class="text-muted">@lang('app.deadline') {{$job->deadline-
>format(get_option('date_format'))}} </p>
            <p class="text-muted"> <a style="color: #1c3faa"
href="{{route('job_applicants', $job->id)}}">@lang('app.applicant') ({{$job-
>application->count()}}) </a>  </p>
        </td>
        <td>
            @if ($job->status == 0)
                <div class="rounded-md flex items-center px-3 py-3 mb-1 bg-
theme-12 text-white">
                    <i class="fal fa-hourglass-half w-5 h-5 mr-1"></i>
                    {!! $job->status_context() !!}
                </div>
            @elseif($job->status == 1)
                <div class="rounded-md flex items-center px-3 py-3 mb-1 bg-
theme-14 text-theme-10">
                    <i class="fal fa-check w-5 h-5 mr-1"></i>
                    {!! $job->status_context() !!}
                </div>
            @else
                <div class="rounded-md flex items-center px-3 py-3 mb-1 bg-
theme-6 text-white">
                    <i class="fal fa-times w-5 h-5 mr-1"></i>
                    {!! $job->status_context() !!}
                </div>
            @endif

            @if($job->is_premium)
            <div class="rounded-md flex items-center px-3 py-3 mb-1 bg-theme-14
text-theme-10">
                <i class="fal fa-crown w-5 h-5 mr-1"></i>
                @lang('app.premium')
            </div>
            @endif
        </td>
        <td>
            {{$job->employer->company}}
        </td>
        <td>
```

Gambar 4.3.34. Script untuk list posted job.php

Gambar 4.47 merupakan *script* untuk menampilkan data daftar lowongan program magang kerja yang telah dibuat oleh perusahaan terkait.

4.3.10 Halaman *Premium Data*

Halaman *applicants* merupakan halaman khusus untuk perusahaan dimana pada halaman ini perusahaan dapat melihat daftar Mahasiswa pelamar data yang diperlihatkan berupa profil Mahasiswa, photo Mahasiswa, ipk, transkrip nilai, surat keterangan pendamping ijazah, file resume, data perusahaan dan *action*. Jika tombol shortlist ditekan data akan dipindahkan ke data *shotlist*.



Gambar 4.3.35. Halaman applicants

```
public function employerApplicant(){
    $title = __('app.applicant');
    $employer_id = Auth::user()->id;
    $applications    =    JobApplication::whereEmployerId($employer_id)-
>where('is_shortlisted', '=', '0')->orderBy('id', 'desc')->paginate(5);

    return view('admin.applicants', compact('title', 'applications'));
}
```

Gambar 4.3.36. Script untuk applicant controller

Gambar diatas merupakan *script* yang berfungsi untuk mengambil data daftar pelamar dari database yang akan diolah dan ditampilkan.

```
public function transkripPdf(Request $request){
    $rules = [
        'nim'  => 'required',
    ];

    $validator = Validator::make($request->all(), $rules);

    if ($validator->fails()){
        return redirect()->back()->withInput($request->input())-
>withErrors($validator);
    }

    $url = 'http://sintak.unika.ac.id/akademik/pdf/index_capk.php';
    $terpilih = 'TRANSKRIP';
    $myvars = '&tpilihh=' . $terpilih . '&nimy=' . $request->nim;

    $ch = curl_init( $url );
    curl_setopt( $ch, CURLOPT_POST, 1);
    curl_setopt( $ch, CURLOPT_POSTFIELDS, $myvars);
    curl_setopt( $ch, CURLOPT_FOLLOWLOCATION, 1);
    curl_setopt( $ch, CURLOPT_HEADER, 0);
    curl_setopt( $ch, CURLOPT_RETURNTRANSFER, 1);

    $response = curl_exec( $ch );

    $url =
"http://sintak.unika.ac.id/akademik/pdf/qrcode_transkrip/".str_replace('
.','',$request->nim).".png";
    $contents = file_get_contents($url);
    $name = substr($url, strrpos($url, '/') + 1);
    $imageFileName = 'downloads/qrcode_transkrip/'.$name;
    Storage::disk('public')->put($imageFileName, $contents);

    $rawreturn = str_replace('img src="qrcode_transkrip/', 'img
src="https://midone1.dev/storage/downloads/qrcode_transkrip/',
$response);
    $rawreturn1 = str_replace('<a href="../index_capkidemik-
091118.php?nimidx23425a3234='.$request->nim.'" target="_self"><img
src="images/left2.gif"></a>', ' ',$rawreturn);
    $return = str_replace('img src="images/logokhs.gif"', 'img
src="https://midone1.dev/storage/downloads/qrcode_transkrip/logokhs.gif"
', $rawreturn1);
    return $return;}
```

Gambar 4.3.37. Script untuk transkrip controller

Gambar 4.50 merupakan *script* yang menjalankan fungsi untuk mengambil data transkrip Mahasiswa Unika Soegijapranata terkait dari API.

```
public function skpiPdf(Request $request){
$ch = curl_init('http://sintak.unika.ac.id/krs_skpi_mhs/index_capkid-
091118.php?nimidx23425a3234='.$request->nim);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
// get headers too with this line
curl_setopt($ch, CURLOPT_HEADER, 1);
$result = curl_exec($ch);
// get cookie
// multi-cookie variant contributed by @Combuster in comments
preg_match_all('/^Set-Cookie:s*([^;]*)/mi', $result, $matches);
$cookies = array();
foreach($matches[1] as $item) {
parse_str($item, $cookie);
$cookies = array_merge($cookies, $cookie);
}

$curl = curl_init();

curl_setopt_array($curl, array(
CURLOPT_URL =>
"http://sintak.unika.ac.id/krs_skpi_mhs/php/pdf/laporan.php",
CURLOPT_RETURNTRANSFER => true,
CURLOPT_ENCODING => "",
CURLOPT_MAXREDIRS => 10,
CURLOPT_TIMEOUT => 0,
CURLOPT_FOLLOWLOCATION => true,
CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
CURLOPT_CUSTOMREQUEST => "GET",
CURLOPT_HTTPHEADER => array(
"Cookie: PHPSESSID=".$cookies['PHPSESSID']
),
));

$response = curl_exec($curl);

header('Cache-Control: public');
header('Content-type: application/pdf');
header('Content-Disposition: attachment; filename="SKPI-'.$request-
>nim.'.pdf"');
header('Content-Length: '.strlen($response));

curl_close($curl);

return ($response);}
```

Gambar 4.3.38. Script untuk surat keterangan pendamping ijazah controller

Gambar 4.51 merupakan *script* yang menjalankan fungsi untuk mengambil data skpi Mahasiswa Unika Soegijapranata yang nantinya diolah menjadi file pdf dengan nama file SKPI-nim mahasiswa.pdf.

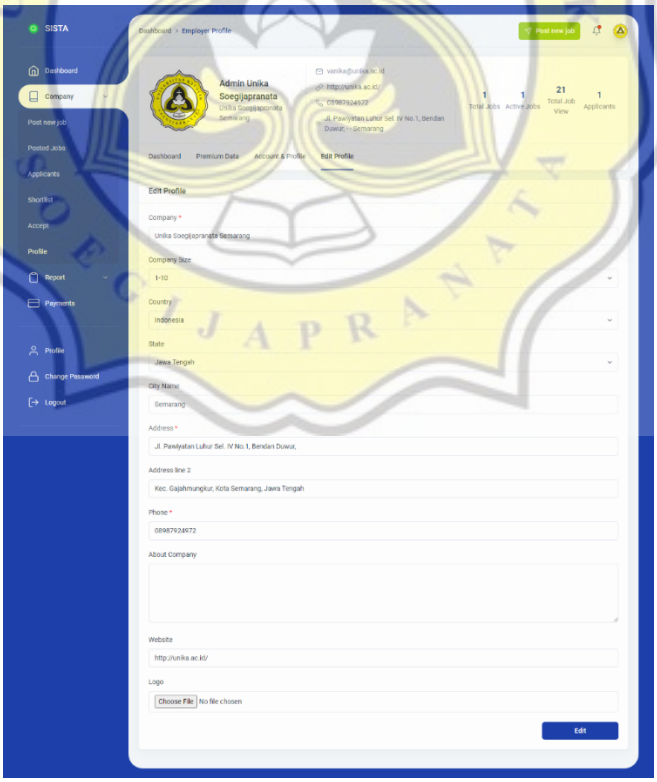
```
public function acceptList($application_id){
$applicant = JobApplication::find($application_id);
$applicant->status = 1;
$applicant->save();

Mail::to($applicant->email)->send(new AcceptMail($applicant->job_id));

return back()->with('success', __('app.accept_notification_success'));
}
```

Gambar 4.3.39. Script untuk accept button controller

Gambar 4.52 merupakan *script* yang menjalankan fungsi untuk mengganti status lamaran program magang kerja terkait, status tersebut adalah diterima, selain fungsi tersebut script ini juga akan mengirimkan pesan ke email pelamar secara otomatis.

4.3.11 Halaman *Premium Data*

Halaman *profil* perusahaan merupakan halaman khusus untuk perusahaan dimana pada halaman ini perusahaan dapat melihat dan mengubah data *profil*.



Gambar 4.3.40. Script Halaman profile perusahaan

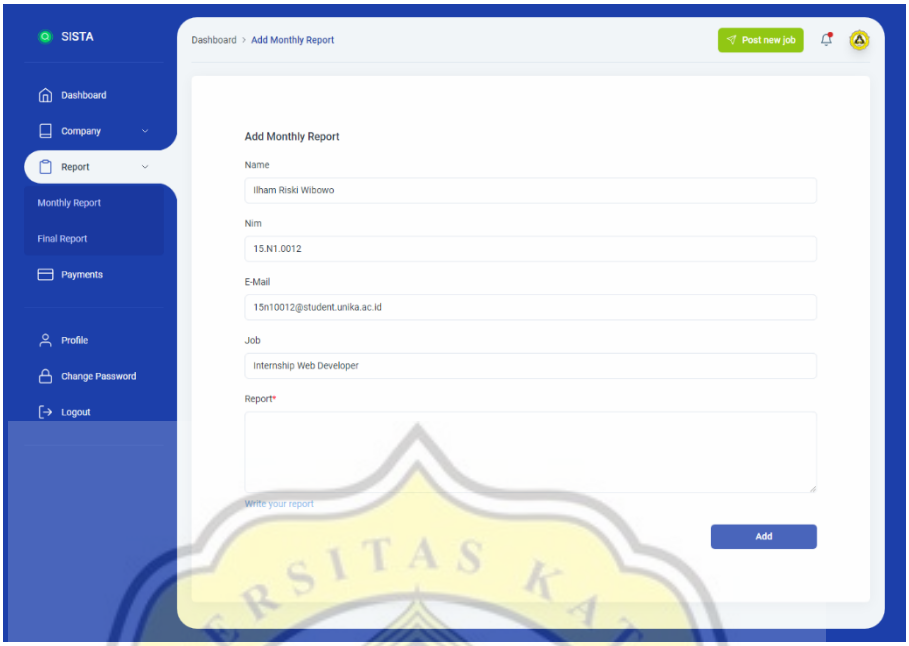
```
public function employerProfile(){
$title = __('app.employer_profile');
$user = Auth::user();

$totalJob = Job::Where('user_id', '=', $user->id)->count();
$totalJobActive = Job::Where('user_id', '=', $user->id)->active()-
>count();
$totalApplicants = JobApplication::where('employer_id', '=', $user->id)-
>count();
$totalJobView = Job::Where('user_id', '=', $user->id)->sum('views');

$countries = Country::all();
$old_country = false;
if ($user->country_id){
$old_country = Country::find($user->country_id);
}

return view('admin.employer-profile', compact('title', 'user',
'countries', 'old_country', 'totalJob', 'totalJobActive',
'totalApplicants', 'totalJobView'));}
```

Gambar 4.3.41. Script untuk profil controller

```
public function employerProfile(){
$title = __('app.employer_profile');
$user = Auth::user();

$totalJob = Job::Where('user_id', '=', $user->id)->count();
$totalJobActive = Job::Where('user_id', '=', $user->id)->active()-
>count();
$totalApplicants = JobApplication::where('employer_id', '=', $user->id)-
>count();
$totalJobView = Job::Where('user_id', '=', $user->id)->sum('views');

$countries = Country::all();
$old_country = false;
if ($user->country_id){
$old_country = Country::find($user->country_id);
}

return view('admin.employer-profile', compact('title', 'user',
'countries', 'old_country', 'totalJob', 'totalJobActive',
'totalApplicants', 'totalJobView'));
```

Gambar 4.3.42. Script untuk profil perusahaan.php

Gambar 4.54 merupakan *script* yang menjalankan fungsi untuk mengambil data profil perusahaan dari database yang nantinya akan ditampilkan pada script tampilan halaman profil pada Gambar 4.55.



Gambar 4.3.43. Halaman report

Gambar 4.56 merupakan tampilan halaman report, di halaman ini perusahaan dapat memberikan penilaian hasil magang Mahasiswa terkait.

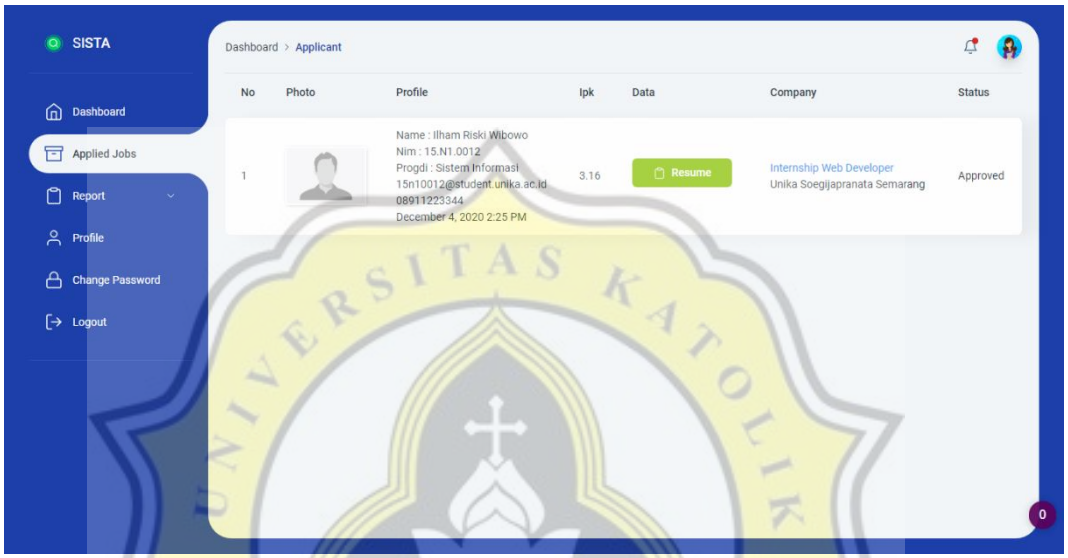
```
public function postFinalReport(Request $request, $id ){
$rules = [
'report'    => 'required',
'nilai'     => 'required',
];
$this->validate($request, $rules);
$user_id = 0;
if (Auth::check()){
$user_id = Auth::user()->id;
}
if (count($report) == $job->job_duration){
try {
$application_data = [
'faculty'                => $progdi->faculty,
'application_id'         => $request->application_id,
'employer_id'            => $user_id,
'user_id'                => $request->user_id,
'job_id'                 => $request->job_id,
'nim'                    => $request->nim,
'name'                   => $request->name,
'progdi'                 => $request->progdi,
'email'                  => $request->email,
'report'                 => $request->report,
'nilai'                  => $request->nilai,
'status'                 => '0',
];
```

Gambar 4.3.44. Script untuk add report controller

```
<div class="intro-y col-span-12 sm:col-span-6 mb-5 {{ $errors->has('nim')? 'has-
error':'' }}">
<div class="mb-2">@lang('app.nim')</div>
<input type="text" class="input w-full border flex-1
{{e_form_invalid_class('nim', $errors)}}" id="nim" value="{{ $application->nim
}}" name="nim" placeholder="@lang('app.nim')" readonly>
<p style="color: #ff0000">{!! e_form_error('nim', $errors) !!}
</div>
<div class="intro-y col-span-12 sm:col-span-6 mb-5 {{ $errors->has('email')?
'has-error':'' }}">
<div class="mb-2">@lang('app.email')</div>
<input type="text" class="input w-full border flex-1
{{e_form_invalid_class('email', $errors)}}" id="email" value="{{ $application-
>email }}" name="email" placeholder="@lang('app.email')" readonly>
<p style="color: #ff0000">{!! e_form_error('email', $errors) !!}
</div>
<div class="intro-y col-span-12 sm:col-span-6 mb-5 {{ $errors->has('job')? 'has-
error':'' }}">
<div class="mb-2">@lang('app.job')</div>
<input type="text" class="input w-full border flex-1
{{e_form_invalid_class('job', $errors)}}" id="job" value="{{ $job->job_title }}"
name="job" placeholder="@lang('app.job')" readonly>
<p style="color: #ff0000">{!! e_form_error('job', $errors) !!}
</div>
<div class="intro-y col-span-12 sm:col-span-6 mb-5 {{ $errors-
>has('job_duration')? 'has-error':'' }}">
<div class="mb-2">@lang('app.job_duration')</div>
<input type="text" class="input w-full border flex-1
{{e_form_invalid_class('job', $errors)}}" id="job_duration" value="{{ $job-
>job_duration }}" name="job_duration" placeholder="@lang('app.job_duration')"
readonly>
<p style="color: #ff0000">{!! e_form_error('job_duration', $errors) !!}
</div>
<div class="intro-y col-span-12 sm:col-span-6 mb-5 {{ $errors-
>has('report_perusahaan_bulanan')? 'has-error':'' }}">
<div class="mb-2">@lang('app.report_perusahaan_bulanan')</div>
<input type="text" class="input w-full border flex-1
{{e_form_invalid_class('job', $errors)}}" id="report_perusahaan_bulanan"
value="{{ $report->count() }}" name="report_perusahaan_bulanan"
placeholder="@lang('app.report_perusahaan_bulanan')" readonly>
<p style="color: #ff0000">{!! e_form_error('report_perusahaan_bulanan', $errors)
!!}</p>
</div>
@if($report->count() == $job->job_duration)
<div class="intro-y col-span-12 sm:col-span-6 mb-5 {{ $errors->has('report')?
'has-error':'' }}">
<div class="mb-2">@lang('app.report')<font color="red">*</font></div>
<textarea name="report" class="input w-full border flex-1
{{e_form_invalid_class('report', $errors)}}" rows="5" required>{{ old('report')
}}</textarea>
```

Gambar 4.3.45. Script untuk add report.php

Gambar 4.57 merupakan *script* yang berfungsi untuk menjalankan perintah menyimpan data, data diambil dari script tampilan *report*, script dapat dilihat pada Gambar 4.58.



Gambar 4.3.46. Halaman daftar lowongan program magang kerja yang dilamar

Gambar 4.59 merupakan halaman khusus untuk Mahasiswa, halaman ini menampilkan daftar lowongan program magang kerja yang dilamar oleh Mahasiswa, data berupa profil mahasiswa, ipk, file resume, judul lowongan program magang kerja dan perusahaan terkait, serta status lamaran. Terdapat tiga status lamaran yaitu *pending* yang berarti lamaran masih dilihat, *cancel* berarti lamaran ditolak, dan yang terakhir *approved* yang berarti lamaran diterima.

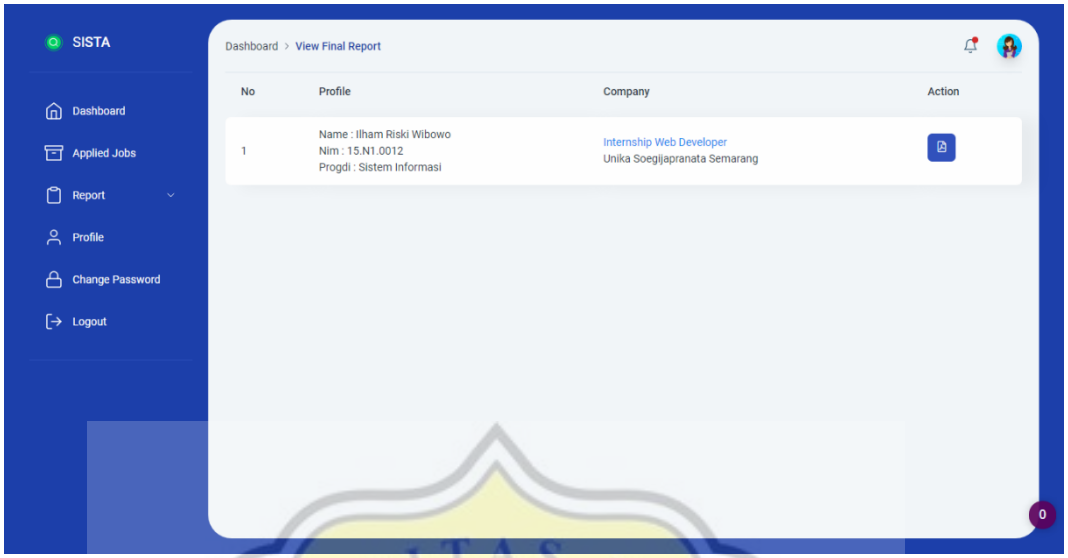
```
public function appliedJobs(){
$title = __('app.applicant');
$user_id = Auth::user()->id;
$applications = JobApplication::whereUserId($user_id)->orderBy('id', 'desc')-
>paginate(10);

return view('admin.applied_jobs', compact('title', 'applications'));
}
```

Gambar 4.61. Script untuk applied controller

```
@if( ! empty($application->job->job_title))
<p>
<a href="{{route('job_view', $application->job->job_slug)}}"
target="_blank"><font color="#1488f0">{{$application->job-
>job_title}}</font></a>
</p>
@endif

@if( ! empty($application->job->employer->company))
<p>{{$application->job->employer->company}}</p>
@endif
</td>
<td>
@if($application->is_shortlisted == 1 && $application->status == 1)
@if($application->user_status == 0)
<button class="button px-2 mr-1 mb-2 bg-theme-9 text-white">
<a href="{{route('make_accept', $application->id)}}">
@lang('app.accept')
</a>
</button>
@else
@lang('app.approved')
@endif
@else
@lang('app.waiting_for_approval')
@endif
</td>
</tr>
</tbody>
@endforeach
```

Gambar 4.3.47. Script untuk applied job.php

Gambar 4.59 merupakan *script* untuk mengambil data daftar lowongan magang kerja yang diikuti oleh Mahasiswa terkait yang nantinya akan dipanggil untuk ditampilkan pada halaman tampilan applied, script dapat dilihat pada Gambar 4.60.



Gambar 4.3.48. Halaman report untuk Mahasiswa

Unika Soegijapranata Semarang

Jl. Pawiyatan Luhur Sel. IV No.1, Bendan Duwur, Kec. Gajahmungkur, Kota Semarang, Jawa Tengah
| 08987924972 | vanika@unika.ac.id

SURAT KETERANGAN MAGANG

Kami yang mengeluarkan surat keterangan ini adalah dari perusahaan Unika Soegijapranata Semarang di Semarang, menyatakan bahwa Mahasiswa seperti yang disebutkan di bawah ini:

Nama	:	Ilham Riski Wibowo
NIM	:	15.N1.0012
Program Studi	:	Sistem Informasi

Telah melakukan magang selama 3 bulan, pada Front End Developer dengan :A+

Adapun saran yang dapat kami berikan untuk mahasiswa magang tersebut adalah sebagai berikut :Laporan akhir baik sekali

23-01-2021

Admin Unika Soegijapranata

Smart Internship System Unika Soegijapranata © 2021

Gambar 4.3.49. Hasil report.php

Gambar 4.61 merupakan tampilan halaman report untuk mahasiswa, data akan muncul apabila perusahaan telah mengisi report. Data berisi profil Mahasiswa, judul lowongan magang kerja dan perusahaan yang membuat serta file report berformat pdf.

```
public function userFinalReport(){
$title = __('app.view_final_report');
$user = Auth::user();
$user_id = 0;
if (Auth::check()){
$user_id = Auth::user()->id;
}

$reports = JobFinalReport::where('user_id', '=', $user_id)-
>where('status', '=', '0')->orderBy('id', 'desc')->paginate(5);

return view('admin.view_report_final_user', compact('title', 'reports',
'user' ));}
```

Gambar 4.3.50. Script untuk user final report controller

```
public function pdfFinalReport($id){
$application = JobApplication::find($id);
$month = JobMonthlyReport::find($id);
$final = JobFinalReport::find($id);

$company = User::find($final->employer_id);
$job = Job::find($final->job_id);

$path = storage_path('app\public\uploads\images\logos\\'.$company->logo);
$image = base64_encode(file_get_contents($path));

$user = Auth::user();

$pdf = PDF::loadView('pdf.report', compact('month', 'final', 'company', 'job',
'image'));
return $pdf->download('report_'.$final->nim.'.pdf');
}
```

Gambar 4.3.51. Script untuk report pdf controller

```
@if( ! empty($report->job->job_title))
<p>
<a href="{{route('job_view', $report->job->job_slug)}}"
target="_blank"><font color="#1488f0">{{$report->job-
>job_title}}</font></a>
</p>
@endif

@if( ! empty($report->job->employer->company))
<p>{{$report->job->employer->company}}</p>
@endif
```

Gambar 4.3.52. Script untuk Halaman report Mahasiswa.php

Gambar 4.63 merupakan *script* untuk mengambil data daftar lowongan magang kerja yang diikuti oleh Mahasiswa, sedangkan *Gambar 4*.64 merupakan *script* untuk mengolah data untuk dijadikan pdf, yang nantinya akan ditampilkan di halaman daftar report untuk Mahasiswa, *script* dapat dilihat pada *Gambar 4*.65.

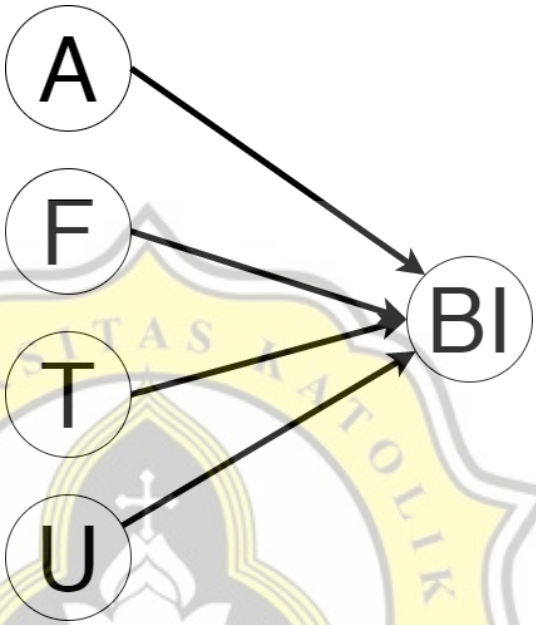
4.4 Analisa Daftar Responden

Setelah dilakukan uji coba terhadap aplikasi smart internship system Unika Soegijapranata (SISTA) berbasis PWA (Progressive Web Apps) dan melakukan pengumpulan data berupa kuesioner dari 37 orang dengan usia antara 18-24 tahun, diperoleh hasil sebagai berikut:

4.4.1 Model Pengujian

Penelitian ini menerapkan model pengujian 2 variable yaitu variable *Dependen* dan Variable *Independen.* Variable dependen adalah variable yang bergantung pada variable lainnya. Variable *dependen* yang digunakan adalah Keiginan untuk menggunakan (BI) adalah variable yang menunjukkan ketertarikan responden untuk menyarankan orang lain agar menggunakan aplikasi, Variable *Independen* adalah variable yang bebas dan tidak bergantung pada variable lain. Variable *independen* yang digunakan antara lain Atraktif (A) adalah variable yang menunjukan bahwa responden senang dengan aplikasi, Fungsi (F) adalah variable yang menunjukkan tingkat kepuasan responden terhadap fungsi aplikasi, Tampilan (T) adalah variable yang menunjukkan kepuasan responden terhadap tampilan aplikasi dan Kebergunaan (U) adalah variable yang menunjukkan presepsi kegunaan keseluruhan aplikasi. Variable *Dependen* dan Variable *Independen* akan diuji menggunakan program SPSS untuk menentukan korelasi dan hubungan antar variable. Adapun model

pengujian aplikasi smart internship system unika soegijapranata (SISTA) Berbasis PWA(progressive web apps) digunakan seperti gambar 4.66.



Gambar 4.4.1. Model variable penelitian

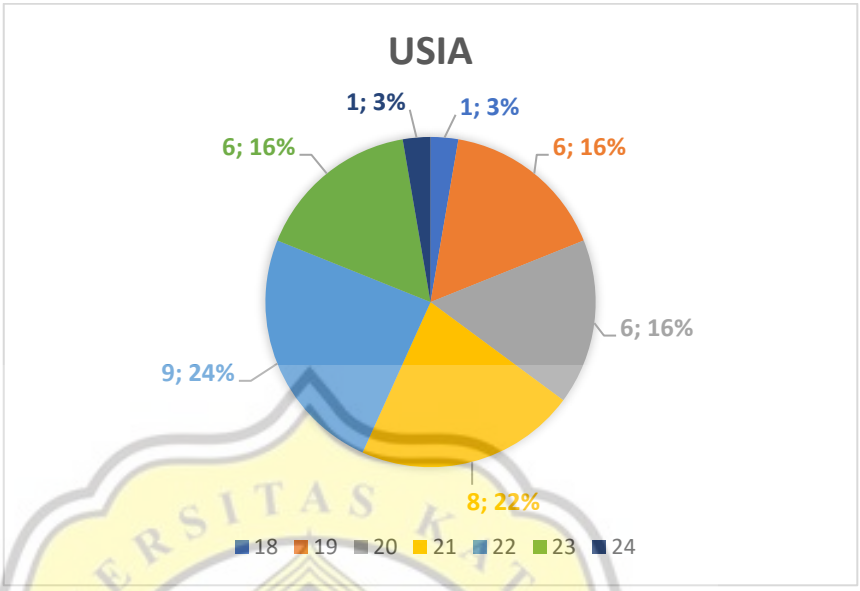
Hipotesa pengujian :

1. Atraktif (A) berpengaruh kuat terhadap Keiginan untuk menggunakan (BI)
2. Fungsi (F) berpengaruh kuat terhadap Keiginan untuk menggunakan (BI)
3. Tampilan (T) berpengaruh kuat terhadap Keiginan untuk menggunakan (BI)
4. Kebergunaan (U) berpengaruh kuat terhadap Keiginan untuk menggunakan (BI)

4.4.2 Profil Responden

Dari 37 responden yang mengikuti uji coba aplikasi smart internship system Unika Soegijapranata (SISTA) Berbasis PWA(*progressive web apps*), dapat diketahui jumlah responden berdasarkan umur, jenis kelamin, dan pengalaman sesuai Gambar dibawah ini.

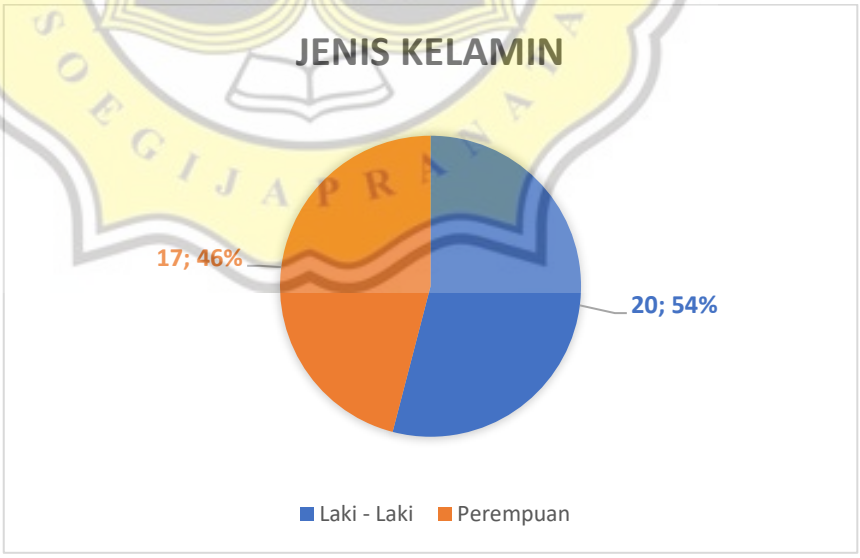
a. Usia



Gambar 4.4.2. Diagram usia responden

Responden terbanyak adalah responden yang berusia antara 21 – 22 tahun dengan jumlah 17 responden.

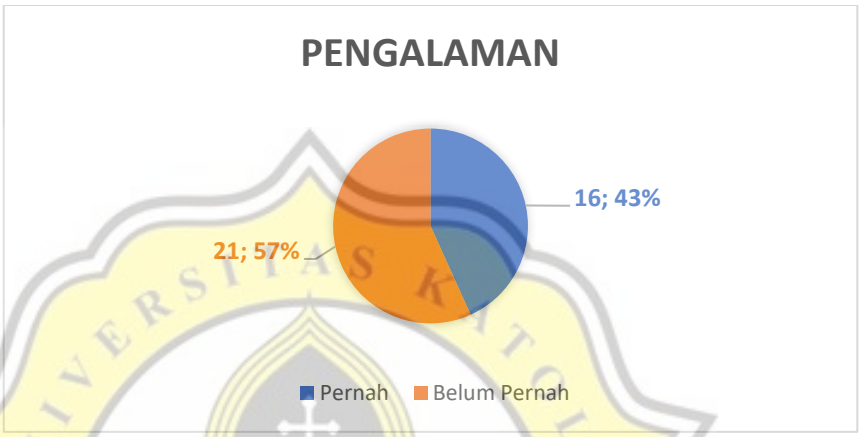
b. Jenis kelamin



Gambar 4.4.3. Diagram jenis kelamin responden

Dari total 37 responden, responden terbanyak merupakan laki – laki dengan jumlah 20 orang sedangkan perempuan berjumlah 17 orang.

c. Pernah mengikuti program megang kerja



Gambar 4.4.4. Diagram pengalaman mengikuti program magang kerja

Data diatas menunjukan bahwa lebih banyak responden yang memiliki pengalaman dalam mengikuti program magang kerja mayoritas belum pernah dengan jumlah responden 21 sedangkan yang sudah memiliki pengalaman berjumlah 16 responden.

4.4.3 Hasil Uji Statistika

Tabel 4.4.1. Tabel Validitas 1

Rotated Component Matrix[a]

	Component		
	1	2	3
Tampilan 1	.461	.718	.263
Tampilan 2	.545	.566	.224
Tampilan 3	.229	.898	-.060
Tampilan 4	.300	.754	.263
Atraktif 1	.812	.333	.227
Atraktif 2	.638	.464	.206
Atraktif 3	.857	.170	.356
Atraktif 4	.866	.181	.287
BI 1	.075	.841	.073
BI 3	.499	.703	.248
BI 2	-.007	.892	.293
Kegunaan 1	.631	.467	.298
Kegunaan 2	.672	.525	.259
Kegunaan 3	.693	.511	.258
Kegunaan 4	.681	.252	.281
Fungsi 1	.576	.595	.336
Fungsi 2	.368	.748	.279
Fungsi 3	.596	.597	.314
Fungsi 4	.334	.809	.248

Extraction Method: Principal Component Analysis.

Rotation Method: Equamax with Kaiser Normalization.[a]

a. Rotation converged in 7 iterations.

Untuk mendapatkan hasil uji korelasi, beberapa tahapan yang harus dikerjakan menggunakan aplikasi *Statistical Package for the Social Sciences* (SPSS). Tahapan awal adalah uji validitas data kuesioner. Uji validitas adalah pengujian untuk mengetahui tingkat keandalan dan kesahihan variabel [12].

Berdasarkan tabel 4.7, hasil kuesioner untuk uji validitas dari 5 variabel Atraktif (A), Fungsi (F), Keinginan untuk menggunakan (BI), Tampilan (T) dan Kebergunaan (U). Didapatkan uji validitas dimana korelasi Keiginan untuk menggunakan 1 (BI1) dan Tampilan 3 (T3) sangat rendah, sedangkan korelasi tertinggi ada pada Tampilan 3 (T3) dan Keinginan untuk menggunakan 2 (BI2) dimana masing – masing memiliki point .898 dan .892.

Tabel 4.4.2. Tabel reliabilitas

Variabel	**Cronbach's Alpha Based on Standardized Items**	**Keterangan**
Keiginan untuk menggunakan (BI)	.638	Questionable
Kebergunaan(U)	.875	Good
Tampilan(T)	.851	Good
Fungsi(F)	.928	Excellent
Atraktif(A)	.858	Good

Uji reliabilitas variabel BI, U, T, F dan A, untuk menentukan pada tabel 8 didasarkan pada, George and Mallery (2005) George and Mallery (2003) *rules of thumb*: "_ > .9 – *Excellent*, _ > .8 – *Good*, _ > .7 – Acceptable, _ > .6 – *Questionable*, _ > .5 – *Poor*, and _ < .5 – *Unacceptable*" [13]. Uji reliabilitas adalah alat ukur yang digunakan dalam penelitian untuk mengetahui konsistensi hasil uji atau kuesioner

[12]. Pada hasil kuesioner untuk uji reliabilitas untuk variabel BI dinyatakan *Questionable* karena nilai *Cronbach's Alpha* menunjukan angka .638, nilai Cronbach's Alpha didapat dengan cara menguji variabel sejenis menggunakan menu *Reliability Analyze* pada SPSS. Uji reliabilitas untuk variabel U, T dan A dinyatakan *Good* dengan nilai *Cronbach's Alpha* masing – masing .875, .851, dan .858. Hasil uji reliabilitas pada variabel F dinyatakan *Excellent* dengan nilai *Cronbach's Alpha* .928.

Tabel 4.4.3. Table uji korelasi

Correlations

		AA	BII	UU	FF	TT	
AA	Pearson Correlation	1	.799**	.803**	.837**	.862**	.
BII	Pearson Correlation	.799**	1	.814**	.770**	.865**	.
UU	Pearson Correlation	.803**	.814**	1	.923**	.854**	.
FF	Pearson Correlation	.837**	.770**	.923**	1	.807**	.
TT	Pearson Correlation	.862**	.865**	.854**	.807**	1	

**. Correlation is significant at the 0.01 level (2-tailed).

Uji korelasi adalah uji Statistik untuk mengetahui hubungan antar dua variabel [14]. Untuk dapat mengetahui hasil uji korelasi, langkah yang perlu dikerjakan adalah mencari nilai rata-rata dari variabel yang sama. Berdasarkan tabel 9 setiap variabel memiliki korelasi pada variabel lainnya,

yang ditandai dengan adanya bintang (*). Untuk mendapatkan hasil ini menggunakan menu *Bivariate Correlation* pada aplikasi SPSS.

Berdasarkan uji korelasi saat pelaksanaan kuesioner kepada 37 responden, didapat bahwa pada aspek Atraktif, Menarik, Tampilan, Kegunaan, dan Fungsi memiliki kaitan satu sama lainnya dengan nilai diatas .600. Dapat diartikan bahwa aplikasi memenuhi Smart Internship System Unika Soegijapranata (SISTA) Berbasis PWA (Progressive Web Apps), aspek – aspek yang diinginkan oleh pengguna.