Pengembangan *Backend* Untuk Aplikasi Monitoring dan Kontrol *Air Purifier* Berbasis IoT dan Sistem Cerdas

1st Raken Putra Athallah
Prodi S1 Teknik Komputer
Fakultas Teknik Elektro
Universitas Telkom
Bandung,Indonesia
rakenputra@student.telkomuniversity.ac.id

2nd Astri Novianty
Dosen Teknik Komputer
Fakultas Teknik Elektro
Universitas Telkom
Bandung,Indonesia
astrinov@telkomuniversity.ac.id

3rd Faisal Candrasyah Hasibuan
Dosen Teknik Komputer
Fakultas Teknik Elektro
Universitas Telkom
Bandung,Indonesia
faicanhasfcb@telkomuniversity.ac.id

***Abstrak* — Udara bersih semakin sulit ditemukan akibat polusi, terutama di kota-kota besar. Polutan seperti partikel PM, ozon, dan karbon monoksida dapat berdampak buruk pada kesehatan manusia. Polusi udara dapat berasal dari faktor alami seperti debu dan letusan gunung berapi, serta aktivitas manusia seperti transportasi, industri, dan pembangkit listrik. Penelitian ini bertujuan untuk mengembangkan sistem air purifier berbasis IoT dan sistem cerdas yang dapat dikontrol dan dimonitor melalui aplikasi mobile. Sistem ini memungkinkan pengguna untuk memantau kualitas udara dalam ruangan dan mengontrol air purifier melalui database yang menyimpan data sensor dan kontrol. Berdasarkan hasil stress testing, ditemukan bahwa database memiliki keterbatasan dalam layanan pembacaan dan penulisan data, yang berpotensi mengganggu implementasi IoT. Air purifier ini dirancang menggunakan fuzzy logic untuk meningkatkan efisiensi penggunaan daya, membutuhkan daya maksimal sebesar 4.7W, dan mampu memurnikan udara hingga 92.5% pada ruangan 3.5m x 2.5m. Suara yang dihasilkan oleh air purifier ini tidak melebihi 61dB.**

Kata kunci— Internet of Things, Air Purifier, Sistem Cerdas, Aplikasi mobile, Firebase Firestore

I. PENDAHULUAN

Kualitas udara yang baik sangat penting untuk kesehatan manusia dan makhluk hidup lainnya.Namun, dengan meningkatnya aktivitas manusia dan aktivitas industri, polusi udara telah menjadi masalah dibanyak kota besar di Indonesia. Polutan seperti partikel PM, ozon dan karbon monksida tidak hanya memperburuk kualitas udara, tetapi juga berdampak pada negatif pada kesehatan manusia, temasuk penyakit pernapasan, kardioviskular, kanker paruparu dan gangguan kesehatan lainnya [1].

Sumber polusi udara dapat dibagi menjadi dua kategori yaitu alami dan antropogenik. Sumber alami termasuk debu dari tanah, letusan gunun berapi dan kebakaran hutan. Sedangkan sumber antropogenik meliputi emisi dari kendaraan bermotor, pabrik, pembangkit listrik dan aktivitas industri lainnya [2]. Kombinasi dari kedua sumber ini menyebabkan peningkatan konsentrasi polutan di udara yang dapat menurunkan kualitas udara di lingkungan perkotaan dan pedesaan.

Untuk mengatasi masalah ini, berbagai teknologi dan solusi telah dikembangkan. Salah satu solusi yang efektif adalah penggunaan *air purifier*, alat yang dirancang untuk membersihkan udara dalam ruangan dari polutan berbahaya. Dengan kemajuan teknologi, sistem *air purifier* kini dapat diintegrasikan dengan teknologi *Internet of Things* (IoT) dan kecerdasan buatan (AI), memungkinkan kontrol dan pemantauan kualitas udara secara *real-time* melalui aplikasi *mobile*.

Penelitian ini bertujuan untuk mengembangkan sistem *air purifier* berbasis IoT dan sistem cerdas yang dapat dikontrol dan dimonitor melalui aplikasi *mobile*. Sistem ini dirancang untuk memberikan solusi efektif dalam meningkatkan kualitas udara dalam ruangan, khususnya di area perkotaan dengan tingkat polusi tinggi. Penggunaan Firebase Firestore sebagai *backend* memungkinkan penyimpanan dan pengelolaan data secara efisien, serta menyediakan fitur monitoring *real-tim*e dan kontrol jarak jauh.

Melalui pengembangan sistem ini, diharapkan pengguna dapat lebih mudah mendapatkan udara bersih dan memantau kualitas udara di dalam rumah mereka, sehingga dapat mengurangi risiko kesehatan akibat polusi udara.

II. KAJIAN TEORI

Pengembangan aplikasi *mobile* untuk *air purifier* bertujuan untuk memberikan kemudahan kepada pengguna dalam memantau kualitas udara dan mengendalikan perangkat *air purifier* melalui aplikasi. Aplikasi ini dirancang untuk memungkinkan pengguna melakukan pemantauan dan kontrol dari berbagai sudut rumah, menjadikannya lebih praktis dan fleksibel. Firestore dari Firebase digunakan sebagai basis data untuk menyimpan data sensor dari *air purifier*, nilai kontrol dari aplikasi, serta informasi pengguna seperti nama pengguna dan kata sandi. *Framework* Flutter digunakan untuk membangun antarmuka pengguna (UI) aplikasi *mobile*, menyediakan pengalaman yang intuitif dan mudah digunakan. Dengan memanfaatkan teknologi ini, penerapan *Internet of Things* (IoT) pada sistem *air purifier* menjadi nyata, memungkinkan pemantauan kualitas udara dan pengendalian perangkat secara efisien dan terintegrasi.

A. *Internet of Things*

Internet of Things (IoT) adalah konsep di mana berbagai perangkat seperti sensor, perangkat elektronik dan objek lainnya dapat terhubung dan berkomunikasi melalui jaringan Intenet [3]. Dalam pengembangan *air purifier*, IoT memungkinkan perangkat untuk beroperasi secara lebih cerdas dengan memantau kualitas udara secara real-time, mengendalikan fungsinya dari jarak jauh, dan mengirimkan data ke *aplikasi mobile*. Sensor yang terintegrasi dalam air purifier mengukur parameter kualitas udara seperti konsentrasi partikel dan kadar karbon dioksida, kemudian mengirimkan data ini ke platform berbasis cloud melalui internet. Pengguna dapat memantau informasi terkini tentang kualitas udara melalui aplikasi *mobile*, serta mengatur berbagai pengaturan perangkat seperti mode operasi dan kecepatan kipas dari jarak jauh. Selain itu, data yang dikumpulkan dapat digunakan untuk integrasi dengan sistem rumah pintar lainnya, menciptakan ekosistem yang lebih terhubung dan responsif. Dengan demikian, IoT tidak hanya meningkatkan efisiensi dan efektivitas pengelolaan kualitas udara tetapi juga memberikan pengalaman pengguna yang lebih baik dan nyaman.

B. Flutter

Flutter merupakan platform yang digunakan *developer* untuk membuat aplikasi *multiplatform* hanya dengan satu basis *coding (codebase)*.Artinya aplikasi yang dihasilkan dapat dipakai di berbagai *platform*, baik *mobile* Android, iOS, *web*, dan *desktop*[4]. *Framework* ini dikembangkan oleh Google dan dikenal karena kemampuannya untuk menghasilkan antarmuka pengguna yang konsisten dan responsif di semua *platform*. Flutter menggunakan bahasa pemrograman Dart dan menyediakan serangkaian *widget* yang dapat disesuaikan untuk membangun *UI* yang menarik dan fungsional. Dalam konteks pengembangan aplikasi *air purifier*, Flutter memungkinkan pembuatan antarmuka yang intuitif dan mudah digunakan, sehingga pengguna dapat memantau kualitas udara secara *real-time* dan mengontrol *air purifier* secara efisien. Dengan kemampuannya untuk menyederhanakan pengembangan lintas platform, Flutter mempermudah proses pengembangan dan pemeliharaan aplikasi sambil memastikan performa yang optimal di berbagai perangkat.

C. Firebase Firestore

Firebase Firestore adalah layanan *database* NoSQL yang disediakan oleh Google, dirancang untuk menyimpan dan menyinkronkan data secara *real-time* dengan efisiensi tinggi. Firestore menyimpan data dalam bentuk dokumen yang terorganisir dalam koleksi, memungkinkan struktur data yang fleksibel dan mudah diatur. Berbeda dengan Firebase Realtime Database, Firestore menawarkan berbagai keunggulan seperti kemampuan *query* yang lebih kuat dan fleksibel, serta skalabilitas yang lebih baik. Dengan Firestore, pengembang dapat melakukan pencarian yang lebih kompleks dan mendapatkan data dengan lebih cepat berkat dukungan fitur *indexing* otomatis. Indeks ini secara otomatis dibuat untuk setiap *field* yang digunakan dalam *query*, sehingga meningkatkan performa pencarian dan akses data. Firestore juga menawarkan dukungan untuk transaksi dan operasi *batch*, yang mempermudah pengelolaan data secara konsisten. Fitur-fitur ini menjadikan Firestore pilihan yang ideal untuk aplikasi yang memerlukan sinkronisasi data *realtime* dan akses data yang efisien, seperti dalam aplikasi *air purifier* yang memerlukan pemantauan dan pengendalian data secara dinamis.

D. Firebase Authentication

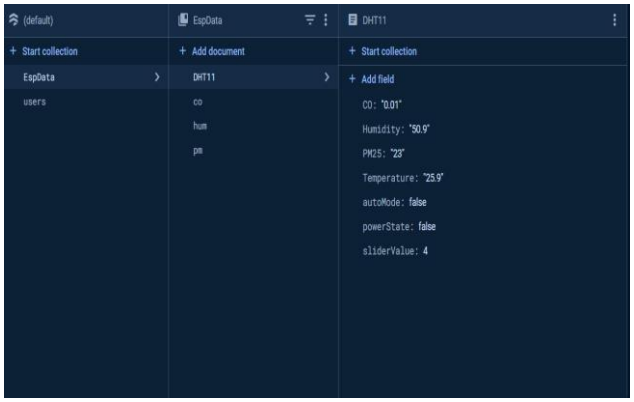
Firebase Authentication berperan penting dalam aplikasi *mobile* yang dikembangkan untuk sistem *air purifier* ini dengan menyediakan mekanisme autentikasi yang aman dan efisien untuk pengguna. Firebase Authentication memungkinkan pengguna untuk mendaftar, masuk, dan mengelola akun mereka dengan berbagai metode otentikasi, seperti email dan kata sandi, nomor telepon, atau akun penyedia identitas pihak ketiga seperti Google. Dengan integrasi Firebase Authentication, aplikasi ini memastikan bahwa hanya pengguna yang terautentikasi yang dapat mengakses fitur kontrol dan *monitoring* pada air purifier. Firebase Authentication memiliki fitur keamanan seperti, enkripsi data dan perlindungan terhadap serangan *brute force*, menjaga data pengguna tetap aman dari potensi ancaman. Selain itu, antarmuka sederhana untuk pengelolaan akun mempermudah proses pendaftaran dan pemeliharaan akun, memungkinkan pengguna untuk dengan mudah mengakses dan mengontrol perangkat air purifier mereka. Firebase Authentication tidak hanya meningkatkan keamanan dan kenyamanan pengguna tetapi juga memastikan bahwa aplikasi dapat menyediakan pengalaman pengguna yang konsisten dan terjamin.

III. METODE

A. Rancagan pengintregasian *air purifier* dengan aplikasi *mobile*

Untuk mengimplementasikan *Internet of Things* pada sistem *air purifer*. Sistem harus diintegrasikan agar dapat bekerja berdampingan seperti yang digambarakan dengan rancangan sederhana pada Gambar 3.1. *Database* akan berisikan *value* dari sensor untuk melakukan *monitoring* secara *real-time* melalui aplikasi yang didapatkan dari sensor pada *air purifier*. Dan terdapat value bertipe data *boolean* untuk kontrol air purifer melalui aplikasi *mobile*. Database juga akan menyimpan data pengguna pada koleksi berbeda agar hanya pengguna yang telahn mendaftarkan akun yang dapat mengakses kontrol dan *monitor air purifer*.Struktur dari database yang digunakan untuk pada pengembangan sistem air purifier ini dapat dilihat pada Gambar 3.2

Gambar 3. 1. Rancangan sederhana peningtegrasian sistem *air purifier*



Gambar 3. 2 Struktur *database* pada Firestore Database

B. Proses pengambilan data untuk melakukan *montirong* melalui aplikasi *mobile*

Untuk membangun UI yang dapat mengambil aliran(*stream*) data, diperlukan StreamBuilder untuk mendapatkan sumber data yang diinginkan pada kasus ini adalah firestore database. Diperlukan nama dari *collection* dan *document* yang mengarah ke data yang akan ditampilkan di dalam UI. Kemudian, data akan diperiksa menggunakan 'if (!snapshot.hasData)'. Jika data tidak ada, fungsi akan dikembalikan ke 'return const CircularProgressIndicator();' yang menunjukkan bahwa data sedang dimuat. Jika data tersebut ada dalam document, maka 'snapshot.data!.data()' akan digunakan untuk mengakses data dari dalam document. Selanjutnya, data dari *field* yang ingin digunakan. Metode pengambila data pada semua sensor untuk mealukan *monitoring* menggunakan metode yang sama hanya saja menggunakan *field* yang berbeda sesuai dengan kebutuhan nilai yang ingin diambil. Contoh dibawah ini merupakan contoh untuk pengambilan data pada field temperatur.

```
StreamBuilder<DocumentSnapshot>(
stream: FirebaseFirestore.instance
       .collection('EspData')
       .doc('DHT11')
       .snapshots(),
builder: (context, snapshot)
if (!snapshot.hasData) {
return const CircularProgressIndicator();
}
if(snapshot.hasData&&snapshot.data!.exist)
{
Map<String, dynamic>? data = snapshot.data!
 .data() as Map<String, dynamic>?; if
       (data != null) {
         String tempValue =
          data['Temperature'] ?? '0.0';
       double temperature =
        double.tryParse(tempValue) ?? 0;
```

C. Proses memperbarui data untuk kontrol *air purifier*

Untuk mengontrol *air purifier* melalui aplikasi *mobile*, proses ini melibatkan pembaruan data di *Firestore* dan pengelolaan interaksi pengguna di antarmuka aplikasi. Dalam aplikasi, berbagai kontrol, seperti mode otomatis atau tombol *power*, diatur dengan menggunakan *query update* di Firestore. Ketika pengguna melakukan interaksi, seperti mengetuk tombol, aplikasi memperbarui nilai *boolean* yang sesuai dalam database, memungkinkan perubahan status seperti mode otomatis atau fungsi lainnya. Selain itu, sifat dan fungsionalitas dari kontrol di aplikasi *mobile* disesuaikan dengan status data *boolean* yang ada di Firestore. Sebagai contoh, jika nilai *power* perangkat dalam database menunjukkan bahwa perangkat mati, kontrol yang bergantung pada status *power*, seperti mode otomatis, tidak akan aktif atau dapat digunakan. Ini memastikan bahwa kontrol yang diterapkan selalu konsisten dengan status perangkat yang sebenarnya dan memberikan umpan balik yang akurat kepada pengguna.Contoh dibawah ini adalah contoh pada tombol mode dan karakteristiknya seperti menampiilkan *slider* kecepatan kipas jika nilai mode berupa *false* untuk kontrol *air purifier* melalui aplikasi mobile

```
onTap: () {  if
(!_isPowerOn) {
  return; // Do nothing if power is off
 }
 setState(() {
  isAutoMode = !_isAutoMode;
   _showSlider =
  !_isAutoMode;//Toggleslidervisibility
});
    FirebaseFirestore.instance
collection('EspData')
      .doc('DHT11') .update({'autoMode':
_isAutoMode}).then((_)=>print('Modeupda
ted successfully'))
.catchError((error) =>
 print('Failed $error')); },
```

D. Menampilkan Statistik Kualitas Udara

Statistik berguna agar pengguna dapat memantau kualitas udara selama alat menyala. Dengan fitur ini pengguna dapat memantau kualitas udara di dalam rumahnya dan kinerja dari *air purifier*. Sama dengan pengambilan data pada sensor penampilan statistik menggunakan metode pengambilan data yang sama dengan menggunakan '*streambuilder'*. Namun kali ini dtiampilkan ke dalam *widget* yang dapat menampilkan grafik bernama 'sfCartesianChart' .Contoh dibawah ini adalah contoh penampilan statistik untuk menampilkan kadar CO

```
return StreamBuilder(
stream:FirebaseFirestore.instance
.collection('EspData').snapshots(),
builder:(context,AsyncSnapshot<QuerySna
pshot> snapshot)
 { if (!snapshot.hasData)
{ return const Center(
child: CircularProgressIndicator(
valueColor:AlwaysStoppedAnimation<Color
>(Colors.red),
  ), );
} else {
var              provider              =
Provider.of<COData>(context,      listen:
false);
var documents =snapshot.data?.docs ??[];
for (var f in documents) {  if (f.id ==
'DHT11') {
  print('current CO = ${f['CO']}');
provider.setGlobalCurrentSensorValue(do
uble.parse(f['CO']));
                    }
                  }
```

E. Pengelolaan data pengguna

Dalam proses pendaftaran pengguna, aplikasi memanfaatkan Firebase Authentication untuk mengelola akun pengguna secara aman. Saat pengguna mendaftar dengan email dan password melalui metode *'createUserWithEmailAndPassword'*, Firebase Authentication secara otomatis menyimpan data autentikasi pengguna, termasuk email dan password. Setelah pengguna berhasil terdaftar, data username dan email disimpan di Firestore berdasarkan uid mereka masing-masing. Dengan demikian setiap pengguna baru akan memiliki *document* berisikan email dan *username* mereka. Ketika membuat akun baru terdapat beberapa aturan yang diberikan kepada pengguna seperti tidak diperbolehkan menggunakan email yang sama dan password tidak boleh dibawah dari enam kata. Pengguna dapat mengetahui ketentuan-ketentuan tersebut dengan memberikan *error handling* seperti pop-up peringatan ketika membuat akun dengan email yang sama atau ketika membuat *password* yang kurang dari enam huruf. *Error handling* tersebut diharapkan dapat membantu aksesibilitas pengguna dan kenyamanan pengguna ketika membuat akun. Dibawah ini adalah metode yang digunakan untuk mengelola akun pengguna

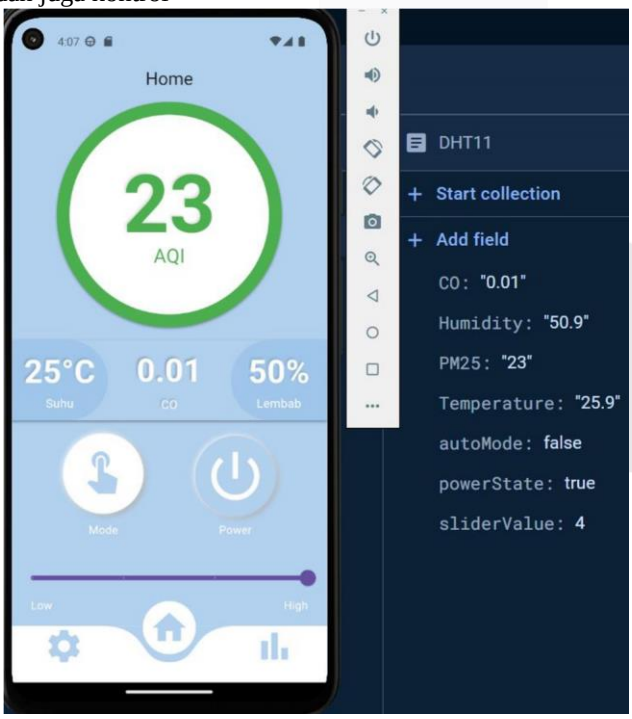
```
void _signUp() async {
String username = _usernameController.text;
String email = _emailController.text;
String password =
_passwordController.text;   if
(password.length < 6) {     setState(() {
       _isPasswordValid = false;
     });
return;
}  try {
  UserCredential userCredential = await
FirebaseAuth.instance.createUserWithEmailA
ndPassword(   email: email,   password:
password,
 );
 User? user = userCredential.user; if
(user != null) {  print("User created
successfully: $user");
FirebaseFirestoredb=FirebaseFirestore.inst
ance;
String     uid     =     user.uid;     await
db.collection('users').doc(uid).set({
 'username': username,
 'email': email,});
print("User     document     created     in
Firestore"); try {  await
_cloneCollection(uid);
 } catch (e) {
 print("Error cloning collection: $e");
              }
print("User     created     and     Firestore
collection initialized for user: $uid");
Navigator.pushNamed(context, "/bottomnav");
} else { print("User registration
failed");
}
```

```
} on FirebaseAuthException catch (e) {
if (e.code == 'email-already-in-use') {
print("Email is already in use");
 _showErrorDialog("Email already used.");
} else {
print("Error creating user: $e");
_showErrorDialog("An error occurred while
creating the account. Please try again.");
  }
} catch (e) {  print("Error
creating user: $e");
_showErrorDialog("An error occurred while
creating the account. Please try again.");
 }
}
```

IV. HASIL DAN PEMBAHASAN

A. Pengintegrasian Aplikasi *Mobile*

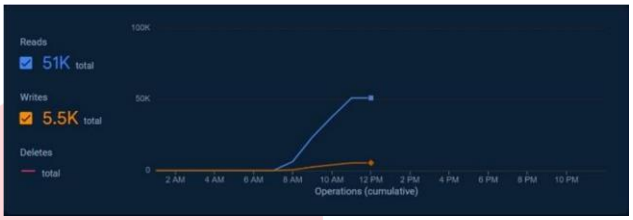
Pengujian dilakukan dengan dengan melakukan *alpha testing* pada aplikasi *mobile*, dengan memantau value di dalam database dapat terintegrasikan dengan aplikasi *mobile* apakah data atau state pada button yang ditampilkan pada aplikasi mobile telah sesuai dengan nilai yang ada di dalam *database*. Dapat dilihat dari Gambar 4.1 aplikasi telat terintergrasi ke *database* untuk dapat melakukan *monitoring* dan juga kontrol



Gambar 4. 1 hasil pengintegrasian aplikasi *mobile* dengan database

B. Stress Testing Database

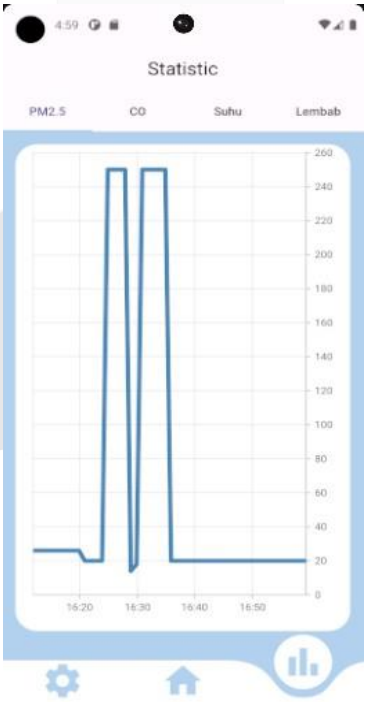
Stress testing database dilakukan untuk mengevaluasi kapasitas dan kemampuan *database* dalam menangani beban kerja yang tinggi. Dalam pengujian ini, *air purifier* dinyalakan secara terus-menerus untuk memastikan bahwa data terus-menerus dikirim ke *database*. Setelah beberapa jam, *database* mencapai batas maksimum untuk operasi baca dan tulis yang diizinkan oleh layanan Firebase berdasarkan berapa banyak data yang dimasukkan setiap detiknya. Hal ini terjadi karena *database* yang digunakan adalah versi gratis dari Firestore, yang memiliki batasan tertentu dalam hal kapasitas dan kinerja. Berdasarkan grafik pada Gambar 4.2 dapat dilihat jika terjadi pelunjakkan untuk fungsi *reading* hingga 50 ribu yang dimana merupakan batasan yang telah ditentukan untuk layanan *database* yang sedang digunakan saat ini.



Gambar 4. 2 hasil pengujian *stress testing database*

C. menampilkan data statistik

Pengujian statistik dilakukan dengan menyalakan *air purifier*. Dengan demikian data pada *database* akan berubah berdasarkan data yang diberikan sensor dari *air purifier* dapat dilihat dari Gambar 4.3 halaman statistik dapat berubah berdasarkan dengan perubahan yang terjadi pada *database* dan data sebelumnya dapata terekam dalam bentuk grafik sehingga pengguna dapat mengetahui kualitas udara di dalam rumahnya selama *air purifier* menyala

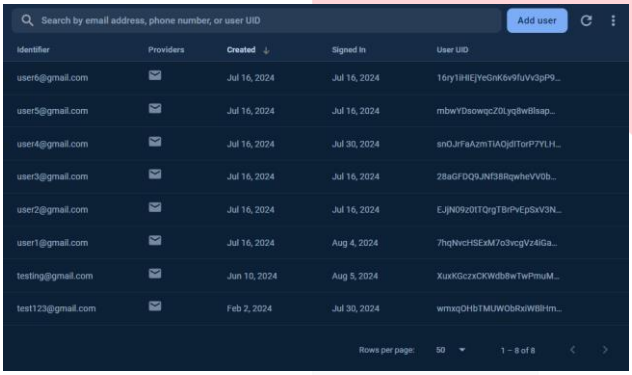


Gambar 4. 3 hasil pengujian menampilkan data statistik

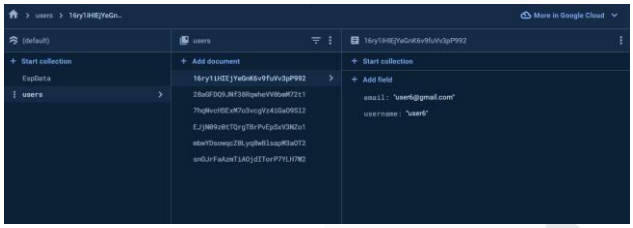
D. autentikasi dan pengelolaan data pengguna aplikasi

Untuk mengetahui pengimplementasian autentikasi dapat berjalan dengan baik, pengujian ini dilakukan dengan membuatkan beberapa akun. Setelah akun dibuat data pada Firebase Authentication akan

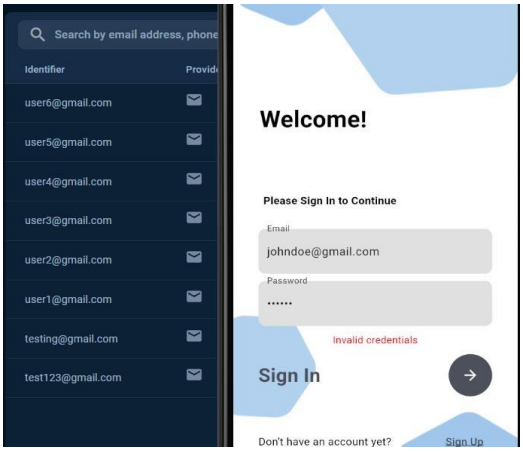
diperiksa apakah akun-akun tersebut telah terdaftar kedalamnya.Data pengguna juga akan dicek di dalam *database* untuk mengetahui apakah *database* berhasil menyimpan *username* dan email berdasarkan uid dari setiap akun yang telah terdaftar seperti yang ditunjukkan pada Gambar 4.4 dan Gambar 4.5. Pengujian ini cukup penting untuk memastikan kenyaman pengguna, dengan memastikan bahwa hanya akun yang terautentikasi yang dapat masuk ke dalam aplikasi untuk melakukan *montiroing* dan kontrol pada *air purifier*. Berdasarkan hasil pengujian yang ditunjukkan oleh Gambar 4.6 dapat disimpulkan jika hanya akun yang telah terdaftar yang dapat masuk ke dalam aplikasi *mobile*.Untuk memastikan bahwa hanya akun yang terdaftar yang dapat masuk kedalam aplikasi semua akun yang dibuat menggunakan password yang sama yaitu ‘123456’ dengan email dan berbeda-beda.



Gambar 4. 4 Akun yang terdaftar kedalam Firebase Authentication



Gambar 4. 5 Data pengguna yang tersimpan ke dalam database



Gambar 4. 6 Pengujian autentikasi penguna aplikasi

V. KESIMPULAN

Implementasi *Internet of Things* (IoT) pada sistem *air purifier* melalui integrasi dengan aplikasi *mobile* telah menunjukkan hasil yang positif. Pengguna dapat memantau kualitas udara yang diberikan oleh *air purifier* di dalam aplikasi, pengguna juga dapat mengendalikan air purifier melalui aplikasi. Untuk meningkatkan kenyamanan dan kemudahan aksesibilitas pengguna terdapat fitur statistik yang dapat memantau kualitas udara di dalam rumah selama *air purifier* beroperasi. Aplikasi juga hanya membiarkan pengguna yang telah terdaftar untuk masuk kedalam aplikasi untuk melakukan *monitoring* dan kontrol *air purifier*. Akan tetapi terdapat suatu hal yang harus diperhatikan, yaitu kapasitas *database*. Layanan Database saat ini hanya memungkinkan penulisan dan penjumlahan data sebanyak 20 ribu dan 50 ribu data per hari. Yang dimana pada *project* ini layanan *database* tersebut dapat bekerja untuk sekitar beberapa jam. Hal ini dapat diatasi dengan mengatur arus data yang ditulis dan dibaca oleh *database*. Namun akan lebih baik jika ada alternatif *database* lainnya agar implementasi IoT pada *air purifier* dapat bekerja lebih optimal

REFERENSI

[1] Tim Medis Siloam Hospitals. "8 Bahaya Polusi Udara bagi Kesehatan, Penting Dipahami!", 27 Oktober 2023, https://www.siloamhospitals.com/informasisiloam/artikel/bahaya-polusi-udara. Diakses pada 2 Agustus 2024.

[2] DLHK Kabupaten Mamuju. "APA ITU POLUSI UDARA DAN DARIMANA SUMBER POLUSI UDARA?", 7 Oktober 2023, https://dlhk.mamujukab.go.id/berita-5227-apa-itu-polusiudara-dan-darimana-sumber-polusi-udara.html. Diakses pada 2 Agustus 2024.

[3] Rita Puspita Sari. "Internet of Things (IoT): Pengertian, Cara Kerja dan Contohnya.", 24 Januari 2024, https://www.cloudcomputing.id/pengetahuan-dasar/iotpengertian-contohnya. Diakses pada 3 Agustus 2024.

[4] Putri Aprilia. "Apa itu Flutter? Simak Pengertian dan Alasan Mengapa Flutter Layak Anda Pakai!" 8 September 2021, https://www.niagahoster.co.id/blog/pengertian-flutter/. Diakses pada 3 Agustus