

LAPORAN MINI PROJECT DATABASE
PENCARIAN KERJA



Kelompok 4

Anggota :

Azarine Aisyah Ramadhani (2206051550)

Juanita Jovana (2206028604)

Maryesta Apriliani S. (2206051531)

Nadira Eka Rahmaha (2206051525)

Yiesha Reyhani Ghozali (2206828115)

DAFTAR ISI

DAFTAR ISI.....	2
BAB 1 PENDAHULUAN.....	3
1.1 Latar Belakang Masalah.....	3
1.2 Tujuan.....	3
BAB 2 DATABASE.....	4
2.1 Rancangan Database.....	4
2.2 ER Diagram.....	7
2.3 Tabel Relasional.....	7
BAB 3 IMPLEMENTASI.....	11
3.1 Implementasi SQL.....	11
3.2 Implementasi GUI.....	17
3.2.1 Package yang Digunakan.....	17
3.2.2 Fitur Login.....	18
3.2.3 Fitur Register.....	20
3.2.3 Menu Utama.....	22
3.2.3 Fitur Jobseeker.....	23
3.2.3 Fitur Company.....	27
3.3 Testing.....	29
3.3.1 Fitur Login.....	30
3.3.2 Fitur Registrasi.....	30
3.3.3 Menu Utama.....	32
3.3.4 Fitur Jobseeker.....	32
3.3.5 Fitur Company.....	35
BAB 4 PENUTUP.....	38
DAFTAR PUSTAKA.....	39
LAMPIRAN.....	40

BAB 1 PENDAHULUAN

1.1 Latar Belakang Masalah

Proses rekrutmen merupakan proses yang rumit, menghabiskan banyak waktu, uang, dan juga sumber daya bagi perusahaan untuk mencari dan menyeleksi calon karyawan. Oleh karena itu, diperlukan metode pengelolaan yang efisien untuk mengatur proses perekrutan pegawai. Perkembangan teknologi yang ada membuat perusahaan mulai melakukan perubahan dalam proses perekrutan karyawan. Perusahaan mulai beralih dari prosedur manual ke prosedur digital dengan menggunakan bantuan situs atau aplikasi lowongan kerja dalam mencari dan menyeleksi calon karyawannya. Melalui situs tersebut, pelamar dan perusahaan sama-sama dimudahkan karena dapat mendaftar dan menyeleksi calon karyawannya dengan waktu yang lebih fleksibel dan singkat, serta tidak perlu menyiapkan atau menyimpan dokumen secara manual.

Perubahan prosedur tersebut memerlukan perancangan basis data terintegrasi dan mencakup semua informasi yang diperlukan dalam proses pencarian kerja sehingga proses pencarian kerja dapat berjalan dengan lebih efisien

1.2 Tujuan

1. Memfasilitasi para *jobseeker* dengan fitur pencarian kerja yang efisien dan dilengkapi filter yang dapat disesuaikan sesuai kebutuhan.
2. Membantu perusahaan untuk mempromosikan pekerjaan mereka secara efektif kepada calon karyawan potensial yang sesuai kriteria.
3. Mengelola informasi tentang deskripsi pekerjaan, gaji, dan penawaran lainnya oleh perusahaan kepada para *jobseeker*.
4. Membantu jalannya proses wawancara dan alur seleksi dengan menyediakan informasi yang lengkap dan terstruktur berkaitan dengan profil *jobseeker* dan pekerjaan yang ditawarkan perusahaan.

BAB 2 DATABASE

2.1 Rancangan Database

Entitas:

1. Company: Untuk menyimpan informasi tentang perusahaan yang memposting pekerjaan
2. Jobseeker: Untuk menyimpan informasi dan data diri tentang pencari kerja
3. Job: Untuk menyimpan informasi mengenai pekerjaan yang ditawarkan oleh perusahaan
4. Resume: Untuk menyimpan informasi resume pelamar kerja
5. Interview: Untuk menyimpan tanggal dan waktu untuk melakukan interview
6. Result: Untuk menyimpan informasi mengenai hasil seleksi oleh perusahaan

Entitas	Atribut	Key	Domain
Company	company_id	Primary Key	Kumpulan digit angka yang merepresentasikan id perusahaan {1, 2, ...}.
	name_com		Kumpulan karakter string yang merepresentasikan nama perusahaan.
	location		Kumpulan karakter string yang merepresentasikan lokasi perusahaan.
	email_com		Kumpulan karakter yang merepresentasikan email perusahaan.
	password_com		Kumpulan karakter yang merepresentasikan sandi akun perusahaan.
Jobseeker	jobseeker_id	Primary Key	Kumpulan digit angka yang merepresentasikan id pencari kerja {1, 2, ...}.
	name_js		Kumpulan karakter string yang merepresentasikan nama pencari kerja.
	phone_number		Kumpulan 10 hingga 12 digit angka yang merepresentasikan nomor telepon pencari kerja.
	address		Kumpulan karakter string yang merepresentasikan alamat tempat tinggal pencari kerja.
	email_js		Kumpulan karakter yang merepresentasikan

			email pelamar kerja
	password_js		Kumpulan karakter yang merepresentasikan sandi akun pencari kerja.
	education		Kumpulan karakter yang merepresentasikan pendidikan terakhir pencari kerja, misalnya Sarjana (<i>Bachelor's Degree</i>).
	birth_date		Kumpulan tanggal dengan format DD/MM/YYYY yang merepresentasikan tanggal lahir pelamar
Job	job_id	Primary Key	Kumpulan digit angka yang merepresentasikan id pekerjaan {1, 2, ...}.
	title		Kumpulan karakter string yang merepresentasikan nama pekerjaan.
	date		Kumpulan tanggal lamaran pekerjaan dibuka dengan format DD/MM/YYYY
	type		Kumpulan karakter string yang merepresentasikan tipe pekerjaan {full time, part time,...}
	description		Kumpulan karakter yang merepresentasikan deskripsi pekerjaan.
	requirement		Kumpulan karakter yang merepresentasikan tanggung jawab pelamar setelah diterima
	qualification		Kumpulan karakter yang merepresentasikan kualifikasi pekerjaan.
	salary		Kumpulan karakter yang merepresentasikan range gaji pekerjaan.
	company_id	Foreign Key	Kumpulan digit angka yang merepresentasikan id perusahaan {1, 2, ...}.
	deadline		Kumpulan karakter yang merepresentasikan waktu terakhir melamar kerja dengan format DD/MM/YYYY.
Resume	resume_id	Primary Key	Kumpulan digit angka yang merepresentasikan id resume pelamar kerja {1, 2, ...}.
	link_file		Kumpulan karakter yang merepresentasikan link resume pelamar kerja

	jobseeker_id	Foreign Key	Kumpulan digit angka yang merepresentasikan id pencari kerja {1, 2, ...}.
Interview	interview_id	Primary Key	Kumpulan digit angka yang merepresentasikan id wawancara kerja.
	job_id	Foreign Key	Kumpulan digit angka yang merepresentasikan id pekerjaan {1, 2, ...}.
	date		Kumpulan tanggal dengan format DD/MM/YYYY yang merepresentasikan tanggal dilakukannya wawancara kerja.
	time		Kumpulan karakter yang merepresentasikan waktu dimulainya wawancara kerja. Contoh: {09.00 WIB, 14.00 WIB, ...}
	jobseeker_id	Foreign Key	Kumpulan digit angka yang merepresentasikan id pencari kerja {1, 2, ...}.
Result	result_id	Primary Key	Kumpulan digit angka yang merepresentasikan id hasil lamaran {1, 2, ...}.
	jobseeker_id	Foreign Key	Kumpulan digit angka yang merepresentasikan id pencari kerja {1, 2, ...}.
	job_id	Foreign Key	Kumpulan digit angka yang merepresentasikan id pekerjaan {1, 2, ...}.
	status		Kumpulan karakter string yang merepresentasikan status diterima/tidaknya pelamar kerja.

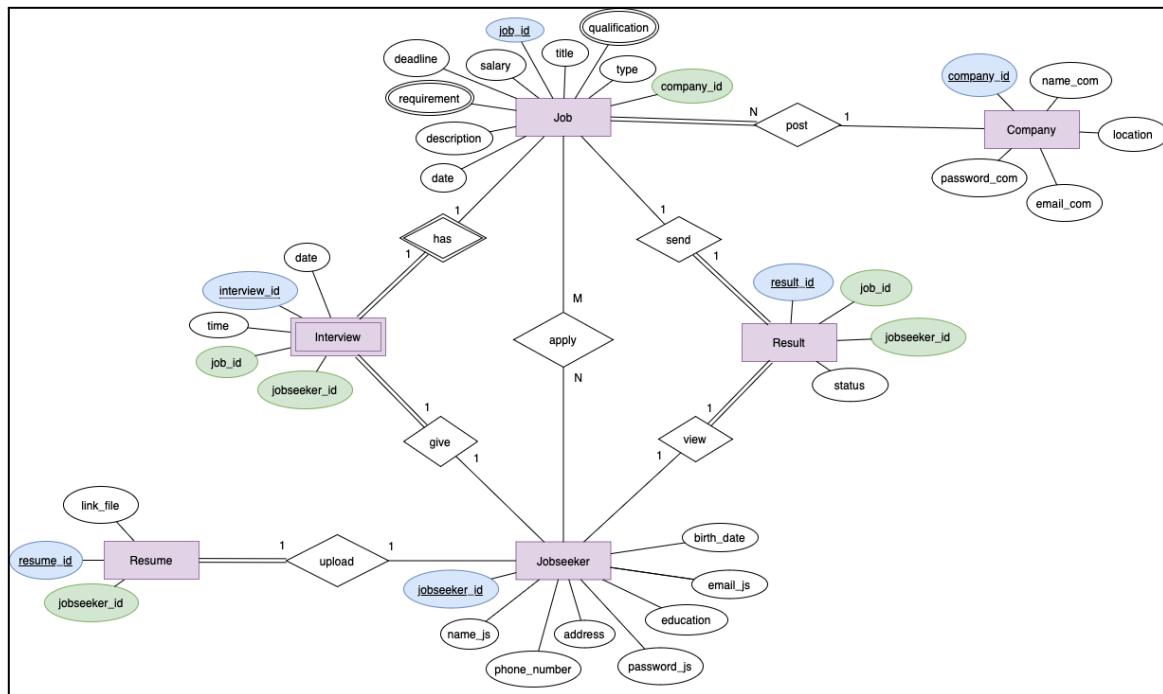
Rancangan Sistem:

- Suatu *company* dapat mengunggah lebih dari satu pekerjaan ke dalam sistem
- Suatu *job* dapat memiliki beberapa *qualification* dan *requirement*
- *Jobseeker* dapat melamar ke beberapa *job* sekaligus, begitu pula *job* dapat memiliki beberapa *jobseeker*
- Suatu *job* mengirimkan *result* kepada masing-masing *jobseeker* sesuai dengan diterima atau tidaknya mereka, hanya terdapat satu hasil bagi setiap *jobseeker*
- Suatu *job* mempunyai jadwal *interview* khusus bagi masing-masing *jobseeker*, satu *jobseeker* hanya memiliki satu jadwal *interview* pada sebuah *job*
- *Jobseeker* memberikan *interview* sesuai jadwalnya dan dapat melihat *result* apakah mereka diterima atau tidak

- *Jobseeker* dapat mengunggah *resume* mereka, masing-masing *jobseeker* hanya diperbolehkan mengunggah satu *resume*

2.2 ER Diagram

Berdasarkan rancangan database yang telah disampaikan pada bab sebelumnya, maka ER Diagram yang dihasilkan adalah sebagai berikut



Keterangan

Warna ungu = Entitas

Warna biru = Primary Key

Warna hijau = Foreign Key

2.3 Tabel Relasional

Tabel 1NF

TABEL JOB

job_id	title	date	type	description	requirement	qualification	deadline	salary	company_id
1	Receptionist/ Admin	30-09-2018	Part-time or Full	The incumbent will work under	Answer telephone calls	Strong analytical and ar	05-11-2018	168001-258	1
1	Receptionist/ Admin	30-09-2018	Part-time or Full	The incumbent will work under	Provide interested pa	Strong analytical and ar	05-11-2018	168001-258	1
1	Receptionist/ Admin	30-09-2018	Part-time or Full	The incumbent will work under	Greet visitors and gui	Strong analytical and ar	05-11-2018	168001-258	1

TABEL COMPANY

company_id	name_com	location	email_com	password_com
1	Career Center N	532 Autumn Brook Apt	recruitment@careercenter.com	aVPoXzCZ
2	EPAM Systems,	999 Natasha Haven	recruitment@epam.com	Xna1bFUP
3	MLN Pharm Ltd	114 Hines Ville Apt. 2	recruitment@mlnpharm.com	y5e1GLW5

TABEL JOBSEEKER

jobseeker_id	name_js	birth_date	phone_number	email_js	password_js	address	education
1	Alan Tran	06/01/1990	(299)375-1746	hgonzalez@zmoxD1twg	9118 Rodriguez	PhD	
2	Alexander Edw	13/05/1973	(966)697-9845	dwilson@ei7Jb5Hsc	826 Chavez Cl	High School	
3	Alyssa Rivers	20/03/1975	(320)788-5710	williamsdariLyKeVYKR	935 Johnny Pa	High School	

TABEL INTERVIEW

interview_num	job_id	date	time	jobseeker_id
1	3	11/12/2018	11:30	1
2	11	26/12/2018	13:30	10
3	6	25/09/2018	12:30	102

TABEL RESULT

result_id	jobseeker_id	job_id	status
1	82	1	Not Accepted
2	80	1	Not Accepted
3	53	10	Not Accepted

TABEL RESUME

resume_id	link_file	jobseeker_id
1	https://drive.google.com/uC	1
2	https://drive.google.com/uC	2
3	https://drive.google.com/uC	3

Sesuai dengan ER diagram, tabel relasional kami terdiri dari 6 tabel dari tiap entitas. Pada proses pembuatan tabel ini, kami melakukan perubahan pada kolom ‘requirement’ dan ‘qualification’ dari tabel ‘job’ karena kedua kolom tersebut terdiri lebih dari satu nilai (*multivalue*).

Tabel 2NF

TABEL JOB									
PK	job_id	title	date	type	description	deadline	salary	company_id	FK
	1	Receptionist/ Administr	30-09-2018	Part-time	The incumbent will work und	05-11-2018	168001-258000 A	1	
	2	QA Engineer	02-11-2017	Full time	EPAM Systems, Inc. is seeking	09-12-2017	348001-378000 A	2	

PK	TABEL QUALIFICATION			PK	TABEL REQUIREMENT		
qualif_id	qualification			req_id	requirement		
1	Excellent communication skills;			1	Answer telephone calls and inquiries,		
2	Good oral and written communicatio			2	Provide interested parties/ visitors wi		

PK	TABEL BRIDGE JOB-QUALIFICATION			PK	TABEL BRIDGE JOB-REQUIREMENT			PK	TABEL BRIDGE JOB-JOBSEEKER		
FK	job_id	qualif_id	FK	FK	job_id	req_id	FK	FK	job_id	jobseeker_id	FK
	1	1			1	1			1	82	
	1	2			1	2			1	80	

PK	TABEL COMPANY				
company_id	name_com	location	email_com	password_com	
1	Career Center	532 Autumn Brook A	recruitment@ca	lvPx0zCZ	
2	EPAM Systems	999 Natasha Haven	recruitment@ep	Xna1bFUP	

PK	FK	TABEL INTERVIEW				FK
interview_id	job_id	date	time	jobseeker_id		
1	1	11/12/2018	11:30	82		
2	1	26/12/2018	13:30	80		

PK	TABEL JOBSEEKER						
jobseeker_id	name_js	birth_date	phone_number	email_js	password_js	address	education
1	Alan Tran	06/01/1990	(299)375-1746	hgonzalez@example.net	zmnoxD1wg	9118 Rodriguez	PhD
2	Alexander Edwards	13/05/1973	(965)697-9845	dwilson@example.org	7Jb5Hksc	826 Chavez Circ	High School

PK	TABEL RESUME			PK	TABEL RESULT			PK
resume_id	link_file	jobseeker_id		result_id	jobseeker_id	job_id	status	
1	https://drive.google.com/uc?export=download&id=1	1		1	82	1	Not Accepted	
2	https://drive.google.com/uc?export=download&id=2	2		1	80	1	Not Accepted	

Tabel 2NF kami sudah sesuai dengan bentuk normal kedua, yakni memenuhi bentuk normal pertama serta atribut yang bukan *key* harus bergantung secara fungsi pada *primary key*. Pada bentuk normal kedua, ketergantungan parsial atribut bukan *key* terhadap *primary key* harus dihilangkan.

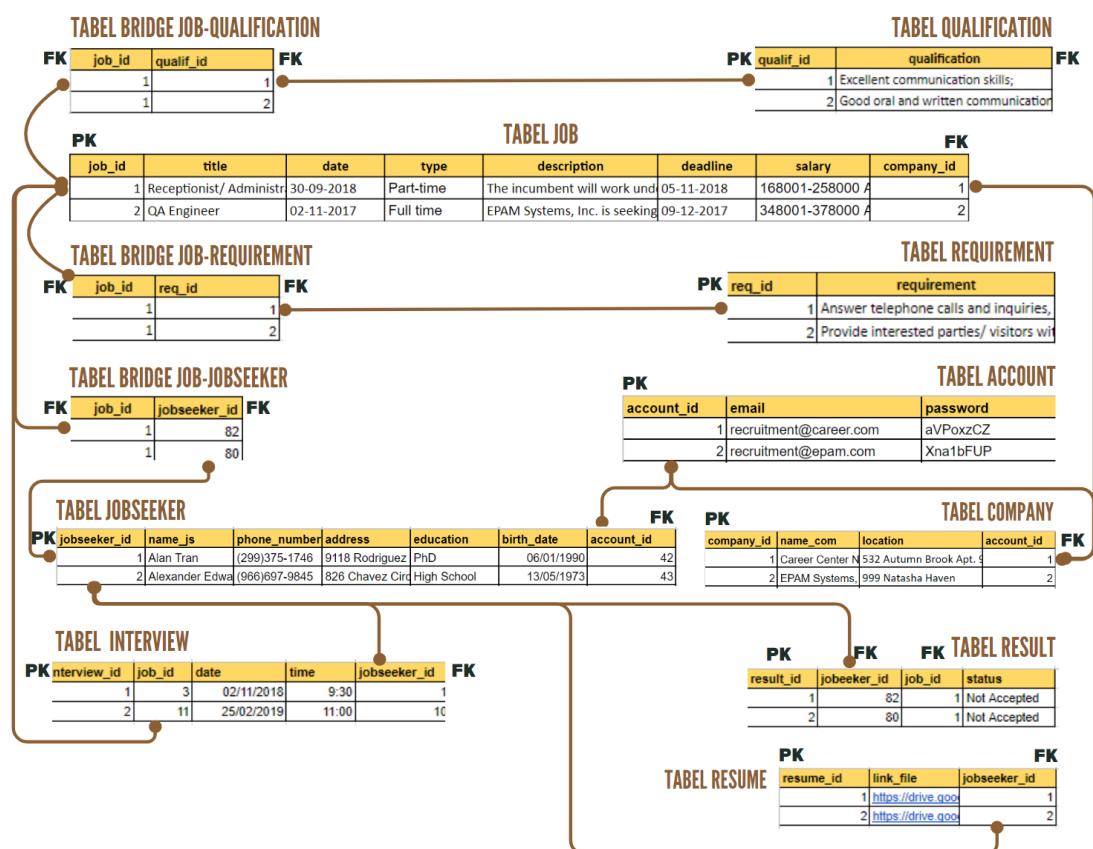
Kami mendekomposisi tabel dan menghasilkan beberapa tabel baru untuk mengatasi ketergantungan parsial yang ada. Tabel tersebut adalah tabel ‘qualification’, tabel ‘requirement’, tabel ‘bridge job-qualification’, tabel ‘bridge job-requirement’, dan tabel ‘bridge job-jobseeker’. Hal tersebut dilakukan karena terdapat hubungan *many to many* dari kolom ‘qualification’ dan kolom ‘requirement’ sehingga ditemukan ketergantungan parsial di antara keduanya. Masing-masing dari kolom ‘qualification’ dan kolom ‘requirement’ membentuk tabel baru beserta *primary key* di masing-masing tabel, yakni ‘qualif_id’ untuk

mengidentifikasi kualifikasi *jobseeker* dan ‘req_id’ untuk mengidentifikasi persyaratan *jobseeker*.

Selain itu, terdapat juga tabel *bridge*. Masih berkaitan dengan sebelumnya terkait mengatasi ketergantungan parsial yang ada, tabel ‘bridge job-qualification’ dibuat untuk menghubungkan antara tabel job dengan tabel qualification dan tabel ‘bridge job-requirement’ dibuat untuk menghubungkan antara tabel job dengan tabel requirement. Sementara itu, tabel ‘bridge job-jobseeker’ dibuat untuk menghubungkan antara tabel job dengan tabel jobseeker. Oleh karena itu, pada tabel bridge hanya berisi *foreign key*.

Tabel 3NF

Tabel 2NF kami pada bagian sebelumnya sudah memenuhi bentuk normal ketiga, yakni atribut yang bukan *primary key* tidak memiliki ketergantungan transitif pada *primary key*. Namun pada tabel 3NF dibawah, kami mengintegrasikan kolom ‘email’ dan ‘password’ dari tabel ‘company’ dan ‘jobseeker’ menjadi satu tabel baru, yaitu tabel ‘account’ yang terdiri dari kolom ‘account_id’ sebagai *primary key* serta kolom ‘email’ dan ‘password’. Hal tersebut dilakukan untuk memudahkan implementasi GUI. Secara keseluruhan terdapat 12 tabel, 3 diantaranya merupakan tabel bridge yang menghubungkan antara 3 *primary key*.



BAB 3 IMPLEMENTASI

3.1 Implementasi SQL

CREATE TABLE

Membuat tabel baru pada database sesuai dengan tabel relasional 3NF

Tabel Qualification

The screenshot shows the 'qualif' table configuration in MySQL Workbench. It has two fields: 'qualification' (TEXT type) and 'qualif_id' (INTEGER type). Primary key constraints are applied to both fields. Below the table definition, the generated SQL code is:

```
1 CREATE TABLE `qualif` (
2     `qualification` TEXT NOT NULL,
3     `qualif_id` INTEGER NOT NULL UNIQUE,
4     PRIMARY KEY(`qualif_id` AUTOINCREMENT)
5 );
```

Tabel Account

The screenshot shows the 'account' table configuration in MySQL Workbench. It has three fields: 'account_id' (INTEGER type), 'email' (TEXT type), and 'password' (TEXT type). Primary key constraints are applied to the 'account_id' field. Below the table definition, the generated SQL code is:

```
1 CREATE TABLE `account` (
2     `account_id` INTEGER NOT NULL UNIQUE,
3     `email` TEXT NOT NULL UNIQUE,
4     `password` TEXT NOT NULL,
5     PRIMARY KEY(`account_id` AUTOINCREMENT)
6 );
```

Tabel Bridge Job-Jobseeker

The screenshot shows the 'br_apply' table configuration in MySQL Workbench. It has two fields: 'job_id' and 'jobseeker_id'. Foreign key constraints link 'job_id' to the 'job' table and 'jobseeker_id' to the 'jobseeker' table. Below the table definition, the generated SQL code is:

```
1 CREATE TABLE `br_apply` (
2     `job_id` INTEGER,
3     `jobseeker_id` INTEGER,
4     FOREIGN KEY(`jobseeker_id`) REFERENCES `jobseeker`(`jobseeker_id`),
5     FOREIGN KEY(`job_id`) REFERENCES `job`(`job_id`)
6 );
```

Tabel Bridge Job-Qualification

The screenshot shows the 'br_qualif' table configuration in MySQL Workbench. It has two fields: 'job_id' and 'qualif_id'. Foreign key constraints link 'job_id' to the 'job' table and 'qualif_id' to the 'qualif' table. Below the table definition, the generated SQL code is:

```
1 CREATE TABLE `br_qualif` (
2     `job_id` INTEGER NOT NULL,
3     `qualif_id` INTEGER NOT NULL,
4     FOREIGN KEY(`qualif_id`) REFERENCES `qualif`(`qualif_id`),
5     FOREIGN KEY(`job_id`) REFERENCES `job`(`job_id`)
6 );
```

Tabel Bridge Job-Requirement

Table

br_requir

Advanced

Fields Constraints

Name	Type	NN	PK	AI	U	Default	Check	Collation	Foreign Key
req_id	INTEGER	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/> "requir"("req_id")
job_id	INTEGER	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/> "job"("job_id")

```

1 CREATE TABLE `br_requir`(
2   `req_id` INTEGER NOT NULL,
3   `job_id` INTEGER NOT NULL,
4   FOREIGN KEY(`job_id`) REFERENCES `job`(`job_id`),
5   FOREIGN KEY(`req_id`) REFERENCES `requir`(`req_id`)
6 );

```

Tabel Company

Table

company

Advanced

Fields Constraints

Name	Type	NN	PK	AI	U	Default	Check	Collation	Foreign Key
company_id	INTEGER	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>				
name_com	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>
location	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>
account_id	INTEGER	<input type="checkbox"/>			<input checked="" type="checkbox"/> "account"("account_id")				

```

1 CREATE TABLE `company`(
2   `company_id` INTEGER NOT NULL UNIQUE,
3   `name_com` TEXT NOT NULL,
4   `location` TEXT NOT NULL,
5   `account_id` INTEGER,
6   PRIMARY KEY(`company_id` AUTOINCREMENT),
7   FOREIGN KEY(`account_id`) REFERENCES `account`(`account_id`)
8 );

```

Tabel Job

Table

job

Advanced

Fields Constraints

Name	Type	NN	PK	AI	U	Default	Check	Collation	Foreign Key
job_id	INTEGER	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>				
title	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>
date	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>
type	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>
description	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>
deadline	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>
salary	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>
company_id	INTEGER	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/> "company"("company_id")

```

1 CREATE TABLE `job`(
2   `job_id` INTEGER NOT NULL UNIQUE,
3   `title` TEXT NOT NULL,
4   `date` TEXT NOT NULL,
5   `type` TEXT NOT NULL,
6   `description` TEXT NOT NULL,
7   `deadline` TEXT NOT NULL,
8   `salary` TEXT NOT NULL,
9   `company_id` INTEGER,
10  PRIMARY KEY(`job_id` AUTOINCREMENT),
11  FOREIGN KEY(`company_id`) REFERENCES `company`(`company_id`)
12 );

```

Tabel Interview

Table **interview**

Fields **Constraints**

Name	Type	NN	PK	AI	U	Default	Check	Collation	Foreign Key
interview_id	INTEGER	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				
job_id	INTEGER	<input checked="" type="checkbox"/>							"job"("job_id")
date	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>						
time	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>						
jobseeker_id	INTEGER	<input checked="" type="checkbox"/>							"jobseeker"("jobseeker_id")

```

1 CREATE TABLE "interview" (
2     "interview_id" INTEGER NOT NULL UNIQUE,
3     "job_id" INTEGER,
4     "date" TEXT NOT NULL,
5     "time" TEXT NOT NULL,
6     "jobseeker_id" INTEGER,
7     PRIMARY KEY("interview_id" AUTOINCREMENT),
8     FOREIGN KEY("jobseeker_id") REFERENCES "jobseeker"( "jobseeker_id"),
9     FOREIGN KEY("job_id") REFERENCES "job"( "job_id")
10    );

```

Tabel Jobseeker

Table **jobseeker**

Fields **Constraints**

Name	Type	NN	PK	AI	U	Default	Check	Collation	Foreign Key
jobseeker_id	INTEGER	<input checked="" type="checkbox"/>							
name_js	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>						
phone_number	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>						
address	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>						
education	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>						
account_id	INTEGER	<input checked="" type="checkbox"/>							"account"("account_id")
birth_date	TEXT	<input checked="" type="checkbox"/>							

```

1 CREATE TABLE "jobseeker" (
2     "jobseeker_id" INTEGER NOT NULL UNIQUE,
3     "name_js" TEXT NOT NULL,
4     "phone_number" TEXT NOT NULL,
5     "address" TEXT NOT NULL,
6     "education" TEXT NOT NULL,
7     "account_id" INTEGER,
8     "birth_date" TEXT,
9     PRIMARY KEY("jobseeker_id" AUTOINCREMENT),
10    FOREIGN KEY("account_id") REFERENCES "account"( "account_id")
11    );

```

Tabel Result

Table **result**

Fields **Constraints**

Name	Type	NN	PK	AI	U	Default	Check	Collation	Foreign Key
result_id	INTEGER	<input checked="" type="checkbox"/>							
jobseeker_id	INTEGER	<input checked="" type="checkbox"/>							"jobseeker"("jobseeker_id")
job_id	INTEGER	<input checked="" type="checkbox"/>							"job"("job_id")
status	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>						

```

1 CREATE TABLE "result" (
2     "result_id" INTEGER NOT NULL UNIQUE,
3     "jobseeker_id" INTEGER,
4     "job_id" INTEGER,
5     "status" TEXT NOT NULL,
6     PRIMARY KEY("result_id" AUTOINCREMENT),
7     FOREIGN KEY("job_id") REFERENCES "job"( "job_id"),
8     FOREIGN KEY("jobseeker_id") REFERENCES "jobseeker"( "jobseeker_id")
9    );

```

Tabel Resume

Fields

Name	Type	NN	PK	AI	U	Default
resume_id	INTEGER	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
link_file	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
jobseeker_id	INTEGER	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

```

1 CREATE TABLE `resume` (
2   `resume_id` INTEGER NOT NULL UNIQUE,
3   `link_file` TEXT NOT NULL,
4   `jobseeker_id` INTEGER,
5   PRIMARY KEY(`resume_id` AUTOINCREMENT),
6   FOREIGN KEY(`jobseeker_id`) REFERENCES `jobseeker`(`jobseeker_id`)
7 );

```

Tabel Requirement

Fields

Name	Type	NN	PK	AI	U	Default
requirement	TEXT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
req_id	INTEGER	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

```

1 CREATE TABLE `requir` (
2   `requirement` TEXT NOT NULL,
3   `req_id` INTEGER NOT NULL UNIQUE,
4   PRIMARY KEY(`req_id` AUTOINCREMENT)
5 );

```

INSERT INTO

Memasukkan atau menyisipkan data ke dalam tabel

```
INSERT INTO company (name_com, location, account_id) VALUES ('Faster Technology', '825 Jason Bypass', '239')
```

company_id	name_com		location	account_id
Filter	Filter	Filter	Filter	Filter
40	NatFood CJSC		0983 Jeremy Burgs	40
41	Ameriabank CJSC		146 Cheryl Highway	41
42	Faster Technology		825 Jason Bypass	239

UPDATE

Memperbarui atau mengubah data

```
UPDATE jobseeker SET phone_number='(320)788-5710' WHERE jobseeker_id='3'
```

Sebelum

jobseeker_id	name_js	phone_number
Filter	Filter	Filter
1	Alan Tran	(299)375-1746
2	Alexander Edwards	(966)697-9845

Sesudah

jobseeker_id	name_js	phone_number
Filter	Filter	Filter
1	Alan Tran	(299)375-1746
2	Alexander Edwards	(966)697-9845

SELECT WHERE

Memilih data, hanya record (baris) yang memenuhi kriteria yang diinginkan

```
SELECT * FROM jobseeker WHERE jobseeker_id='7'
```

jobseeker_id	name_js	phone_number	address	education	account_id	birth_date
7	Amy Alexander	(572)393-4713	5994 Martha Place Apt.766	PhD	48	01/04/1992

SELECT ORDER BY

Memilih data dan mengurutkan hasilnya berdasarkan kriteria yang diinginkan

```
SELECT * FROM company ORDER BY name_com
```

company_id	name_com	location	account_id
33	Alpha Food Service LLC	18403 Matthews Inlet Apt.555	33
41	Ameriabank CJSC	146 Cheryl Highway	41
7	Armenian Card CJSC	746 Brooke Fords Apt.920	7
8	BetArchitect LLC	728 Carter Plain	8

SELECT COUNT

Menghitung record (baris) yang memenuhi kriteria yang diminta

```
SELECT COUNT(*) FROM br_apply WHERE jobseeker_id = '82'
```

COUNT(*)
3

JOIN ON

Menggabungkan baris pada beberapa tabel dalam database sesuai dengan kolom yang berkaitan antara tabel tersebut

```
SELECT job.title, result.status FROM result JOIN job ON result.job_id = job.job_id WHERE result.jobseeker_id = '82'
```

	title	status
1	Receptionist/ Administrative Assistant	Not Accepted
2	Sociologist/ Quality Control Manager	Accepted
3	Construction Foreman/ Supervisor	Not Accepted

DELETE

Menghapus data pada tabel

```
DELETE FROM account WHERE email = 'fasttechnology@gmail.com'
```

Sebelum

account_id	email
Filter	Filter
236	alexcollins@example.org
237	whamilton@example.com
238	nixonstephanie@example.org
239	fasttechnology@gmail.com

Sesudah

account_id	email
Filter	Filter
236	alexcollins@example.org
237	whamilton@example.com
238	nixonstephanie@example.org

3.2 Implementasi GUI

Graphical User Interface (GUI) adalah antarmuka program yang bertindak sebagai media komunikasi antara pengguna dan perangkat lunak (Mandel, 2002). GUI telah menjadi cara penting dan telah diterima dalam berinteraksi dengan perangkat lunak saat ini (Muhtadi, M. M., Friyadi, M. D., & Rahmani, A. (2019)). Graphical User Interface (GUI) adalah suatu antarmuka yang memungkinkan user berinteraksi melalui elemen atau yang biasa disebut *widgets*. Dengan adanya GUI, pengguna dapat mengakses fitur-fitur dengan lebih mudah. Melalui GUI yang dibuat, pengguna dapat melakukan query ke database dan mendapatkan respons secara *real-time* tanpa perlu menulis *query* SQL secara manual. Dengan adanya implementasi GUI, diharapkan database pencarian dan rekrutmen kerja dapat lebih mudah digunakan dan lebih efektif dalam membantu proses pencarian dan rekrutmen kerja.

3.2.1 Package yang Digunakan

Berikut adalah beberapa package yang digunakan dalam implementasi GUI:

- 1. sqlite3:**

sqlite3 adalah modul Python yang menyediakan interface untuk berinteraksi dengan database SQLite. sqlite3 digunakan untuk membuat koneksi ke database SQLite, mengeksekusi query SQL, dan mengelola interaksi database.

- 2. pandas.io.sql as pds:**

pandas.io.sql adalah submodul dari library pandas yang mendukung operasi SQL. pandas.io.sql digunakan untuk membaca, mengakses, dan menampilkan data antara Pandas DataFrame dan database SQL.

- 3. pandas as pd:**

pandas adalah library *open-source* untuk analisis data dan manipulasi data dalam Python. pandas digunakan untuk berbagai operasi seperti membaca dan menulis data dalam berbagai format (CSV, Excel, SQL, dll.), mengelola dan menganalisis data, dan melakukan operasi data yang kompleks dengan DataFrame dan Series.

- 4. Datetime:**

`datetime` adalah modul bawaan Python yang menyediakan fitur untuk memanipulasi tanggal dan waktu. `datetime` digunakan untuk membuat, memanipulasi, dan memformat objek tanggal dan waktu.

5. `ipywidgets`:

`ipywidgets` merupakan library dalam Python, yang memfasilitasi pembuat notebook Jupyter dan Google Colab dalam menciptakan GUI yang interaktif secara langsung di dalam notebook.

- Layout: Untuk menyediakan berbagai opsi tata letak untuk widget, seperti ukuran, margin, dan lainnya.
- widgets: Untuk membuat antarmuka pengguna interaktif, seperti tombol, kotak teks, dropdown, dll.
- Style: Untuk mengatur gaya visual dari widget.
- Button: Untuk membuat tombol interaktif.

6. `IPython.display`:

- display: Untuk menampilkan objek.
- clear_output: Untuk menghapus output cell notebook Jupyter.

3.2.2 Fitur *Login*

Sebelum *user* dapat mengakses fitur-fitur, *user* diharuskan untuk login terlebih dahulu. Berikut adalah beberapa *function* yang digunakan dalam implementasi GUI untuk fitur *login*

1. `connect_to_db()`: Fungsi ini digunakan untuk membuat koneksi ke *database*.

```
# Koneksi ke Database
def connect_to_db():
    global conn, cursor
    conn = sqlite3.connect('Database Final Project.db')
    cursor = conn.cursor()
```

Cuplikan *codes* untuk koneksi ke *database*

2. `enter(email, password)`: Fungsi ini digunakan untuk memeriksa apakah email yang diinput sudah pernah digunakan sebelumnya. Jika email belum pernah digunakan, maka fungsi ini akan menambahkan data ke dalam tabel account.

```

# Memeriksa apakah email sudah pernah digunakan sebelumnya
def enter(email, password):
    query = "SELECT * FROM account WHERE email =?"
    cursor.execute(query, (email,))
    result = cursor.fetchone()
    if result:
        with output_widget:
            clear_output()
            print("You must enter an email that you have never used")
        return False
    else:
        query = "INSERT INTO account (email, password) VALUES (?,?)"
        cursor.execute(query, (email, password))
        conn.commit()
        query = "SELECT last_insert_rowid()"
        cursor.execute(query)
        account_id = cursor.fetchone()[0] # Mengambil account_id
        with output_widget:
            clear_output()
        return account_id # Return account_id

```

Cuplikan *codes* untuk memeriksa email yang di-*input user*

3. **check(email, password)**: Fungsi ini digunakan untuk memeriksa apakah email dan password yang diinput sesuai dengan data yang ada di tabel account.

```

# Mencocokkan input dengan data email di tabel account
def check(email, password):
    query = 'SELECT * FROM account WHERE email =? AND password =?'
    cursor.execute(query, (email, password))
    result = cursor.fetchone()
    conn.commit()
    if result:
        global account_id
        account_id = result[0] # Menyimpan account_id
        return True
    else:
        return False

```

Cuplikan *codes* untuk mencocokkan *input* dengan data email di tabel account

4. **login(b)**: Fungsi ini digunakan untuk memeriksa apakah email dan password yang di-*input* sesuai dengan data yang ada di tabel “account”. Jika sesuai, maka fungsi ini akan menampilkan menu sesuai dengan jenis akun yang diinput (jobseeker atau company).

```

# Login
def login(b):
    with output_widget:
        clear_output()
        email = email_widget.value
        password = password_widget.value
        if check(email, password):
            query_jobseeker = "SELECT 1 FROM jobseeker WHERE account_id =?"
            query_company = "SELECT 1 FROM company WHERE account_id =?"
            cursor.execute(query_jobseeker, (account_id,))
            result_jobseeker = cursor.fetchone()
            cursor.execute(query_company, (account_id,))
            result_company = cursor.fetchone()
            if result_jobseeker:
                display_menu_jobseeker()
            elif result_company:
                display_menu_company()
            else:
                print("Account not found in either jobseeker or company table")
        else:
            print("Something wrong")

```

Cuplikan codes untuk *login*

3.2.3 Fitur *Register*

Jika belum mempunyai akun, *user* dapat melakukan *register* terlebih dahulu untuk membuat akun baru. Berikut adalah beberapa *function* yang digunakan dalam implementasi GUI untuk fitur *register*

1. **register(b)**: Fungsi ini digunakan untuk memeriksa apakah email yang di-*input* sudah pernah digunakan sebelumnya. Jika email belum pernah digunakan, maka fungsi ini akan menambahkan data ke dalam tabel account dan menampilkan menu sesuai dengan jenis akun yang diinput (jobseeker atau company).

```

# Register akun baru
def register(b):
    with output_widget:
        clear_output()
        email = email_widget.value
        password = password_widget.value
        account_id = enter(email, password) # Mengambil account_id dari fungsi enter
        if not account_id:
            print("Failed")
        else:
            print("Success! Please choose:")
            display_register_menu(account_id)

```

Cuplikan codes untuk *register* akun baru

2. **display_register_menu(account_id)**: Fungsi ini digunakan untuk menampilkan menu pendaftaran akun jobseeker atau company setelah mendaftar akun baru. User dapat memilih jenis akun (sebagai *jobseeker* atau *company*) ketika ingin mendaftar akun baru.

```

def display_register_menu(account_id):
    login_menu_widget = widgets.VBox([
        widgets.Button(description='Jobseeker', button_style='info', tooltip='Login as Jobseeker'),
        widgets.Button(description='Company', button_style='info', tooltip='Login as Company')
    ])
    display(login_menu_widget)
    jobseeker_button = login_menu_widget.children[0]
    jobseeker_button.on_click(lambda b: register_jobseeker(account_id))
    company_button = login_menu_widget.children[1]
    company_button.on_click(lambda b: register_company(account_id))

```

Cuplikan *codes* untuk menampilkan menu pendaftaran

3. **register_company(account_id)**: Fungsi ini digunakan untuk mendaftar akun sebagai *company*. Jika mendaftar berhasil, maka akan menampilkan menu pendaftaran akun *company*.
4. **complete_register_company(company_name, location, account_id)**: Fungsi ini digunakan untuk menyelesaikan pendaftaran akun *company*. Jika pendaftaran berhasil, maka akan menampilkan konfirmasi pendaftaran dan login lagi.

```

# Register akun sebagai Company
def register_company(account_id):
    clear_output()
    company_name_widget = widgets.Text(value='', description='Company Name', style={'description_width': 'initial'})
    location_widget = widgets.Text(value='', description='Location', style={'description_width': 'initial'})
    display(company_name_widget, location_widget)
    register_company_button = widgets.Button(description='Register Company')
    register_company_button.on_click(lambda b: complete_register_company(company_name_widget.value, location_widget.value, account_id))
    display(register_company_button)

def complete_register_company(company_name, location, account_id):
    query = "INSERT INTO company (name_com, location, account_id) VALUES (?, ?, ?)"
    cursor.execute(query, (company_name, location, account_id))
    conn.commit()
    print("Company registration complete!")
    login_again_button = widgets.Button(description='Login Again')
    login_again_button.on_click(lambda b: login_again())
    display(login_again_button)

```

Cuplikan *codes function* jika register sebagai *company*, yakni *function* `register_company(account_id)` dan `complete_register_company(company_name, location, account_id)`

5. **register_jobseeker(account_id)**: Fungsi ini digunakan untuk mendaftar akun sebagai *jobseeker*. Jika mendaftar berhasil, maka akan menampilkan menu pendaftaran akun *jobseeker*.
6. **complete_register_jobseeker(name, address, phone, education, birth_date, account_id)**: Fungsi ini digunakan untuk menyelesaikan pendaftaran akun *jobseeker*. Jika pendaftaran berhasil, maka akan menampilkan konfirmasi pendaftaran dan login lagi.

```

# Register akun sebagai Jobseeker
def register_jobseeker(account_id):
    clear_output()
    name_widget = widgets.Text(value='', description='Name:', style={'description_width': 'initial'})
    address_widget = widgets.Textarea(value='', description='Address:', style={'description_width': 'initial'})
    phone_widget = widgets.Text(value='', description='Phone:', style={'description_width': 'initial'})
    education_widget = widgets.Dropdown(options=['High School', 'Bachelor\'s Degree', 'Master\'s Degree', 'PhD'],
                                         value=None,
                                         description='Education:',
                                         style={'description_width': 'initial'})
    birth_date = widgets.DatePicker(description='Birth Date:', style={'description_width': 'initial'})
    display(name_widget, address_widget, phone_widget, education_widget, birth_date)
    register_jobseeker_button = widgets.Button(description='Register Jobseeker')
    register_jobseeker_button.on_click(lambda b: complete_register_jobseeker(
        name_widget.value,
        address_widget.value,
        phone_widget.value,
        education_widget.value,
        birth_date.value.strftime('%d/%m/%Y'),
        account_id
    ))
    display(register_jobseeker_button)

def complete_register_jobseeker(name, address, phone, education, birth_date, account_id):
    query = "INSERT INTO jobseeker (name_js, address, phone_number, education, birth_date, account_id) VALUES (?, ?, ?, ?, ?, ?)"
    cursor.execute(query, (name, address, phone, education, birth_date, account_id))
    conn.commit()
    print("Jobseeker registration complete!")
    login_again_button = widgets.Button(description='Login Again')
    login_again_button.on_click(lambda b: login_again())
    display(login_again_button)

```

Cuplikan *codes function* jika register sebagai *jobseeker*, yakni *function* `register_jobseeker(account_id)` dan `complete_register_jobseeker(name, address, phone, education, birth_date, account_id)`

7. **`login_again()`:** Fungsi ini digunakan untuk *login* ulang setelah mendaftar akun baru.

```

# Login ulang setelah Register
def login_again():
    clear_output()
    with output_widget:
        clear_output()
        display(email_widget)
        display(password_widget)
        display(login_button)
    display(output_widget)

```

Cuplikan *codes* untuk *login* kembali setelah berhasil mendaftar akun baru

3.2.3 Menu Utama

Terdapat menu utama yang berbeda untuk *jobseeker* dan *company*.

1. **`display_menu_jobseeker()`:** Fungsi ini digunakan untuk menampilkan menu utama bagi jobseeker.

```

## Menu Jobseeker
def display_menu_jobseeker():
    clear_output()
    print("Main Menu")
    main_menu_widget = widgets.VBox([
        widgets.Button(description='Profile', button_style='info', tooltip='View your profile'),
        widgets.Button(description='Search Job', button_style='info', tooltip='Search for jobs'),
        widgets.Button(description='Result', button_style='info', tooltip='View results'),
        widgets.Button(description='Interview', button_style='info', tooltip='View interview schedule'),
        widgets.Button(description='Apply Job', button_style='info', tooltip='Apply job'),
        widgets.Button(description='Update Resume', button_style='info', tooltip='Upload or update your resume')
    ])
    display(main_menu_widget)
    jobseeker_info_button = main_menu_widget.children[0]
    jobseeker_info_button.on_click(lambda b: display_jobseeker_info(account_id))
    search_job_button = main_menu_widget.children[1]
    search_job_button.on_click(lambda b: search_job(b))
    view_results_button = main_menu_widget.children[2]
    view_results_button.on_click(lambda b: display_results(b))
    view_interview_button = main_menu_widget.children[3]
    view_interview_button.on_click(lambda b: display_interview(b))
    apply_job_button = main_menu_widget.children[4]
    apply_job_button.on_click(lambda b: apply_job(b))
    update_resume_button = main_menu_widget.children[5]
    update_resume_button.on_click(lambda b: update_resume(b))

```

Cuplikan codes untuk menampilkan menu bagi *jobseeker*

2. **display_menu_company()**: Fungsi ini digunakan untuk menampilkan menu utama untuk company.

```

## Menu Company
def display_menu_company():
    clear_output()
    print("Main Menu")
    company_menu_widget = widgets.VBox([
        widgets.Button(description='Post Job', button_style='info', tooltip='Post a new job'),
        widgets.Button(description='Jobseeker', button_style='info', tooltip='View applicants for your jobs'),
        widgets.Button(description='Interview', button_style='info', tooltip='Manage interviews'),
        widgets.Button(description='Result', button_style='info', tooltip='Add result for the jobseeker')
    ])
    display(company_menu_widget)
    post_job_button = company_menu_widget.children[0]
    post_job_button.on_click(lambda b: post_job(b))
    view_applicants_button = company_menu_widget.children[1]
    view_applicants_button.on_click(lambda b: view_applicants(account_id))
    interview_button = company_menu_widget.children[2]
    interview_button.on_click(lambda b: display_interview_menu())
    add_result_button = company_menu_widget.children[3]
    add_result_button.on_click(lambda b: create_interview_result(account_id))

```

Cuplikan codes untuk menampilkan menu *company*

3.2.3 Fitur *Jobseeker*

Jika *user* berhasil login sebagai *jobseeker*, akan ditampilkan menu *jobseeker* di mana *user* dapat memilih fitur yang ingin digunakan sebagai *jobseeker*.

3. **search_job(b)**: Fungsi ini digunakan untuk mencari pekerjaan yang sesuai dengan kriteria yang dimasukkan. Jika pencarian berhasil, maka akan menampilkan data pekerjaan yang sesuai.

```

# Mencari pekerjaan
def search_job(b):
    clear_output()
    job_title_input = widgets.Text(value='', description='Job Title:', style={'description_width': 'initial'})
    search_output = widgets.Output()
    display(job_title_input, search_output)
    def search_job_callback(b):
        job_title = job_title_input.value
        query = """
            SELECT j.job_id, j.title, j.date, j.type, j.description, j.deadline, j.salary, c.name_com, q.qualification, r.requirement
            FROM job j
            LEFT JOIN company c ON j.company_id = c.company_id
            LEFT JOIN br_qualif bq ON j.job_id = bq.job_id
            LEFT JOIN qualif q ON bq.Qualif_id = q.qualif_id
            LEFT JOIN br_requir br ON j.job_id = br.job_id
            LEFT JOIN requir r ON br.req_id = r.req_id
            WHERE j.title LIKE '%' || ? || '%'
        """
        cursor.execute(query, (job_title,))
        results = cursor.fetchall()

    if results:
        data = {}
        for row in results:
            job_id = row[0]
            if job_id not in data:
                data[job_id] = {
                    'title': row[1],
                    'date': row[2],
                    'type': row[3],
                    'description': row[4]
                }
    """

```

Cuplikan codes untuk fitur mencari pekerjaan bagi *jobseeker*

4. **display_results(b):** Fungsi ini digunakan untuk menampilkan hasil lamaran pekerjaan yang sudah dilakukan. Jika ada hasil lamaran, maka akan menampilkan data pekerjaan yang dilamar.

```

# Menampilkan hasil lamaran
def display_results(b):
    clear_output()
    result_widget = widgets.Output()
    query = "SELECT jobseeker_id FROM jobseeker WHERE account_id =?"
    cursor.execute(query, (account_id,))
    jobseeker_id = cursor.fetchone()[0]
    query1 = "SELECT j.title, r.status FROM result r JOIN job j ON r.job_id = j.job_id WHERE r.jobseeker_id =?"
    cursor.execute(query1, (jobseeker_id,))
    results = cursor.fetchall()
    if not results:
        with result_widget:
            print("There is no result")
            back_button = widgets.Button(description='Back to Main Menu')
            back_button.on_click(lambda b: display_menu_jobseeker())
            display(back_button)
    else:
        df = pd.DataFrame(results, columns=["Job Title", "Status"])
        with result_widget:
            display(df)
            back_button = widgets.Button(description='Back to Main Menu')
            back_button.on_click(lambda b: display_menu_jobseeker())
            display(back_button)
        display(result_widget)

```

Cuplikan codes untuk fitur menampilkan status hasil lamaran pekerjaan

5. **display_interview(b):** Fungsi ini digunakan untuk menampilkan jadwal *interview* yang sudah dijadwalkan. Jika ada jadwal *interview*, maka akan menampilkan data *interview* yang sudah dijadwalkan.

```

# Menampilkan jadwal interview
def display_interview(b):
    clear_output()
    interview_widget = widgets.Output()
    query = """
        SELECT j.title, c.name_com, i.date, i.time
        FROM interview i
        JOIN job j ON i.job_id = j.job_id
        JOIN company c ON j.company_id = c.company_id
        JOIN jobseeker js ON i.jobseeker_id = js.jobseeker_id
        WHERE js.account_id = ?
    """
    cursor.execute(query, (account_id,))

    results = cursor.fetchall()
    if not results:
        with interview_widget:
            print("There is no result")
            back_button = widgets.Button(description='Back to Main Menu')
            back_button.on_click(lambda b: display_menu_jobseeker())
            display(back_button)
    else:
        df = pd.DataFrame(results, columns=['Title', 'Company', 'Date', 'Time'])
        with interview_widget:
            display(df)
            back_button = widgets.Button(description='Back to Main Menu')
            back_button.on_click(lambda b: display_menu_jobseeker())
            display(back_button)
            display(interview_widget)

```

Cuplikan *codes* untuk fitur menampilkan jadwal *interview* yang sudah dijadwalkan

6. **apply_job(b)**: Fungsi ini digunakan untuk melamar pekerjaan. Jika melamar berhasil, maka akan menampilkan konfirmasi lamaran pekerjaan.

```

# Melamar pekerjaan
def apply_job(b):
    clear_output()
    global job_id_input
    job_id_input = widgets.Text(value='', placeholder='Insert job_id', description='Job ID:', disabled=False)
    apply_button = widgets.Button(description='Apply')
    apply_button.on_click(apply_job_callback)
    display(job_id_input)
    display(apply_button)
    back_button = widgets.Button(description='Back to Main Menu')
    back_button.on_click(lambda b: display_menu_jobseeker())
    display(back_button)

```

Cuplikan *codes* untuk fitur melamar pekerjaan

7. **display_jobseeker_info(account_id)**: Fungsi ini digunakan untuk menampilkan informasi akun jobseeker. Jika informasi akun jobseeker sudah ada, maka akan menampilkan data akun jobseeker.

```

# Menampilkan profile
def display_jobseeker_info(account_id):
    clear_output()

    cursor.execute("SELECT * FROM jobseeker WHERE account_id =?", (account_id,)) # Mengambil data jobseeker berdasarkan account_id
    jobseeker_data = cursor.fetchone()

    cursor.execute("SELECT COUNT(*) FROM br_apply WHERE jobseeker_id =?", (jobseeker_data[0],)) # Jumlah pekerjaan yang sudah dilamar
    num_jobs_applied = cursor.fetchone()[0]

    cursor.execute("SELECT link_file FROM resume WHERE jobseeker_id =?", (jobseeker_data[0],)) # Mengambil link_file dari tabel resume
    resume_data = cursor.fetchone()
    link_file = resume_data[0] if resume_data else None

    df = pd.DataFrame([{
        'Name': jobseeker_data[1],
        'Phone Number': jobseeker_data[2],
        'Address': jobseeker_data[3],
        'Education': jobseeker_data[4],
        'Birth Date': jobseeker_data[6],
        'Number of Jobs Applied': num_jobs_applied,
        'Resume': link_file,
    }], columns=['Name', 'Phone Number', 'Address', 'Education', 'Birth Date', 'Number of Jobs Applied', 'Resume'])

    # display
    display(df)
    update_button = widgets.Button(description='Update Profile')
    update_button.on_click(lambda b: update_profile(account_id))
    display(update_button)
    back_button = widgets.Button(description='Back to Main Menu')
    back_button.on_click(lambda b: display_menu_jobseeker())
    display(back_button)

```

Cuplikan *codes* untuk fitur menampilkan informasi akun *jobseeker*

8. **update_profile(account_id)**: Fungsi ini digunakan untuk mengupdate informasi akun jobseeker. Jika update berhasil, maka akan menampilkan konfirmasi update.

```

def update_profile_callback(b): # Fungsi untuk update profile
    name = form_widgets[0].value
    phone_number = form_widgets[1].value
    address = form_widgets[2].value
    education = form_widgets[3].value
    birth_date = form_widgets[4].value

    # Update data jobseeker
    cursor.execute("UPDATE jobseeker SET name=?, phone_number=?, address=?, education=?, birth_date=? WHERE account_id=?", (name, phone_number, address, education, birth_date, account_id))
    conn.commit()

    print("Profile updated successfully!") # Print statement setelah berhasil
    back_button = widgets.Button(description='Back to Main Menu')
    back_button.on_click(lambda b: display_menu_jobseeker())
    display(back_button)

    update_button = widgets.Button(description='Submit')
    update_button.on_click(update_profile_callback)

    # display
    display(*form_widgets)
    display(update_button)

```

Cuplikan *codes* untuk fitur mengupdate informasi akun *jobseeker*

9. **update_resume(b)**: Fungsi ini digunakan untuk mengupdate resume akun *jobseeker*. Jika update berhasil, maka akan menampilkan konfirmasi *update*.

```

def update_resume(b): # Mengupdate resume
    clear_output()

    link_input = widgets.Text(value='', placeholder='Enter the link to your resume', description='Link:', disabled=False)
    display(link_input)

    def upload_resume(b):
        cursor.execute("SELECT jobseeker_id FROM jobseeker WHERE account_id =?", (account_id,))
        jobseeker_id = cursor.fetchone()[0]

        cursor.execute("SELECT * FROM resume WHERE jobseeker_id =?", (jobseeker_id,)) # Periksa apakah jobseeker_id sudah ada di tabel resume
        if cursor.fetchone():
            cursor.execute("UPDATE resume SET link_file=? WHERE jobseeker_id=?", (link_input.value, jobseeker_id)) # Jika ya, update
            conn.commit()
            print("Resume updated successfully!")
        else:
            cursor.execute("INSERT INTO resume (jobseeker_id, link_file) VALUES (?,?)", (jobseeker_id, link_input.value)) # Jika tidak, masukkan record baru
            conn.commit()
            print("Resume added successfully!")

    back_button = widgets.Button(description='Back to Main Menu')
    back_button.on_click(lambda b: display_menu_jobseeker())
    display(back_button)

# display
upload_button = widgets.Button(description='Submit')
upload_button.on_click(upload_resume)
display(upload_button)

```

Cuplikan codes untuk fitur mengupdate *link resume jobseeker*.

3.2.3 Fitur *Company*

Jika *user* berhasil login sebagai *company*, akan ditampilkan menu *company* di mana *user* dapat memilih fitur yang ingin digunakan sebagai *company*.

1. **view_schedule(b):** Fungsi ini digunakan untuk melihat jadwal *interview* dari *jobseeker* yang melamar pada *company* tersebut

```

# Menampilkan jadwal interview
def view_schedule(b):
    clear_output()
    output = widgets.Output()

    # Mengambil data jobseeker yang sudah melamar di company
    query = """
        SELECT js.jobseeker_id, js.name_js, j.title, i.date, i.time
        FROM jobseeker js
        JOIN interview i ON js.jobseeker_id = i.jobseeker_id
        JOIN job j ON i.job_id = j.job_id
        WHERE j.company_id = (SELECT c.company_id FROM company c WHERE c.account_id =?)
    """
    cursor.execute(query, (account_id,)) # Berdasarkan account_id
    results = cursor.fetchall()

    if not results: # Jika tidak ada, print
        output.clear_output()
        with output:
            print("There are no interview scheduled.")
    else:
        df = pd.DataFrame(results, columns=['Jobseeker ID', 'Name', 'Job Title', 'Date', 'Time']) # Buat Dataframe

        with output:
            display(df)

    # display
    display(output)
    back_button = widgets.Button(description='Back to Main Menu')
    back_button.on_click(lambda b: display_menu_company())
    display(back_button)

```

Cuplikan codes untuk fitur melihat jadwal *interview jobseeker* yang melamar

2. **create_interview_schedule(account_id):** Fungsi ini digunakan untuk membuat jadwal interview baru. Jika jadwal interview berhasil akan diberikan pesan konfirmasi.

```

# Membuat jadwal interview
def create_interview_schedule(account_id):
    clear_output()

    cursor.execute("SELECT company_id FROM company WHERE account_id =?", (account_id,)) # Mengambil company_id berdasar account_id
    company_id = cursor.fetchone()[0]

    cursor.execute("SELECT job_id, title FROM job WHERE company_id =?", (company_id,)) # Mengambil job_id berdasar account_id
    job_ids = cursor.fetchall()

    # input
    job_id_input = widgets.Dropdown(options=[(job_id, title) for job_id, title in job_ids], description='Job ID:', disabled=False)
    date_input = widgets.DatePicker(description='Date:', disabled=False)
    time_input = widgets.Text(value='', placeholder='HH:MM', description='Time:', disabled=False)
    jobseeker_id_input = widgets.Text(value='', placeholder='Jobseeker ID', description='Jobseeker ID:', disabled=False)

    create_button = widgets.Button(description="Create Schedule", button_style='success') # tombol untuk create schedule

    def create_schedule(b):
        job_title = job_id_input.value
        date = date_input.value
        if date is not None:
            date_str = date.strftime('%d/%m/%Y')
        else:
            print("Please select a date!")
    .....

```

Cuplikan *codes* untuk fitur membuat jadwal interview baru

3. **view_applicants(b):** Fungsi ini digunakan untuk menampilkan daftar pelamar pekerjaan. Jika ada pelamar, maka akan menampilkan data pelamar pekerjaan.

```

# Menampilkan pelamar kerja
def view_applicants(b):
    clear_output()
    output = widgets.Output()

    # Mengambil data jobseeker yang sudah melamar di company
    query = """
    SELECT js.jobseeker_id, js.name_js, js.phone_number, js.address, js.education, r.link_file, j.title, js.birth_date
    FROM jobseeker_js
    JOIN resume r ON js.jobseeker_id = r.jobseeker_id
    JOIN br_apply ap ON js.jobseeker_id = ap.jobseeker_id
    JOIN job j ON ap.job_id = j.job_id
    WHERE j.company_id = (SELECT c.company_id FROM company c WHERE c.account_id =?)
    """

    cursor.execute(query, (account_id,)) # berdasar account_id
    results = cursor.fetchall()

    if not results:
        output.clear_output()
        with output:
            print("There are no jobseekers who apply at this company.") # apabila tidak ada data
    else:
        # buat dataframe
        df = pd.DataFrame(results, columns=['Jobseeker ID', 'Name', 'Phone Number', 'Address', 'Education', 'Resume', 'Job Title', 'Birth Date'])
        df['Age'] = df['Birth Date'].apply(lambda x: calculate_age(x)) # menambahkan kolom age
        df = df.drop('Birth Date', axis=1) # hapus kolom birth date

        # display
        output.clear_output()
        with output:
            display(df)

    display(output) # display interface
    back_button = widgets.Button(description='Back to Main Menu')
    back_button.on_click(lambda b: display_menu_company())
    display(back_button)

```

Cuplikan *codes* untuk fitur menampilkan informasi *jobseeker* yang melamar kerja pada suatu *company*

```

def calculate_age(birth_date): # menghitung umur
    today = datetime.today()
    birth_date = datetime.strptime(birth_date, '%d/%m/%Y')
    return today.year - birth_date.year - ((today.month, today.day) < (birth_date.month, birth_date.day))

```

Cuplikan *codes* untuk menghitung umur *jobseeker* berdasarkan tanggal lahir dari *database*.

4. **post_job(b):** Fungsi ini digunakan untuk mem-*posting* pekerjaan baru. Jika *posting* berhasil, maka akan menampilkan konfirmasi *posting* pekerjaan.

```

# Menambahkan job/lamaran kerja baru
def post_job(b):
    clear_output()

    form_widgets = [
        widgets.Text(value='', placeholder='Title', description='Title:'),
        widgets.DatePicker(value=date.today(), description='Today:'),
        widgets.Dropdown(options=['Part-time', 'Full-time', 'Freelance'], description='Type:'),
        widgets.Textarea(value='', placeholder='Description', description='Description:'),
        widgets.DatePicker(value=date.today(), description='Deadline:'),
        widgets.Text(value='', placeholder='Range Salary (xxx-xxx)', description='Salary:'),
        widgets.Textarea(value='', placeholder='Requirement (Setiap kalimat dipisahkan dengan ;)', description='Requirement:'),
        widgets.Textarea(value='', placeholder='Qualification (Setiap kalimat dipisahkan dengan ;)', description='Qualification:')
    ]

    submit_button = Button(description='Submit', button_style='info')
    submit_button.on_click(lambda x: add_job_to_db(account_id, *[widget.value for widget in form_widgets]))

    # display
    display(*form_widgets)
    display(submit_button)
    back_button = widgets.Button(description='Back to Main Menu')
    back_button.on_click(lambda b: display_menu_company())
    display(back_button)

```

Cuplikan *codes* untuk fitur menambahkan penawaran kerja bagi suatu *company*

5. **create_interview_result(account_id):** Fungsi ini digunakan untuk menambahkan hasil *interview* ke dalam tabel result.

```

# Menambahkan hasil interview
def create_interview_result(account_id):
    clear_output()

    cursor.execute("SELECT company_id FROM company WHERE account_id =?", (account_id,)) # company_id berdasar account_id
    company_id = cursor.fetchone()[0]

    cursor.execute("SELECT job_id, title FROM job WHERE company_id =?", (company_id,)) # job_id berdasar company_id
    job_ids = cursor.fetchall()

    cursor.execute("SELECT jobseeker_id FROM jobseeker WHERE account_id =?", (account_id,))
    jobseeker_ids = cursor.fetchall()

    # menu untuk input
    job_id_input = widgets.Dropdown(options=[(job_id, title) for job_id, title in job_ids], description='Job ID:', disabled=False)
    jobseeker_id_input = widgets.Text(value='', placeholder='Jobseeker ID', description='Jobseeker ID:', disabled=False)
    status_input = widgets.Dropdown(options=[('Accepted', 'Accepted'), ('Not Accepted', 'Not Accepted')], description='Status:', disabled=False)

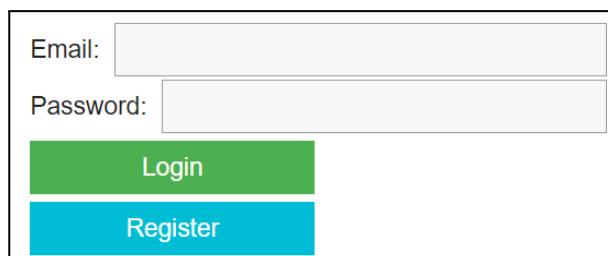
    add_button = widgets.Button(description="Add Result", button_style='success') # tombol add result

```

Cuplikan *codes* untuk fitur menambahkan status diterima atau tidaknya pelamaran kerja

3.3 Testing

Ketika pertama kali mengakses GUI, pengguna akan melihat tampilan awal seperti pada gambar dibawah.



Gambar 3.1

3.3.1 Fitur Login

Jika pengguna memilih "Login", pengguna akan diminta memasukkan email dan kata sandinya. Sistem kemudian memeriksa apakah email dan kata sandi cocok dengan akun yang ada pada database.

- Jika login berhasil, pengguna akan diarahkan ke menu utama baik sebagai *jobseeker* (Gambar 3.2) maupun *company* (Gambar 3.3), tergantung jenis akunnya.

The image contains two separate login forms side-by-side. Both forms have fields for 'Email' and 'Password'. The left form has an 'Email' field containing 'akuntes@gmail.com' and a 'Password' field with five dots. Below these are two buttons: a green 'Login' button and a blue 'Register' button. To the right of these buttons is a 'Main Menu' section with six blue buttons: 'Profile', 'Search Job', 'Result', 'Interview', 'Apply Job', and 'Update Resume'. The right form has an 'Email' field containing 'recruitment@career.com' and a 'Password' field with six dots. It also has a green 'Login' button and a blue 'Register' button. To its right is a 'Main Menu' section with four blue buttons: 'Post Job', 'Jobseeker', 'Interview', and 'Result'.

Gambar 3.2

Gambar 3.3

- Jika login gagal, pengguna diberitahu dan diberikan pilihan untuk mencoba lagi.

This image shows a login screen where both the email ('akuntes@gmail.com') and password fields are filled. Below the password field is a red rectangular error message box containing the text 'Something wrong'. The rest of the interface is identical to Gambar 3.2, with the 'Login' and 'Register' buttons and the 'Main Menu' options.

Gambar 3.4

3.3.2 Fitur Registrasi

Jika pengguna belum mempunyai akun pada database, maka pengguna harus registrasi terlebih dahulu. Langkah-langkahnya adalah sebagai berikut:

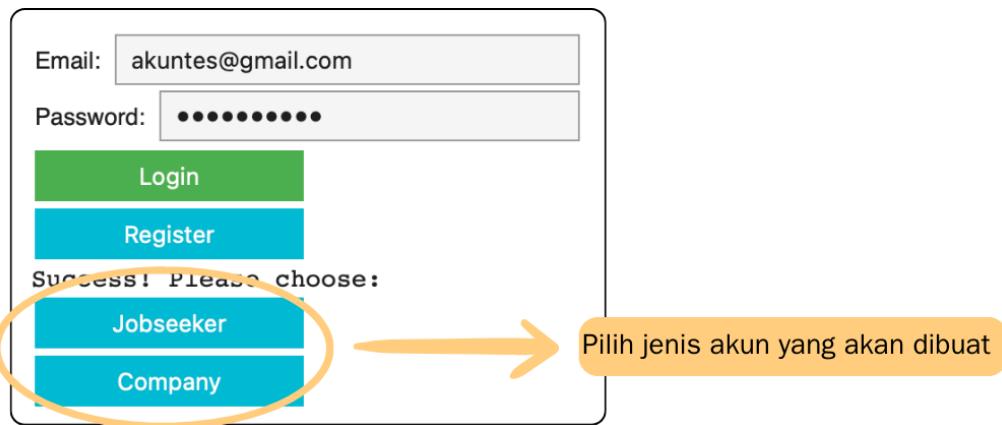
1. Masukkan email dan password pada tampilan awal.

2. Tekan tombol registrasi. Gunakanlah email yang belum pernah digunakan. Jika email yang dimasukkan sudah ada dalam database, maka akan ditampilkan kalimat pada Gambar 3.5.

You must enter an email that you have never used
Failed

Gambar 3.5

3. Setelah menekan tombol registrasi, pengguna diarahkan untuk memilih mendaftarkan akun sebagai *jobseeker* atau *company*.



Gambar 3.6

4. Setelah memilih salah satu, pengguna akan diarahkan untuk mengisi data. Gambar 3.7 sebagai *jobseeker* dan Gambar 3.8 sebagai *company*.

Gambar 3.8

Gambar 3.7

5. Setelah berhasil, klik tombol *login again* dan pengguna akan diarahkan untuk login kembali agar dapat menggunakan fitur selanjutnya.

3.3.3 Menu Utama

The screenshot shows a login form with fields for Email and Password. Below the form is a vertical menu titled "Main Menu" containing the following items: Profile, Search Job, Result, Interview, Apply Job, and Update Resume. The "Profile" item is highlighted with a green background.

Gambar 3.9

The screenshot shows a login form with fields for Email and Password. Below the form is a vertical menu titled "Main Menu" containing the following items: Post Job, Jobseeker, Interview, and Result. The "Post Job" item is highlighted with a green background.

Gambar 3.10

1. Main Menu Jobseeker (Gambar 3.9)

Menu utama sebagai jobseeker mencakup opsi untuk :

- melihat profil,
- mencari pekerjaan,
- melihat hasil,
- melihat jadwal wawancara,
- melamar pekerjaan, dan
- memperbarui resume.

2. Main Menu Company (Gambar 3.10)

Menu utama sebagai *company* mencakup opsi untuk

- memposting pekerjaan,
- melihat pelamar,
- membuat jadwal wawancara, dan
- menambahkan hasil untuk *jobseeker*.

3.3.4 Fitur Jobseeker

1. Profile

Fitur "Profile" memungkinkan pencari kerja untuk melihat informasi pribadi, seperti nama, nomor telepon, alamat, pendidikan, dan tanggal lahir (Gambar 3.11). Mereka juga dapat melihat jumlah pekerjaan yang telah mereka lamar dan link resume mereka. Untuk memperbarui profil, user dapat mengklik tombol "Update Profile"

(Gambar 3.12) dan mengklik tombol “Submit” setelah selesai. Pengguna akan diarahkan mengklik tombol “Back to Main Menu” setelahnya (Gambar 3.13).

Name	Phone Number	Address	Education	Birth Date	Number of Jobs Applied	Resume
0 Leslie Reynolds	(801)916-6749	693 Moore Mews Apt. 551	Master's Degree	17/04/1971		0 https://drive.google.com/uc?export=view&id=50c...
Update Profile Back to Main Menu						

Gambar 3.11

Name:

Phone Number:

Address:

Education:

Birth Date:

Profile updated successfully!

[Back to Main Menu](#)

Gambar 3.13

Gambar 3.12

2. Search Job

Fitur "Search Job" memungkinkan pencari kerja untuk mencari pekerjaan yang sesuai dengan *Job Title*. Gambar 3.14 menunjukkan contoh hasil pencarian pekerjaan dengan *keyword* software.

Job Title: Software									
Job ID	Title	Date	Type	Description	Deadline	Salary	Company	Qualifications	Requirements
0 4	Mid-Level Software Developer	04-08-2018	Full-time	The Mid-level Developer will focus on core soft...	18-09-2018	288001-318000 AMD	Synergy International Systems Inc., Armenian B...	Strong knowledge of OOA/OOD; From 3 to 5 yea...	Design, proto-type, develop and manage the tec...
1 19	Software Developer	19-05-2018	Full time	Software Delivery is one of the service offer...	25-06-2018	288001-318000 AMD	HSBC Bank Armenia CJSC	At least 3 years of experience in working with...	Assure full conformance of source codes to pro...
2 21	Junior Software Developer	27-11-2017	Full-time	The incumbent will join a team responsible for...	08-01-2018	228001-258000 AMD	FIP Software LLC	Experience with version control systems such a...	Identify innovative ideas and proofs of the co...
3 26	Java Software Developer	26-10-2017	Full time	Sourcio is seeking experienced Java Developers...	27-11-2017	348001-378000 AMD	Sourcio CJSC	At least 2 years of practical experience in Ja...	Communicate effectively with local management ...

Gambar 3.14

3. Result

Fitur "Result" memungkinkan jobseeker untuk melihat hasil dari pelamaran kerja (Gambar 3.15). Jika *jobseeker* belum pernah melamar kerja, akan ditampilkan seperti Gambar 3.16.

	Job Title	Status
0	QA Engineer	Not Accepted
1	Supervisor of Program Activities in Stepanakert	Not Accepted
Back to Main Menu		

Gambar 3.15

There is no result
[Back to Main Menu](#)

Gambar 3.16

4. Interview

Fitur "Interview" akan menampilkan jadwal interview yang telah dijadwalkan untuk *jobseeker*.

	Title	Company	Date	Time
0	QA Engineer	EPAM Systems, Inc.	21/01/2018	10:30
1	Supervisor of Program Activities in Stepanakert	Medecins Sans Frontieres - France, Armenian Br...	16/08/2018	13:00
Back to Main Menu				

Gambar 3.17

5. Apply Job

Fitur "Apply Job" memungkinkan *jobseeker* untuk melamar kerja dengan meng-*input* Job ID seperti pada Gambar 3.18.

Job ID:	<input type="text"/>
Apply	
Back to Main Menu	

Gambar 3.18

Jika *jobseeker* melamar pada *job* yang sudah pernah dilamar, akan ditampilkan Gambar 3.19. Jika *jobseeker* berhasil melamar kerja, akan ditampilkan Gambar 3.20.

Job ID:	<input type="text" value="6"/>
Apply	
Back to Main Menu	
You have already applied to be Sales Director in Coffee Trade LLC	

Gambar 3.19

Job ID:	<input type="text" value="31"/>
Apply	
Back to Main Menu	
You have successfully applied to be Assistant to Regional Manager in MLN Pharm Ltd	

Gambar 3.20

6. Update Resume

Fitur "Update Resume" memungkinkan *jobseeker* untuk mengupdate *link* resume (Gambar 3.21). Setelah memasukkan link dan mengklik tombol "Submit", akan ditampilkan Gambar 3.22.

Link: Enter the link to your resume
Submit

Gambar 3.21

Link: https://drive.google.com/file/d/1Vf...
Submit
Resume updated successfully!
Back to Main Menu

Gambar 3.22

3.3.5 Fitur Company

1. Post Job

Fitur "Post Job" memungkinkan company untuk menambahkan tawaran pekerjaan baru. Setelah berhasil submit job baru, akan ditampilkan Gambar 3.23.

Title: Title
Today: 06 / 06 / 2024
Type: Part-time
Description: Description
Deadline: 06 / 06 / 2024
Salary: Range Salary (xxx-xxx)
Requirement: Requirement (Setiap kalimat dipisahkan dengan ;)
Qualification: Qualification (Setiap kalimat dipisahkan dengan ;)
Submit
Back to Main Menu

Job has been added!

Gambar 3.23

2. Jobseeker

Fitur "Jobseeker" memungkinkan company untuk melihat jobseeker yang sudah melamar ke company mereka dan pekerjaan apa yang mereka lamar (Gambar 3.24).

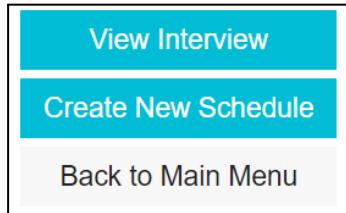
Jobseeker ID	Name	Phone Number	Address	Education	Link File	Job Title	Age
0	82 Jennifer Drake	(866)680-3729	975 Mark Cape Apt. 378	High School	https://drive.google.com/uc?export=view&id=466...	Receptionist/ Administrative Assistant	54
1	80 Jeffrey Stone	(742)366-4392	723 Laura Route	PhD	https://drive.google.com/uc?export=view&id=bf9...	Receptionist/ Administrative Assistant	59

Back to Main Menu

Gambar 3.24

3. Interview

Ketika pengguna mengklik tombol “Interview”, akan ditampilkan Gambar 3.25 yang memungkinkan pengguna untuk memilih antara “View Interview” atau “Create New Schedule”.



Gambar 3.25

Fitur "View Interview" memungkinkan company untuk melihat interview yang terjadwal (Gambar 3.26).

Jobseeker ID	Name	Job Title	Date	Time
0	102	Julia Green	Mathematician/ Statistician	14/04/2017 11:30
1	45	Deborah Bailey	Mathematician/ Statistician	18/04/2017 12:00
2	11	Anne Ingram	Mathematician/ Statistician	07/06/2024 12:00

Back to Main Menu

Gambar 3.26

Fitur "Create New Schedule" memungkinkan company untuk menjadwalkan interview dengan jobseeker. Apabila jobseeker ID yang dimasukkan bukanlah pelamar pada company, akan ditampilkan “Jobseeker has not yet applied for the job”. Jika sudah, akan ditampilkan “The interview schedule has been created” (Gambar 3.27).

The form consists of several input fields and a button. At the top, there is a dropdown menu labeled "Job ID" with the value "8". Below it are two text input fields: "Date: dd/mm/yyyy" and "Time: HH:MM". Further down is a dropdown menu labeled "Jobseeker ...". A large green button at the bottom is labeled "Create Schedule". Below the form are two message boxes. The first message box, which is light blue, contains the text "Jobseeker has not yet applied for the job!". The second message box, which is light green, contains the text "The interview schedule has been created!".

Gambar 3.27

4. Result

Fitur “Result” memungkinkan company untuk menambahkan hasil lamaran kerja dengan pilihan Acceptor/Not Accepted (Gambar 3.28).

Job ID:	1
Jobseeker ...	Jobseeker ID
Status:	Accepted
Add Result	
Back to Main Menu	

Gambar 3.28

BAB 4 PENUTUP

Berdasarkan hasil simulasi yang telah dilakukan, dapat disimpulkan bahwa implementasi SQL dan GUI dalam Mini Project Database untuk Sains Data telah berhasil. Database yang digunakan adalah "Database Final Project.db" yang terdiri dari beberapa tabel relasional, yaitu account, sqlite_sequence, qualif, requir, br_apply, br_qualif, br_requir, interview, jobseeker, result, resume, dan job. Setiap tabel memiliki kolom dan kunci primer yang sesuai dengan kebutuhan data.

Implementasi SQL dalam Mini Project ini meliputi query untuk memasukkan, mengubah, menghapus, dan mengambil data dari database. Query-query ini digunakan untuk berinteraksi dengan database melalui GUI yang telah dibuat. GUI ini terdiri dari beberapa menu, yaitu login, register, menu utama jobseeker, dan menu utama company. Setiap menu memiliki fungsi yang berbeda-beda, seperti mencari pekerjaan, melamar pekerjaan, melihat hasil lamaran, dan lain-lain.

Dari kesimpulan ini, dapat disimpulkan bahwa Mini Project Database untuk Sains Data telah berhasil diimplementasikan menggunakan SQL dan GUI. Database yang digunakan telah terstruktur dengan baik serta dapat digunakan untuk menyimpan dan mengelola data yang diperlukan dalam Mini Project ini.

DAFTAR PUSTAKA

- Putra, D. W. T., & Putra, J. J. (2018). Perancangan Sistem Informasi Pencarian Lowongan Pekerjaan. *Jurnal Teknoif Teknik Informatika Institut Teknologi Padang*, 6(1), 48-54.
- Mandel, T. (2002). User/System Interface Design. *Encyclopedia of Information Systems*, 1, 1-4.
- Mathisugan, R. (2021). *Online job portal* (Doctoral dissertation).
- Muhtadi, M. M., Friyadi, M. D., & Rahmani, A. (2019, August). Analisis GUI testing pada aplikasi e-commerce menggunakan katalon. In Prosiding Industrial Research Workshop and National Seminar (Vol. 10, No. 1, pp. 1387-1393).
- Setiawati, P. (2018). Analisa Dan Perancangan Sistem Informasi Penyedia Lowongan Pekerjaan Yang Direkomendasi Berdasarkan Standar Kompetensi Kerja Nasional Indonesia (Skkni). *Ilmu Komput*, 3(2), 136-147.
- Suwarno, S., Widada, B., & Siswanti, S. (2015). Sistem Informasi Lowongan Pekerjaan Berbasis Web pada Balai Latihan Kerja Boyolali. *Jurnal Teknologi Informasi dan Komunikasi (TIKomSiN)*, 3(1).

LAMPIRAN

Lampiran 1: *Database* yang digunakan

▫ Project Database B Kelompok 4

Lampiran 2: *Codes* untuk implementasi GUI

♾ Final Project Kelompok 4