

## **TUGAS 3 - Aplikasi Foto Sharing Berbasis AWS**



### **DISUSUN OLEH:**

Achmed Hibatillah 225150707111024

Kevin Josua Situmorang 225150707111057

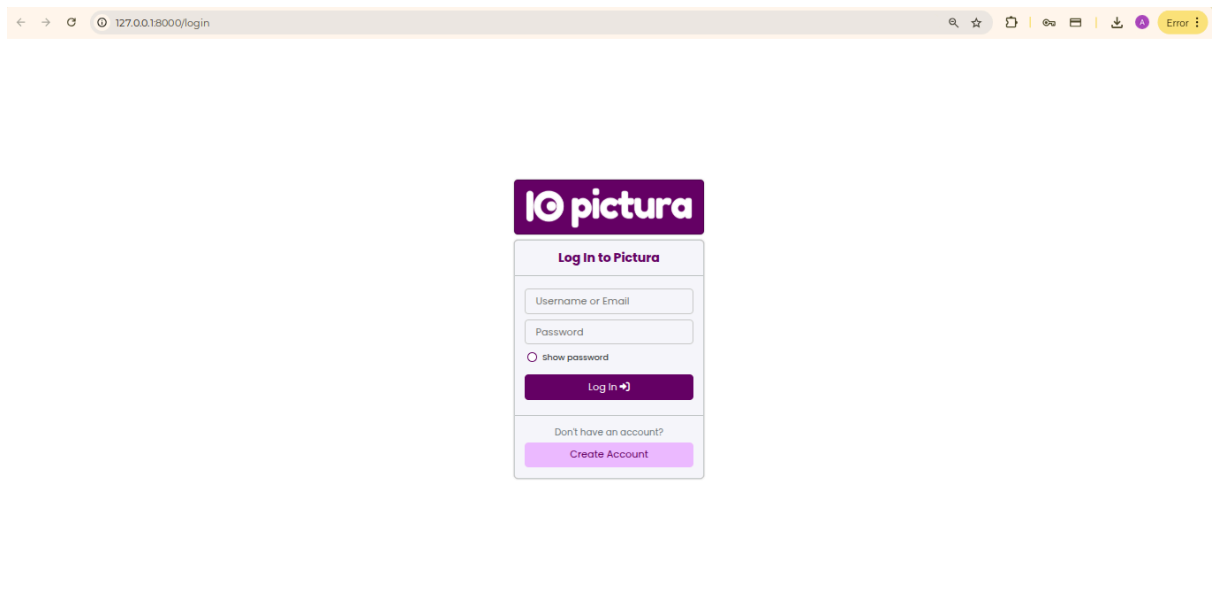
### **MATA KULIAH TEKNOLOGI BERBASIS CLOUD**

**PROGRAM STUDI TEKNOLOGI INFORMASI  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS BRAWIJAYA  
2025**



# Alur Kerja Aplikasi

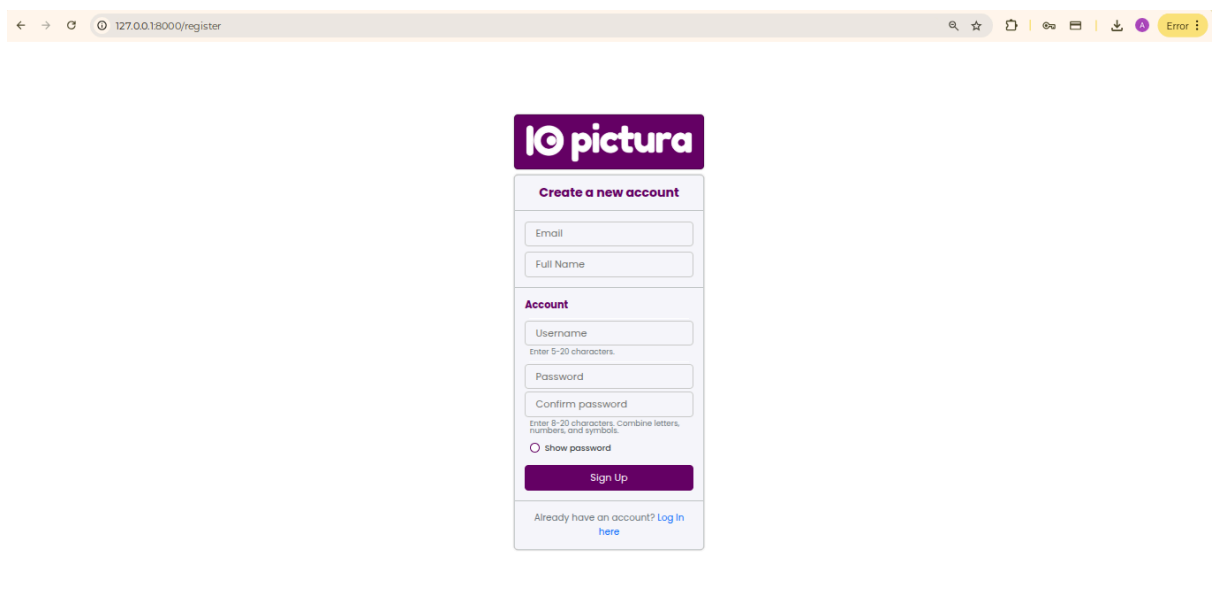
## 1. Halaman login



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/login". The page features the "pictura" logo at the top. Below the logo is a "Log in to Pictura" form. The form includes input fields for "Username or Email" and "Password", a "show password" toggle, a "Log in" button, and a "Create Account" link for users who do not have an account.

Form login akan melakukan validasi atas username/email dan password (hash) di database pictura.

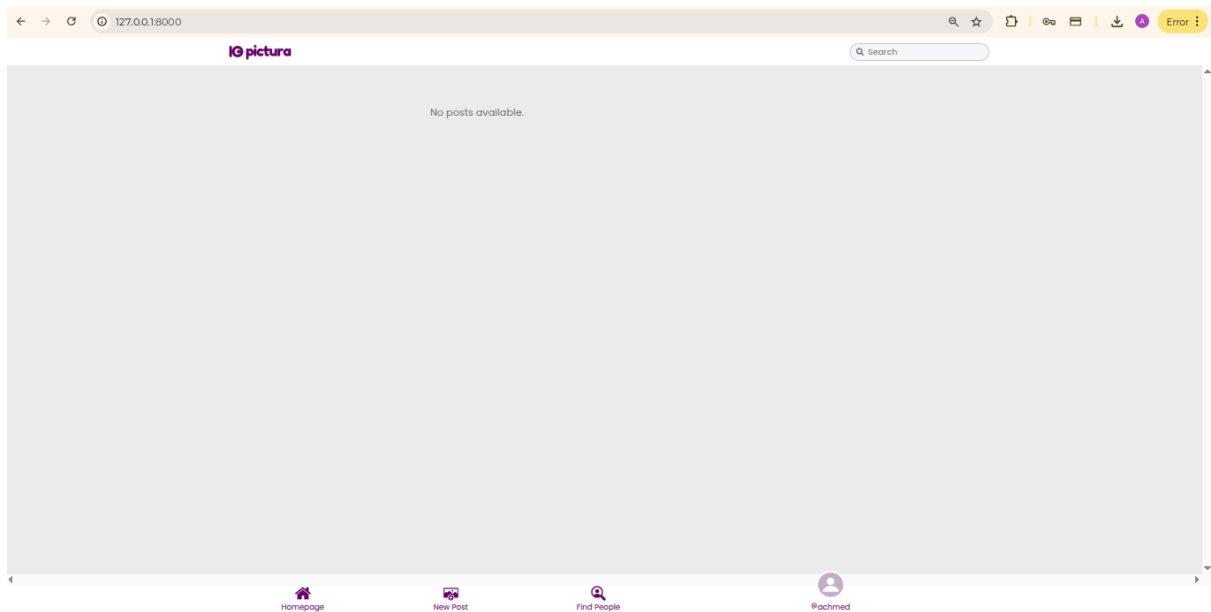
## 2. Halaman registrasi



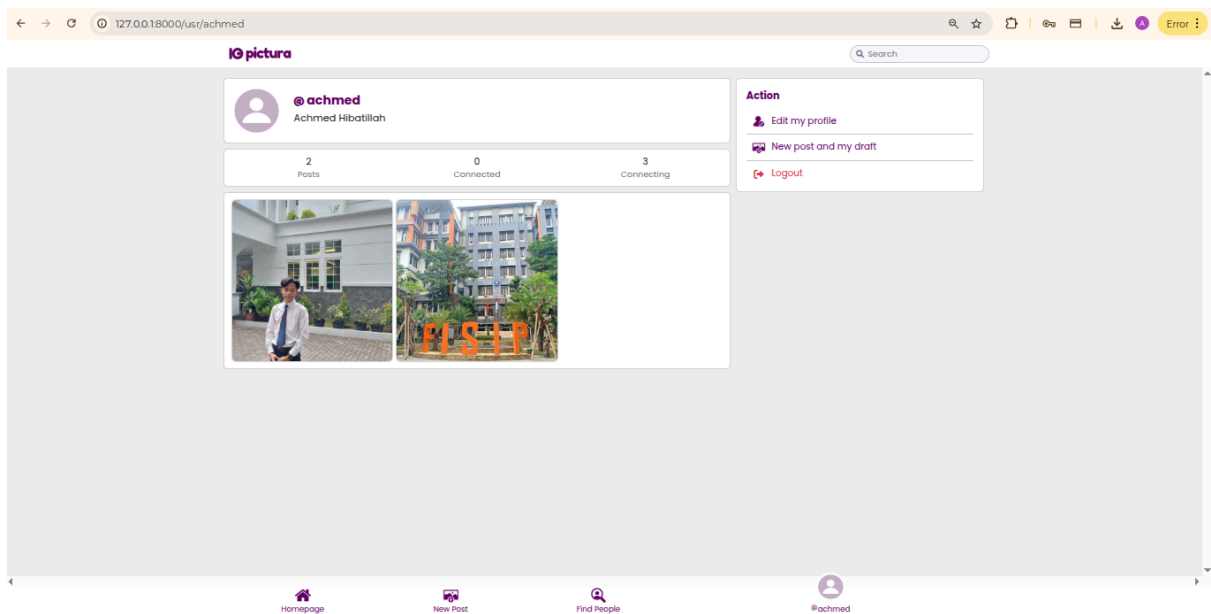
The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/register". The page features the "pictura" logo at the top. Below the logo is a "Create a new account" form. The form includes input fields for "Email", "Full Name", "Username", "Password", and "Confirm password". It also includes a "show password" toggle and a "Sign Up" button. At the bottom, there is a link "Already have an account? Log in here".

User baru dapat melakukan registrasi dengan mengisi form ini. Secara default, user baru akan berstatus sebagai user biasa (pengenalnya adalah user\_who = 3).

## 3. Halaman dashboard default

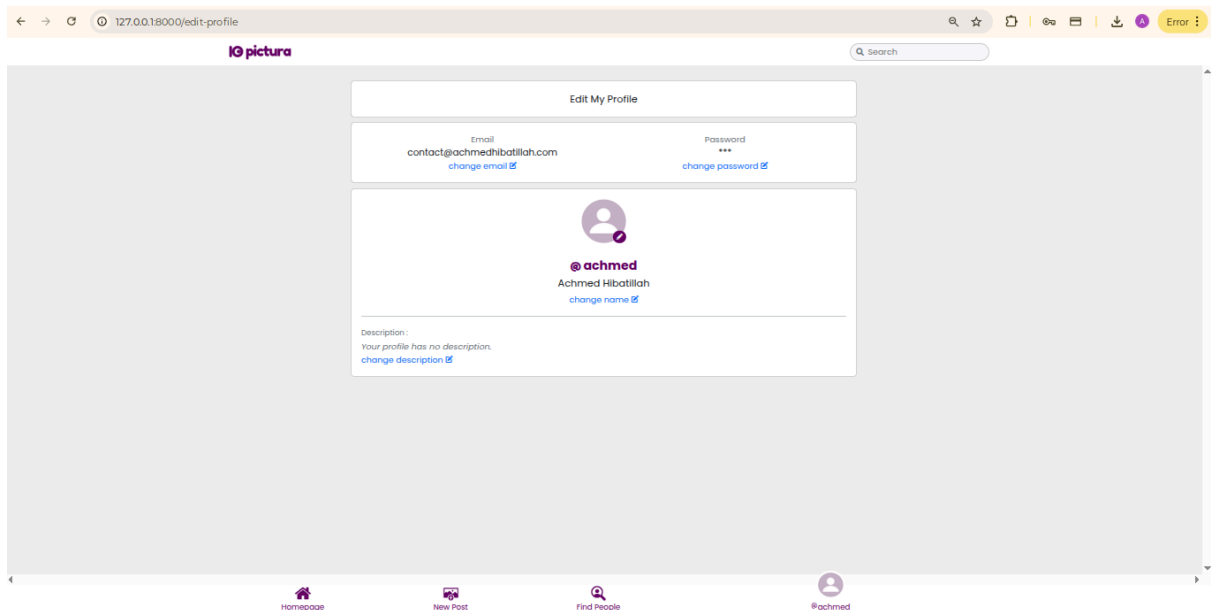


#### 4. Halaman profil default



Menampilkan profil user

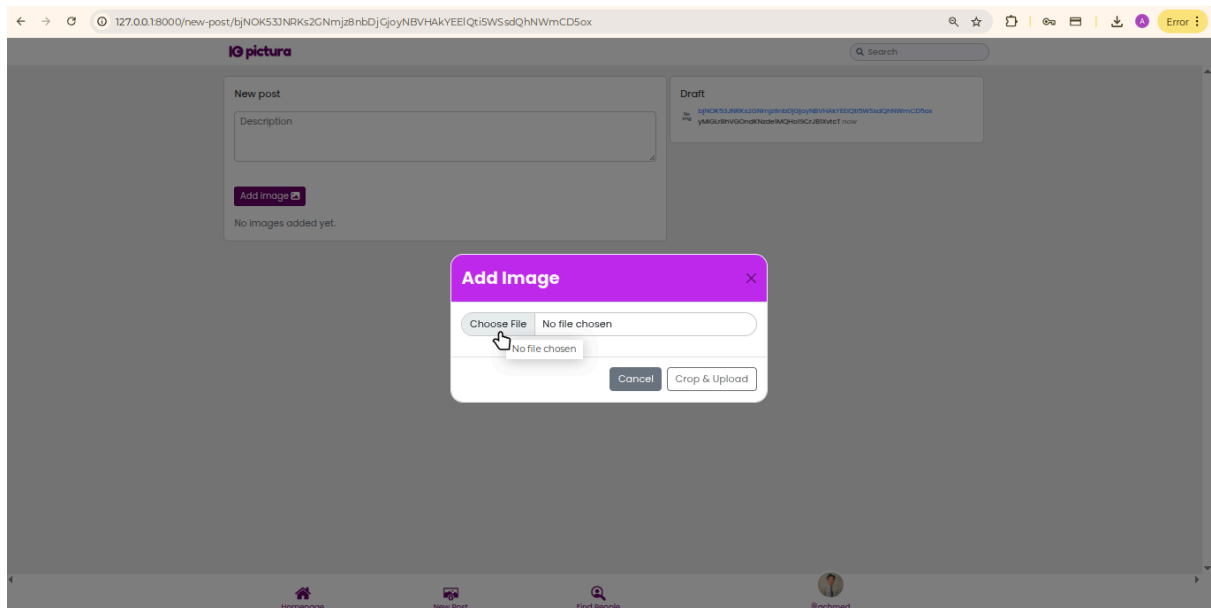
#### 5. Halaman edit profil



Mengedit informasi usernya sendiri, antara lain:

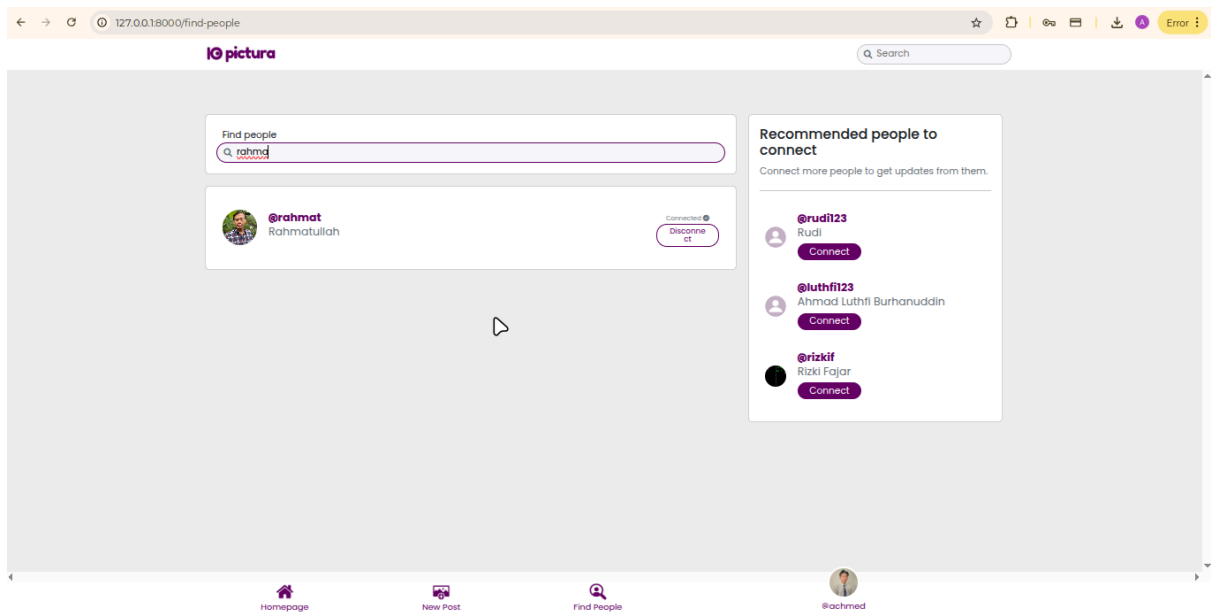
- a. email
- b. password
- c. full name
- d. username
- e. foto profil
- f. deskripsi

## 6. Halaman tambah postingan



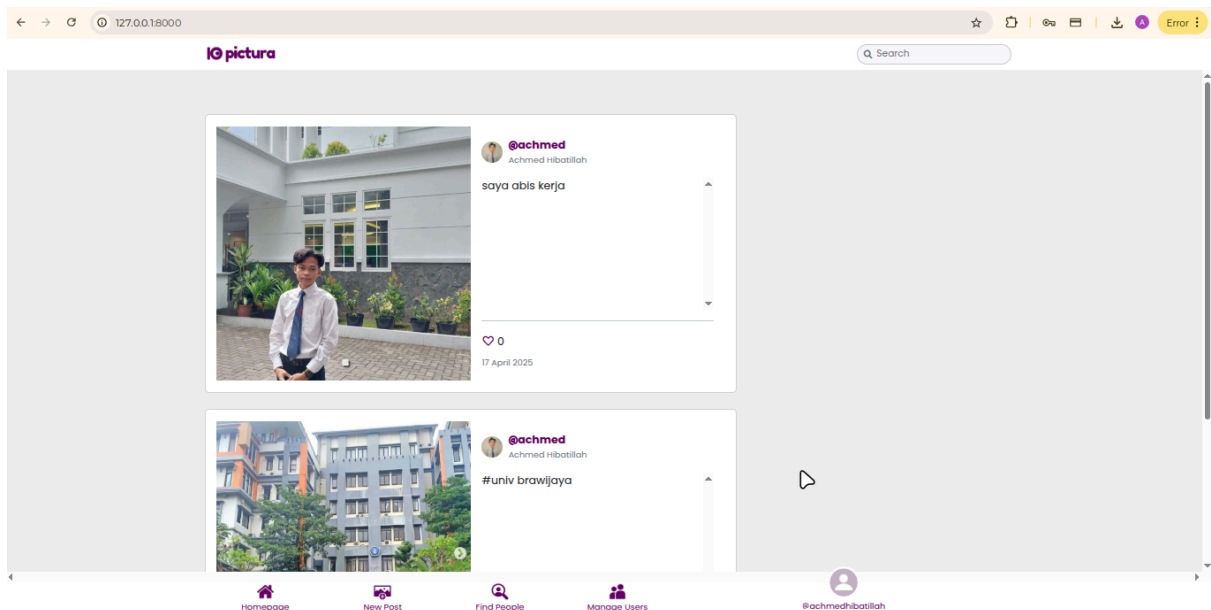
Memasukkan beberapa file image untuk diupload

## 7. Halaman find people



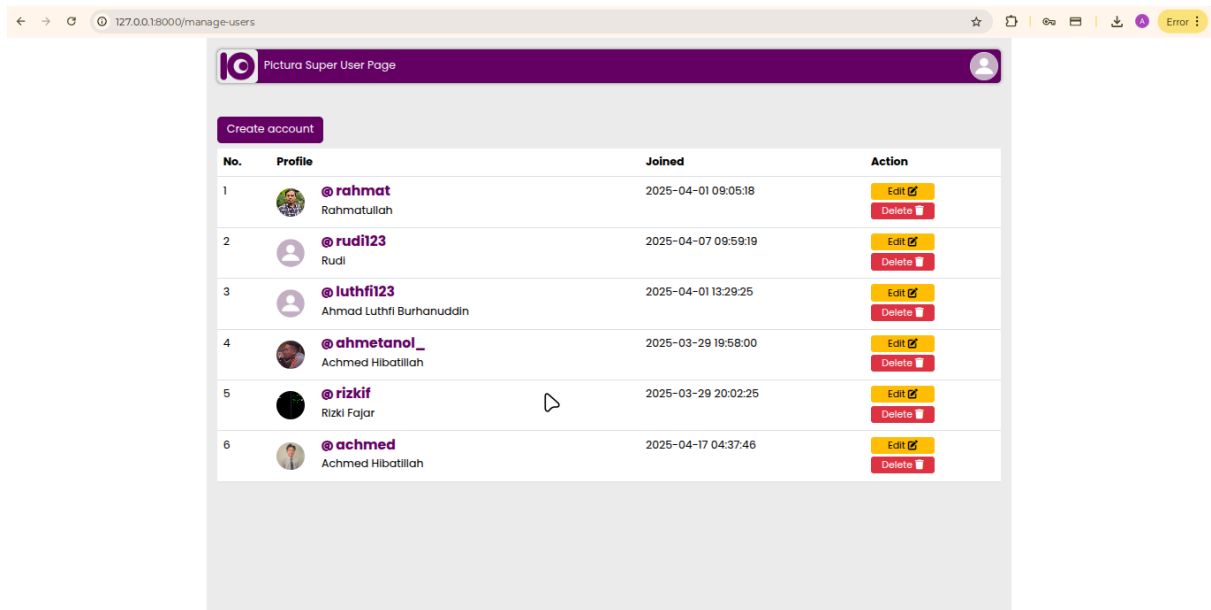
Melihat siapa aja user selain kita agar dapat melihat foto yang dibagikan, serta berkoneksi satu sama lain.

## 8. Halaman dashboard khusus untuk admin



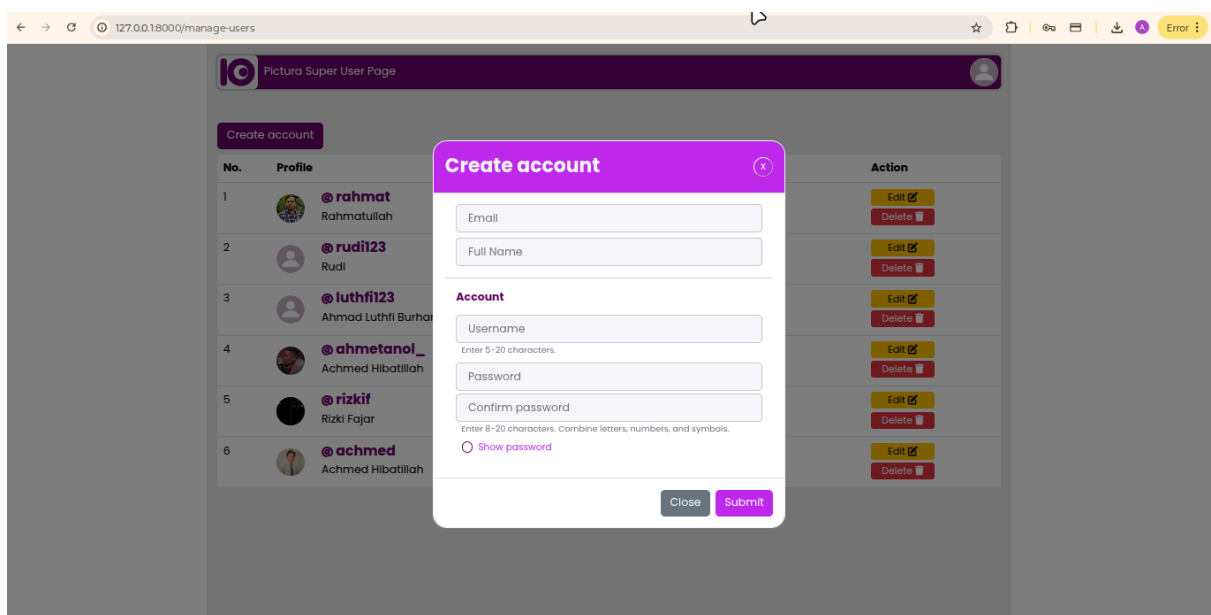
Terdapat empat kolom di navbar bawah, yang salah satunya adalah “Manage Users”

## 9. Halaman Manage Users (admin)

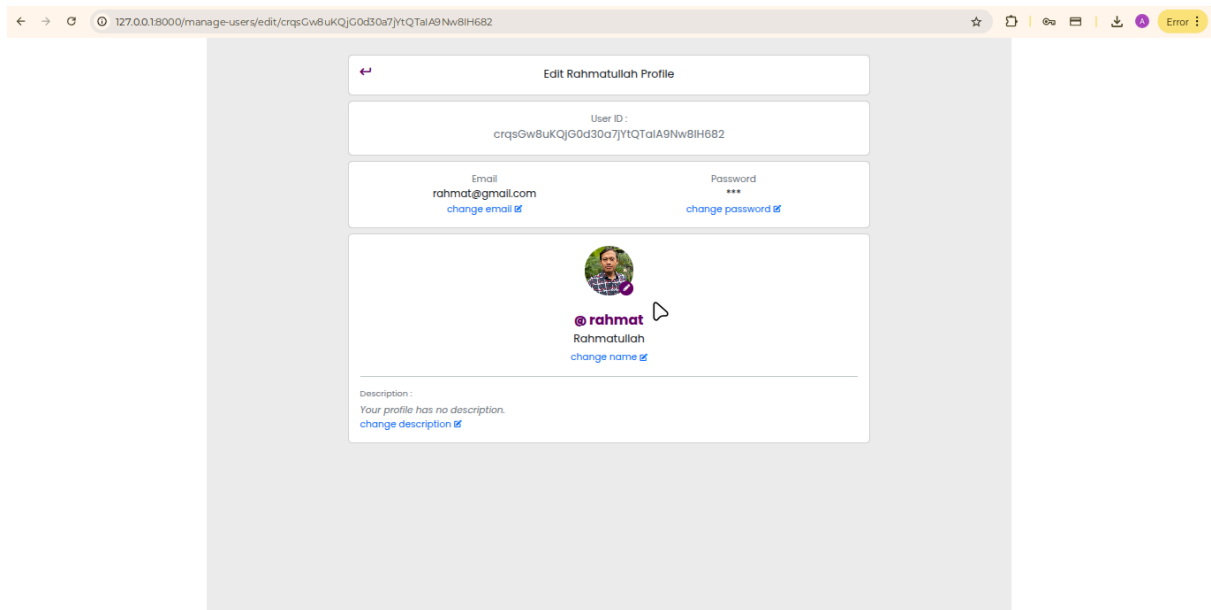


Admin dapat menambah, mengubah, hingga menghapus user yang ada dalam sistem.

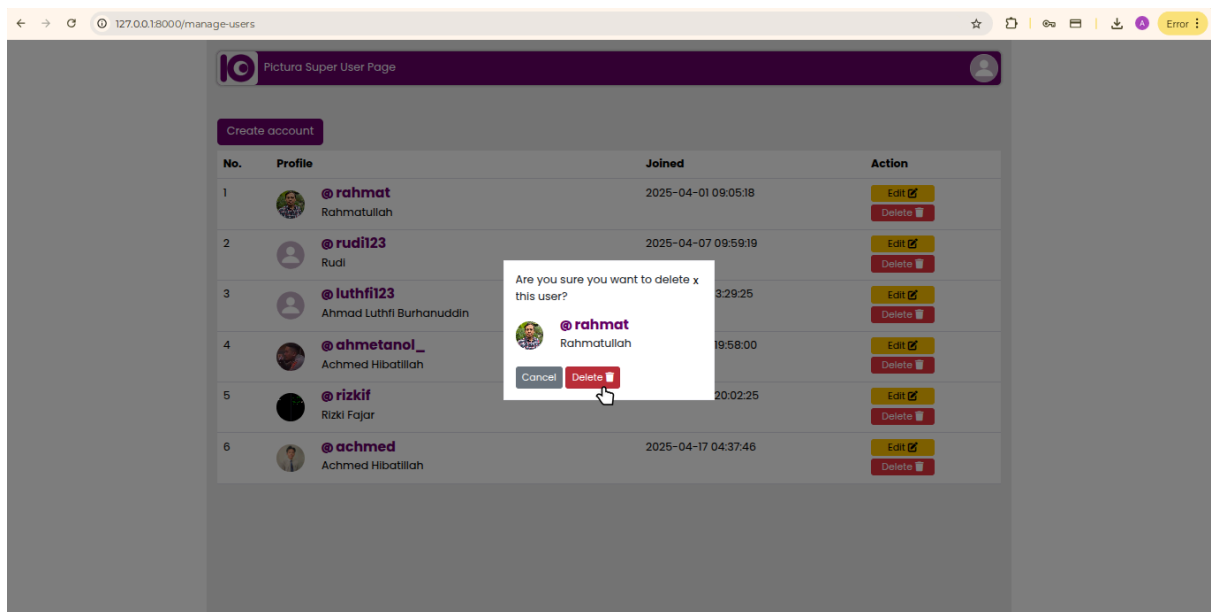
Tambah user baru:



Edit user:



Delete user:





## Source code :

The screenshot shows the GitHub repository page for 'pictura' by user 'achmedhibatillah'. The repository is public and has 1 branch and 0 tags. The file list shows the following structure:

File/Folder	Last Commit	Time Ago
app	Updated!	last week
bootstrap	First!	3 weeks ago
config	First!	3 weeks ago
database	Updated!	2 weeks ago
public	Updated!	last week
resources	Updated!	last week
routes	Updated!	last week
storage	First!	3 weeks ago
tests	First!	3 weeks ago
.editorconfig	First!	3 weeks ago
.env.example	First!	3 weeks ago

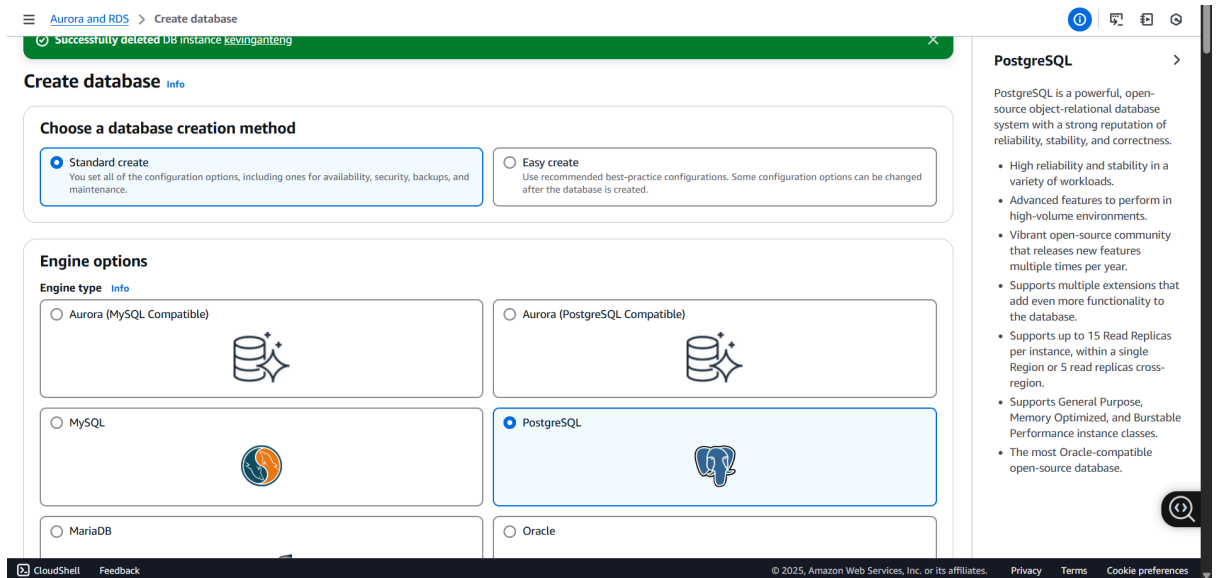
The repository description states: 'Pictura is a simple photo-sharing app built on AWS, allowing users to register, upload, and display photos on their profiles. It utilizes EC2 for hosting, RDS for the database, and S3 for image storage. Pictura ensures secure authentication, account management, and public profile views showcasing users' photo collections.'

Repository statistics: 0 stars, 1 watching, 0 forks.

<https://github.com/achmedhibatillah/pictura>

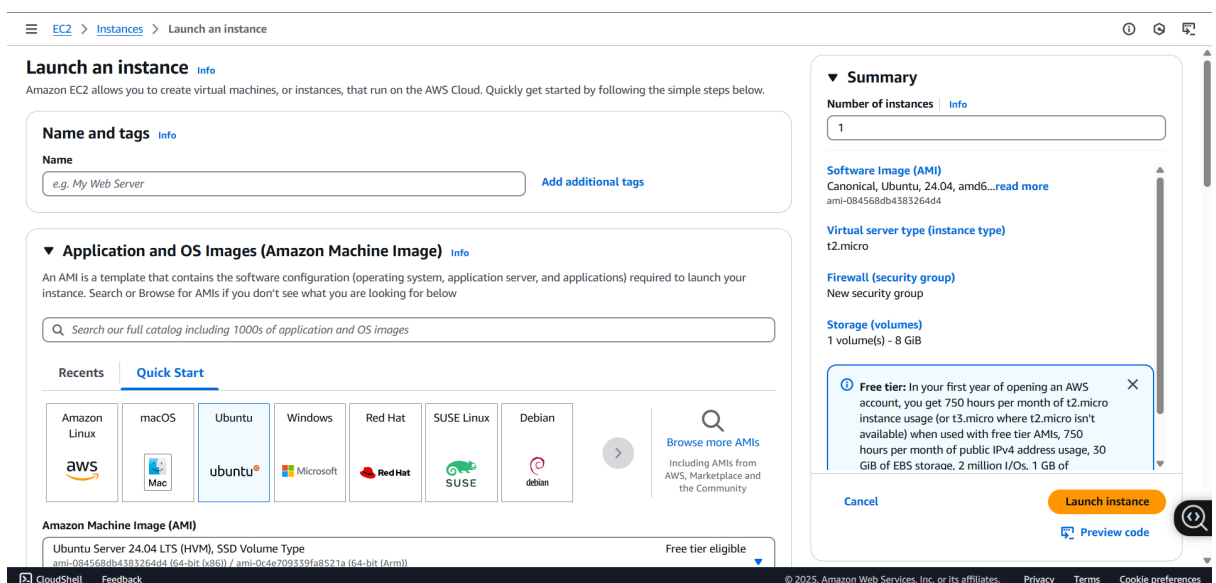
# Alur Instalasi AWS

## 1. Create RDS



Untuk membuat database RDS PostgreSQL di AWS, pertama-tama buka layanan Amazon RDS di AWS Management Console, lalu klik "Create database". Pilih metode "Standard create" agar bisa mengatur semua opsi konfigurasi secara manual. Setelah itu, pilih engine database **PostgreSQL**. Lanjutkan dengan mengisi detail seperti versi PostgreSQL, nama instance, username dan password, jenis instance, serta kapasitas penyimpanan. Selanjutnya, atur jaringan dan keamanan, termasuk VPC, subnet, dan security group, pastikan port 5432 terbuka jika ingin akses dari luar. Setelah semua konfigurasi selesai, klik "Create database" dan tunggu beberapa menit hingga database siap digunakan.

## 2. Create EC2 Instance



Untuk membuat instance EC2 di AWS, pertama-tama saya membuka layanan Amazon EC2 di AWS Management Console, lalu klik "Launch an instance". Di bagian "Name and tags", saya memasukkan nama untuk instance yang akan saya buat. Setelah itu, saya memilih sistem operasi dari daftar AMI (Amazon Machine Image), dalam hal ini saya memilih Ubuntu Server 24.04 LTS. Kemudian, saya memilih tipe instance t2.micro agar bisa menggunakan layanan Free Tier. Setelah itu, saya mengatur security group, misalnya dengan membuka port SSH (22) agar bisa remote ke server nanti. Setelah semua pengaturan selesai, saya klik tombol "Launch instance" di kanan bawah. Beberapa menit kemudian, instance saya sudah aktif dan siap digunakan.

### 3. Clone Repo Github ke EC2

```
ubuntu@ip-172-31-90-246:~$ cd /var/www
ubuntu@ip-172-31-90-246:/var/www$ ls
html  pictura
ubuntu@ip-172-31-90-246:/var/www$ cd pictura
ubuntu@ip-172-31-90-246:/var/www/pictura$ ls
README.md  artisan  composer.json  config  package.json  public  routes  tests  vite.config.js
app        bootstrap  composer.lock  database  phpunit.xml  resources  storage  vendor
```

Untuk meng-clone repository Pictura dari GitHub ke server EC2 saya yang menggunakan Ubuntu, pertama-tama saya pastikan Git sudah terinstall dengan menjalankan perintah `sudo apt update` lalu `sudo apt install git -y`. Setelah itu, saya masuk ke direktori tempat saya ingin menyimpan project, misalnya dengan perintah `cd /var/www`. Kemudian, saya menjalankan perintah `git clone <URL-repository>` untuk meng-clone repository Pictura. Setelah proses clone selesai, sebuah folder baru dengan nama project akan muncul di dalam `/var/www`. Saya lalu masuk ke folder tersebut menggunakan `cd pictura` untuk mulai mengatur atau menjalankan aplikasinya. Sebelum lanjut, saya juga menjalankan perintah seperti `composer install`.

### 4. Setup .env

```
GNU nano 7.2 .env
APP_MAINTENANCE_DRIVER=file
# APP_MAINTENANCE_STORE=database

PHP_CLI_SERVER_WORKERS=4

BCRYPT_ROUNDS=12

LOG_CHANNEL=stack
LOG_STACK=single
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=pgsql
DB_HOST=pg-pictura.ci2sgtszgs8v.us-east-1.rds.amazonaws.com
DB_PORT=5432
DB_DATABASE=postgres
DB_USERNAME=postgres
DB_PASSWORD=pgsqlpictura

SESSION_DRIVER=database
SESSION_LIFETIME=120
SESSION_ENCRYPT=false
SESSION_PATH=/
SESSION_DOMAIN=null

BROADCAST_CONNECTION=log

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location  M-U Undo     M-A Set Mark
^X Exit      ^R Read File ^N Replace   ^U Paste     ^J Justify  ^_ Go To Line M-E Redo     M-G Copy
```

Setelah saya berhasil meng-clone repository Laravel bernama Pictura ke dalam direktori `/var/www/pictura` di Ubuntu server EC2, langkah selanjutnya yang saya lakukan adalah menyiapkan file konfigurasi environment. Saya menyalin file `.env.example` menjadi `.env` dengan perintah `cp .env.example .env`, lalu saya generate application key menggunakan perintah `php artisan key:generate`. Setelah itu, saya mengatur permission direktori agar server dapat menjalankan Laravel dengan baik. Saya memberikan kepemilikan folder kepada user `www-data` dan memberikan permission `775` untuk folder `storage` dan `bootstrap/cache` agar dapat ditulis oleh server. Kemudian, saya menjalankan `composer install` untuk mengunduh semua dependensi PHP yang dibutuhkan oleh Laravel. Karena saya tidak menggunakan `npm`, proses instalasi hanya terbatas pada Composer saja. Terakhir, saya mengedit file `.env` untuk menyesuaikan konfigurasi database dan informasi lainnya, seperti nama database, username, password, dan host agar terhubung dengan database PostgreSQL yang telah saya buat sebelumnya.

## 5. Instalasi dependency

Setelah berhasil mengakses server EC2 saya melalui SSH, langkah pertama yang saya lakukan adalah memperbarui sistem dengan menjalankan `sudo apt update && sudo apt upgrade -y` untuk memastikan semua paket berada dalam versi terbaru. Setelah itu, saya menginstal Nginx menggunakan perintah `sudo apt install nginx -y`. Nginx akan berfungsi sebagai web server yang menangani request ke aplikasi Laravel saya. Selanjutnya, karena Laravel membutuhkan PHP, saya menambahkan repository resmi dari *Ondřej Surý* untuk mendapatkan versi PHP terbaru, yaitu PHP 8.2, dengan perintah `sudo add-apt-repository ppa:ondrej/php -y && sudo apt update`. Setelah itu, saya menginstal PHP 8.2 beserta modul-modul yang dibutuhkan Laravel seperti `php8.2-cli`, `php8.2-common`, `php8.2-fpm`, `php8.2-mysql`, `php8.2-xml`, `php8.2-mbstring`, `php8.2-curl`, `php8.2-zip`, dan `php8.2-bcmath` menggunakan satu perintah: `sudo apt install php8.2 php8.2-fpm php8.2-mysql php8.2-xml php8.2-mbstring php8.2-curl php8.2-zip php8.2-bcmath -y`. Saya juga menginstal Composer, dependency manager untuk PHP, dengan mengikuti perintah resmi dari [getcomposer.org](https://getcomposer.org). Composer ini penting karena digunakan untuk menginstal dan mengelola library Laravel. Setelah semua terpasang, saya memastikan PHP-FPM berjalan dengan baik menggunakan `sudo systemctl status php8.2-fpm`, dan memulai service Nginx dengan `sudo systemctl start nginx`. Dengan semua dependency ini, server saya siap untuk menjalankan aplikasi Laravel dengan lancar.

## 6. Konfigurasi nginx



```
GNU nano 7.2 laravel
server {
    listen 80;
    server_name _;

    root /var/www/pictura/public;
    index index.php index.html;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php8.2-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.ht {
        deny all;
    }
}
```

Saya mengonfigurasi server Nginx untuk menangani permintaan HTTP di port 80 dengan root direktori di `/var/www/pictura/public`. Ketika pengunjung mengakses situs, Nginx akan mencari file di direktori itu dan jika tidak ada, akan mengarahkannya ke `index.php`. Untuk permintaan PHP, Nginx akan mengirimkan ke PHP-FPM untuk dieksekusi. Saya juga melarang akses ke file yang diawali dengan `.ht` demi keamanan.

Untuk memindahkan konfigurasi ini, saya buat file konfigurasi di `/etc/nginx/sites-available/` dengan nama `pictura`, lalu membuat symlink ke `/etc/nginx/sites-enabled/` agar Nginx bisa membaca dan mengaktifkannya. Setelah itu, saya cek konfigurasi dengan `nginx -t` dan jika tidak ada error, saya reload Nginx dengan `systemctl reload nginx` agar perubahan berlaku.

## 7.



Alur