BAB I
PENDAHULUAN

1.1 Latar Belakang

Di era modern ini, perkembangan teknologi yang terjadi begitu pesat membuat kehidupan manusia sehari-hari semakin bergantung pada perangkat elektronik. Peralatan, seperti *smartphone*, mesin cuci, dan kipas angin menjadi contoh penggunaan teknologi dalam kehidupan sehari-hari. Sebagian besar perangkat elektronik menggunakan energi listrik sebagai sumber daya utama untuk beroperasi [1]. Jumlah perangkat yang relatif banyak membuat pengguna sering kali tidak menyadari besarnya konsumsi energi listrik yang digunakan.

Penggunaan energi listrik yang tidak terkontrol dapat menyebabkan pemborosan dan meningkatnya biaya tagihan penggunaan energi listrik [2]. Oleh karena itu, diperlukan sebuah sistem yang mampu membantu mengawasi dan mengontrol konsumsi energi listrik. Salah satu solusi modern untuk mengatasi permasalahan ini, yaitu perancangan *Smart Electrical System* (SES) untuk Pemantauan, Prediksi, dan Keamanan berbasis Android. SES telah terintegrasi dengan teknologi *Internet of Things* (IoT) dan dilengkapi dengan algoritma *Random Forest* yang dapat menganalisis dan memprediksi konsumsi energi listrik pengguna.

Teknologi IoT sendiri telah banyak diterapkan dalam berbagai bidang, salah satunya dalam bidang pengelolaan energi listrik [2]. Dengan memanfaatkan sensor tegangan dan arus seperti PZEM-004T, SES mampu membaca dan menampilkan informasi penggunaan energi listrik perangkat elektronik secara *realtime* melalui aplikasi Android. Informasi ini akan ditampilkan dalam bentuk grafik interaktif yang menggambarkan rata-rata konsumsi energi listrik pengguna berdasarkan durasi yang dipilih oleh pengguna. Adanya fitur ini tentunya dapat membantu pengguna memantau dan mengoptimalkan penggunaan listrik mereka.

Salah satu tantangan dalam pengelolaan konsumsi energi listrik, yaitu kemampuan untuk memprediksi total penggunaan energi listrik pada masa mendatang [3]. Untuk mengatasi permasalahan ini, SES dilengkapi dengan algoritma *Random Forest* untuk memprediksi total konsumsi listrik (kWh) pada

akhir bulan berdasarkan data penggunaan energi listrik pengguna selama bulan berjalan. *Random Forest* dipilih karena kemampuannya dalam menangani data yang kompleks dalam jumlah besar, memiliki toleransi yang tinggi terhadap '*missing data*', serta sifatnya yang *time-dependant* membuat *Random Forest* menjadi algoritma yang lebih stabil dibandingkan KNN [4].

Selain pemantauan konsumsi energi listrik, SES juga memperhatikan aspek keamanan penggunaan energi listrik. Karena itu SES dilengkapi dengan *relay* yang memungkinkan pengguna untuk mengontrol perangkat elektronik dari jarak jauh [2]. Selain itu, SES juga mampu memutuskan aliran listrik secara otomatis apabila terdeteksi anomali pada aliran listrik. Fitur ini tentunya sangat memudahkan kita dan dapat mengurangi risiko terjadinya kebakaran akibat korsleting listrik atau beban listrik yang berlebih.

Penelitian oleh Angga Aditya, dkk membuktikan bahwa sistem pemantauan konsumsi energi listrik berbasis IoT dapat memudahkan proses pengelolaan energi listrik dengan memanfaatkan mikrokontroler dan sensor yang terhubung dengan sebuah sistem [5]. Hal ini juga dikuatkan dengan penelitian karya Zefanya Rumpesak yang membahas tentang penerapan algoritma *K-Nearest Neighbor* (KNN) untuk memprediksi total konsumsi energi listrik guna mengoptimasi penggunaan listrik [6]. Dalam penelitian karya Annaya, dkk juga menunjukkan bahwa penggunaan perangkat seperti ESP32 dan PZEM-004T menjadi pilihan yang tepat untuk mengontrol perangkat elektronik [7].

Penerapan algoritma *Random Forest* juga telah dibuktikan mampu oleh penelitian karya Ergi Putra, dkk yang membahas tentang penggunaan algoritma *Random Forest* dalam berbagai aplikasi prediksi energi, salah satunya energi listrik. *Random Forest* dinilai mampu untuk memproses data yang kompleks dalam jumlah besar [8]. Dalam penelitian oleh Ashfania, dkk juga telah membuktikan penggunaan *Random Forest* untuk klasifikasi energi listrik dengan 2 kelas yaitu 'normal' dan 'gagal' serta kemampuan *Random Forest* untuk mendeteksi terjadinya anomali dalam sistem pembangkit listrik [9].

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, berikut ini merupakan permasalahan yang dirumuskan untuk Tugas Akhir ini:

1. Bagaimana merancang sistem pemantauan konsumsi energi listrik menggunakan teknologi *Internet of Things* (IoT) yang dapat menampilkan data konsumsi secara *realtime* berbasis Android?
2. Bagaimana menerapkan algoritma *Random Forest* untuk memprediksi total konsumsi energi listrik pengguna untuk membantu mengoptimalkan konsumsi energi listrik?
3. Bagaimana mengintegrasikan lapisan keamanan tambahan untuk perangkat elektronik guna mencegah risiko terjadinya kebakaran akibat malfungsi perangkat elektronik?

1.3 Tujuan Tugas Akhir

Berikut ini merupakan tujuan dari dilaksanakannya Tugas Akhir:

1. Merancang *Smart Electrical System* (SES) berbasis IoT yang mampu memantau konsumsi energi listrik perangkat elektronik pengguna secara *realtime* melalui aplikasi Android.
2. Menerapkan algoritma *Random Forest* untuk memprediksi total konsumsi energi listrik bulanan pengguna, berdasarkan data konsumsi energi listrik pengguna sehari-hari guna membantu pengguna mengoptimalkan penggunaan energi listrik mereka.
3. Mengintegrasikan fitur keamanan tambahan yang memungkinkan pengguna mengontrol nyala matinya perangkat elektronik serta pemutusan aliran listrik otomatis apabila terjadi anomali pada aliran listrik.

1.4 Manfaat Tugas Akhir

Berikut ini adalah manfaat yang diterima oleh masyarakat:

1. Sistem yang dibangun dapat memudahkan proses pemantauan konsumsi energi listrik sehari-hari.
2. Sistem yang dibangun menawarkan prediksi total konsumsi energi listrik yang dapat membantu pengguna untuk mengatur pemakaian listrik sehari-hari.

3. Sistem yang dibangun dapat memberikan lapisan keamanan tambahan bagi peralatan elektronik yang digunakan.

1.5 Batasan Masalah

Adapun batasan masalah yang telah ditetapkan untuk menjaga besaran ruang lingkup pengembangan, yaitu:

1. SES yang dirancang merupakan sebuah prototipe dan bukan bentuk final.
2. Aplikasi yang dirancang hanya dapat dijalankan pada perangkat dengan sistem operasi minimal Android Q (Android 10).
3. SES membutuhkan koneksi Internet yang memadai agar dapat beroperasi dengan optimal.
4. SES hanya dapat mengontrol fungsi nyala dan mati perangkat elektronik.
5. SES hanya dapat membaca tegangan (*Volt*), arus (*Ampere*), daya (*Watt*), dan energi (kWh).
6. SES hanya dapat memprediksi total konsumsi energi listrik untuk akhir bulan berjalan berdasarkan data penggunaan selama bulan berjalan.
7. SES menggunakan daya 220V sebagai acuan utama penggunaan daya.
8. Sistem SES yang dibangun bersifat statis dengan jumlah perangkat SES yang telah ditentukan sebelumnya.

1.6 Metode Penelitian

Berikut ini adalah tahapan dari metode penelitian yang digunakan:

1. Identifikasi Masalah

   Mengidentifikasi permasalahan yang ada dalam kehidupan sehari-hari dan solusi dari permasalahan tersebut.
2. Penentuan Tujuan

   Menentukan tujuan penelitian berdasarkan hasil identifikasi pada tahap sebelumnya.
3. Tinjauan Pustaka

   Berisikan kajian teori pendukung yang digunakan dalam penelitian ini, seperti teori perangkat elektronik, sistem elektrikal, teknologi IoT, dan sebagainya.
4. Analisis

Di tahapan ini akan dilaksanakan pengumpulan data dan pengolahan data yang telah dikumpulkan selama masa penelitian berlangsung.

5. Perancangan

   Di tahapan ini akan dilaksanakan proses perancangan untuk perangkat IoT yang akan digunakan, dan menyusun rancangan aplikasi Android.
6. Implementasi

   Di tahapan ini akan dilaksanakan proses pengembangan aplikasi Android dan menghubungkan aplikasi dengan perangkat elektronik.
7. Pengujian

   Di tahapan ini akan dilaksanakan proses pengujian kelayakan untuk perangkat dan aplikasi yang dibangun, serta menentukan apakah sistem yang dibangun dapat menyelesaikan permasalahan yang diangkat.

1.7 Sistematika Penulisan

Berikut ini adalah sistematika penulisan yang akan digunakan untuk menyusun laporan Tugas Akhir:

1. BAB I PENDAHULUAN

   Bagian ini berisikan latar belakang, rumusan masalah, tujuan Tugas Akhir, manfaat Tugas Akhir, batasan masalah yang telah ditentukan, metode penelitian, dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA

   Bagian ini berisikan kajian literatur terkait dengan topik Tugas Akhir, yang merujuk kepada penelitian terdahulu dengan topik serupa.
3. BAB III ANALISIS

   Bagian ini berisikan hasil analisis terhadap data yang telah dikumpulkan, dan penjelasan yang lebih lanjut terkait dengan Tugas Akhir ini serta peralatan yang digunakan selama perancangan Tugas Akhir.
4. BAB IV PERANCANGAN

   Bagian ini berisikan hasil rancangan terkait perangkat IoT dan aplikasi Android yang dikembangkan.
5. BAB V IMPLEMENTASI

Bagian ini berisikan proses implementasi dari rancangan yang telah dibuat sebelumnya.

6. BAB VI PENGUJIAN

   Bagian ini berisikan hasil pengujian yang dilakukan pada sistem yang dibangun, menyangkut kelayakan dan kemampuan SES dalam menyelesaikan permasalahan yang diangkat.

7. BAB VII KESIMPULAN DAN SARAN

   Bagian ini berisikan kesimpulan penulis terkait penelitian yang dilaksanakan dan saran untuk penelitian dan pengembangan selanjutnya.

BAB II
STUDI PUSTAKA

Bab ini berisi tentang teori pendukung seputar topik pengembangan Tugas Akhir.

2.1 Sistem Pemantauan

Sistem merupakan sekumpulan elemen yang saling berinteraksi satu sama lain guna mencapai suatu tujuan. Dalam konteks teknologi, sistem biasanya terdiri atas perangkat keras (*hardware*), perangkat lunak (*software*), dan media komunikasi yang saling bekerja sama untuk mengolah data. Di lain sisi, pemantauan merupakan proses pengawasan suatu aktivitas atau kegiatan yang dilakukan secara bekerlanjutan dengan tujuan untuk mengontrol, menganalisis, atau mengotimalkan kinerja suatu sistem. Proses pemantauan ini telah diterapkan pada berbagai bidang, seperti industri, kesehatan, transportasi, dan pengelolaan energi [10].

Dalam hal pemantauan konsumsi energi listrik, umumnya dirancang untuk mengukur penggunaan daya listrik dalam suatu lingkungan, seperti rumah tangga atau kantor. Dengan memanfaatkan teknologi *Internet of Things* (IoT) sistem yang dibangun umumnya memungkinkan pengguna untuk memantau konsumsi energi listrik secara *realtime*, mengidentifikasi pola penggunaan energi, dan mengoptimalkan penggunaan energi listrik [11].

2.2 Prediksi

Prediksi merupakan proses memperkirakan suatu kejadian atau nilai yang akan dihasilkan berdasarkan data historis atau analisis tren yang sedang berlangsung. Prediksi telah digunakan dalam berbagai bidang, seperti ekonomi untuk memprediksi kenaikan atau penurunan harga atau bidang meteorologi yang digunakan untuk memprediksi cuaca [11]. Secara umum, metode prediksi ini dapat diklasifikasikan menjadi 2 bagian, yaitu prediksi berbasis statistik dan prediksi berbasis AI (*Artificial Intelligence*). Pendekatan statistik biasanya mencakup metode regresi linier, *time series analysis*, sedangkan pendekatan AI

melibatkan teknik, seperti *Neural Networks*, *Machine Learning*, ataupun *Deep Learning* [12].

Dalam penerapannya, prediksi memiliki peranan penting dalam berbagai sistem teknologi modern, Dalam pengelolaan energi listrik, prediksi dapat digunakan untuk memperkirakan kebutuhan listrik di masa mendatang atau total konsumsi dalam satu periode [12].

2.3 Keamanan

Keamanan secara umum merupakan kondisi bebas ancaman atau bahaya yang dapat mengganggu stabilitas suatu sistem. Menurut Kamus Besar Bahasa Indonesia (KBBI), keamanan adalah keadaan bebas dari kejahatan, yang mencakup perlindungan terhadap kejahatan, kecelakaan, dan berbagai bentuk bahaya lainnya.

Sedangkan dalam konteks pemantauan konsumsi energi listrik, keamanan memainkan peranan yang penting guna menjaga keselamatan sistem kelistrikan. Pemantauan yang baik dapat membantu mengidentifikasi anomali pada aliran listrik yang dapat menandakan adanya masalah atau potensi bahaya seperti korsleting [13]. Tentunya keberadaan sistem pengaman ini dapat mengurangi rasa khawatir para pengguna terkait kemanan listrik mereka.

2.4 Daya Listrik

Daya listrik merupakan jumlah energi yang digunakan atau dihasilkan dalam suatu rangkaian listrik. Secara umum, daya listrik dipengaruhi oleh arus, tegangan, dan hambatan yang ada pada rangkaian listrik. Dalam Sistem Internasional (SI) telah ditetapkan bahwa satuan daya listrik adalah watt (W), yang setara dengan Joule per detik. Dalam menghitung jumlah listrik dalam suatu rangkaian dapat digunakan rumus [14] sebagai berikut:

$$P = V \times I \quad \text{.....(2.1)}$$

Keterangan:

P = Daya Listrik (Watt)

V = Tegangan (Volt)

I = Arus Listrik (Ampere)

Dalam menghitung total konsumsi energi listrik dalam 1 bulan serta rata-rata konsumsi harian pengguna dapat menggunakan rumus sebagai berikut [14]:

$$Total\ Energy\ Used: \sum_{i=1}^{n} \text{energi listrik harian}_i \quad \text{.....(2.2)}$$

$$Average\ Daily\ Usage: \frac{\text{Total konsumsi harian}}{\text{Jumlah data harian}} \quad \text{.....(2.3)}$$

Adapun rumus yang digunakan untuk menghitung persentase jumlah hari dan sisa hari dalam bulan berjalan [14]:

$$Month\ Progress: \frac{\text{Jumlah data harian}}{\text{Jumlah hari dalam 1 bulan}} \times 100 \quad \text{.....(2.4)}$$

$$Days\ Remaining: \text{Jumlah hari dalam 1 bulan} - \text{Jumlah data harian} \quad (2.5)$$

Sedangkan untuk menghitung besar penyebaran atau variabilitas data dari nilai rata- rata digunakan rumus standar deviasi, yaitu:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad \text{.....(2.6)}$$

Keterangan:

s = Standar deviasi sampel

n = Jumlah sampel

x_i = Nilai data ke-i

$\bar{x}$ = Rata-rata dari data

$(x_i - \bar{x})^2$ = Kuadrat sel

2.5 *Random Forest Algorithm*

Algoritma merupakan serangkaian langkah-langkah atau instruksi yang dilakukan untuk menyelesaikan suatu masalah atau menjalankan suatu tugas. Dalam ilmu komputasi, algoritma biasanya digunakan untuk memproses data,

mengatasi masalah, atau optimalisasi sistem. Algoritma sendiri dapat diekspresikan dalam berbagai bentuk, seperti dalam bentuk teks, atau dalam bentuk diagram alur (*flowchart*), ataupun dalam bentuk kode pemrograman.

Random Forest merupakan salah satu algoritma dari cabang ilmu pengetahuan *Machine Learning* yang cukup populer, umumnya digunakan untuk tugas klasifikasi atau regresi. *Random Forest* bekerja dengan membangun sekumpulan pohon keputusan dan menggabungkan hasilnya untuk meningkatkan akurasi prediksi serta mengurangi risiko terjadinya *overfitting*. Setiap pohon yang digunakan menggunakan sampel secara acak dari *dataset*, dan setiap node *subset* acak dari fitur dipilih untuk mendapatkan *split* terbaik, teknik ini dikenal dengan nama *Bootstrap Sampling*. Pendekatan menggunakan *bootstrap sampling* dapat meningkatkan akurasi prediktif dan mengontrol *overfitting* dengan cara mengurasi korelasi antara satu pohon dengan pohon yang lainnya [15].

Keunggulan utama dari algoritma *Random Forest*, yaitu kemampuannya untuk menangani *dataset* dengan jumlah fitur yang besar tanpa harus melakukan seleksi fitur. Selain itu *Random Forest* juga dapat memberikan estimasi pentingnya variabel dalam proses klasifikasi dan memiliki toleransi yang tinggi terhadap *data-loss*. Namun *Random Forest* merupakan algoritma yang memiliki tingkat kompleksitas yang tinggi dibandingkan dengan algoritma pohon keputusan lainnya sehingga proses interpretasi model menjadi lebih rumit [16]. Berikut ini *pseudocode* untuk regresi menggunakan *Random Forest* [16]:

1. Tentukan jumlah pohon yang diinginkan (*n_trees*).
2. Analisis data yang dimiliki.
3. *Sampling* data historis untuk *training*.
4. Mengekstrak fitur dari data yang dimiliki.
5. Lakukan pemprosesan dengan *decision tree* untuk setiap i = 1 sampai *n_trees*:
    a. Ambil sampel secara acak dari *dataset* D → D_i (*bootstrap sampling*).
    b. Bangun pohon regresi T_i untuk setiap D_i dengan:
        i. Pada setiap node, pilih *subset* acak dari fitur (*feature_subset*).

ii. Tentukan fitur dan *threshold* terbaik dari *subset* fitur untuk menghasilkan *split* terbaik (Lakukan berdasarkan minimisasi MSE (*Mean Squared Error*)).

iii. Lanjutkan sampai mencapai titik akhir sampel per node tercapai.

c. Ambil rata-rata hasil dari setiap pohon.

2.6 Teknologi Pengembangan Sistem

Berikut ini merupakan pengertian seputar teknologi pengembangan sistem yang digunakan:

2.6.1 Android

Android merupakan sistem operasi berbasis Linux yang dirancang secara khusus untuk perangkat seperti *smartphone* atau *tablet*. Android pertama kali dikembangkan oleh *Open Handset Alliance* yang dipimpin oleh Google, seri pertama Android kemudian rilis pada 23 September 2008. Sistem operasi Android bersifat *open-source* sehingga memungkinkan pengembang dan produsen perangkat lunak untuk memodifikasi dan mendistribusikannnya secara bebas. Dengan dukungan dari komunitas, Android menjadi *platform* dominan dalam industri perangkat *mobile* [17].

Android telah mengalami berbagai pembaruan seiring berkembangnya teknologi. Setiap versi dari Android diberi nama berdasarkan makanan, seperti *Cupcake*, *Jellybeans*, *KitKat*, dan seterusnya. Namun, penamaan ini berubah menjadi angka ketika memasuki era Android 10. Tentunya keunggulan utama dari Android, yaitu sifatnya yang *open-source*, hal ini membuat banyak produsen perangkat lunak memilih untuk mengadopsi dan menggunakan Android. Saat ini Android tidak hanya digunakan oleh *smartphone* atau *tablet* saja. Terdapat perangkat lain yang dikembangkan menggunakan Android, seperti *Smart TV* atau *Smart Watch* [17].

2.6.2 *Internet of Things*

Internet of Things (IoT) mengarah pada jaringan objek fisik yang biasanya dilengkapi dengan mikrokontroler, sensor dan aktuator. Objek ini terhubung

dengan suatu jaringan dan bertujuan agar terkoneksi dan dapat bertukar data dengan sistem melalui Internet. Umumnya objek tersebut dapat mengumpulkan data dari sensor yang kemudian dikirimkan oleh mikrokontroler pada antarmuka perangkat lunak melalui suatu jaringan. Hal tersebut memungkinkan pengguna untuk dapat mengetahui informasi yang diperoleh melalui sensor. Pengguna juga dapat berkomunikasi dengan objek, contohnya pengguna dapat memberikan perintah melalui perangkat lunak yang terhubung, dan mikrokontroler akan memerintahkan untuk mengeksekusi perintah yang diberikan [18].

Dalam praktiknya, IoT dapat ditemukan pada berbagai hal termasuk hal-hal sederhana, contohnya pelacak kebugaran yang mengukur langkah harian kita. Konektivitas ini memungkinkan pengumpulan data yang lebih banyak sehingga meningkatkan keakuratan informasi yang diolah berdasarkan data. Kerap kali penerapan IoT dipadukan dengan *Artificial Intelligence* (AI) sehingga memperluas potensi yang dapat dilakukan dengan teknologi IoT [19].

2.6.3 Kotlin

Kotlin merupakan bahasa pemrograman modern yang dikembangkan oleh JetBrains, sebuah perusahaan yang juga terkenal dengan produk seperti IntelliJ IDEA. Kotlin dirancang untuk beroperasi pada *Java Virtual Machine* (JVM) dan pertama kali diperkenalkan tahun 2011. Hubungan erat antara Kotlin dengan JVM, pengembang dapat menulis kode yang berjalan di lingkungan Java dengan lebih mudah. Selain itu Kotlin juga mendukung kompilasi ke JavaScript, hal ini dapat memudahkan pengembang dalam pengembangan *website* tanpa harus mempelajari bahasa baru [20].

Kemampuan *multi-platform* ini menjadikan Kotlin bahasa pemrograman yang fleksibel dan dapat digunakan di berbagai jenis proyek, seperti pengembangan aplikasi *desktop*, *mobile*, hingga *server-side*. Selain *mutli-platform* ada beberapa fitur lain yang mendukung kepopuleran Kotlin. Struktur bahasa Kotlin lebih ringkas dibandingkan dengan pendahulunya, yaitu Java. Hal ini juga yang membuat Kotlin memungkinkan pengguna untuk menulis kode yang lebih rapi dan mudah dibaca. Fitur-fitur seperti *'Null-Safety'* dan interoperabilitas penuh dengan Java menjadi fitur pendukung yang dimiliki Kotlin [21].

2.6.4 Firebase

Firebase merupakan sebuah *platform* pengembangangan aplikasi yang dikembangkan oleh *Firebase Incorporation*, yang kemudian diambil alih oleh Google pada tahun 2014. Saat ini Firebase menjadi salah satu *platform* terfavorit para pengembang untuk mengelola basis data aplikasi mereka. Firebase menawarkan banyak fitur yang mendukung proses pengolahan *back-end* tanpa harus mengelola *server* sendiri (*serverless*). Hal ini tentunya membantu proses pengembangan dan mengelola aplikasi. Ditambah Firebase sendiri telah terintegrasi dengan ekosistem Google yang juga membantu para pengembang. Beberapa fitur unggul yang ditawarkan oleh Firebase, yakni *RealTime Database*, *FireStore*, *Google Auth* dan masih banyak lagi [22].

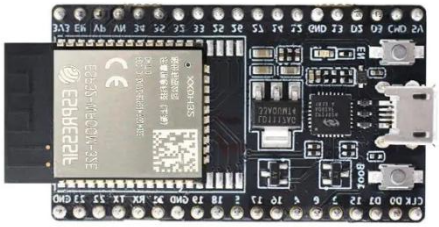
2.6.5 Android Studio IDE

Android Studio merupakan *Integrated Development Environment* (IDE) resmi yang dikembangkan oleh Google pada tahun 2013. Sesuai dengan namanya, Android Studio IDE dirancang khusus untuk pengembangan aplikasi berbasis Android. Dibangun di atas *platform* IntelliJ IDEA oleh JetBrains, Android Studio menawarkan banyak fitur yang dapat membantu para pengembang untuk mengembangkan aplikasi berbasis Android. Android Studio umumnya dijalankan menggunakan bahasa pemrograman Kotlin atau bahasa pemrograman C yang terintegrasi dengan Java [23].

2.7 Perangkat Keras yang Digunakan

Dalam Tugas Akhir ini, digunakan beberapa perangkat keras seperti Mikrokontroler ESP32 untuk mengontrol perangakat yang lain, Sensor PZEM-004T untuk membaca tegangan arus listrik, dan *Solid State Relay* untuk memutuskan atau menyalurkan aliran listrik ke perangkat elektronik. Berikut ini adalah penjelasan singkat untuk masing masing perangkat keras.

2.7.1 ESP32



Gambar 2.1 ESP32 [24]

ESP32 merupakan sebuah mikrokontroler yang dilengkapi dengan konektivitas *Bluetooth* dan *Wi-Fi*. Mikrokontroler ini biasanya digunakan untuk proyek berbasis IoT, dan umumnya menggunakan prosesor Xtensa *dual-core* 32-bit LX6 yang dapat beroperasi hinggan 240 MHz. Ditambah dengan memori 520 KiB SRAM dan 449 KiB ROM yang menunjang kemampuan dari mikrokontroler ini. Dari segi konektivitas, ESP32 menggunakan standar *Wi-Fi* 802.11 b/g/n untuk jaringan nirkabel dan *Bluetooth* 4.2 BR/EDR dan BLE untuk komunikasi antar perangkat. Selain itu, ESP32 juga memiliki berbagai antarmuka periferal, misalnya ADC SAR 12-bit sampai dengan 18 saluran, 34 GPIO yang dapat diprogram, dan antarmuka komunikasi seperti SPI, I2C, I2S, dan UART [25].

2.7.2 Sensor PZEM-004T

Gambar 2.2 PZEM-004T [26]

PZEM-004T merupakan sebuah modul sensor yang dirancang untuk mengukur berbagai parameter aliran listrik AC, misalnya tegangan, arus, daya aktif, energi aktif dan lain lain. Modul ini menggunakan protokol komunikasi Modbus-RTU melalui antarmuka TTL sehingga lebih mudah untuk diintegrasikan dengan mikrokontroler seperti Arduino atau ESP32. PZEM-004T sendiri terdapat dua varian, yang pertama versi 10A yang menggunakan *shunt internal* dan versi 100A yang menggunakan transformator arus eksternal. PZEM-004T mampu untuk mengukur tegangan dalam rentang 80-260V dengan akurasi 0,5% dan daya aktif hingga 2,3kW untuk varian 10A atau 23kW untuk varian 100A. Selain itu PZEM-004T juga dapat membaca frekuensi listrik dalam rentang 45-65Hz, faktor daya antara 0.00-1.00, dan mencatat konsumsi energi sampai 9999.99 kWh dengan akurasi tinggi [27].

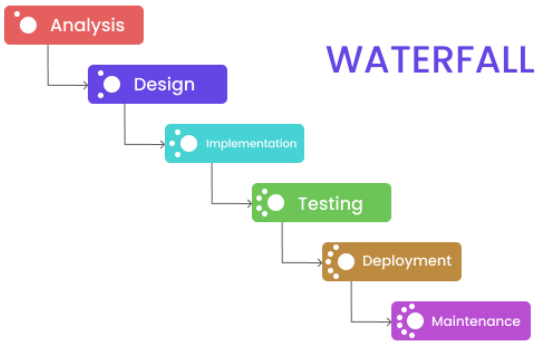
2.7.3 *Solid State Relay*

Gambar 2.3 *Solid State Relay* [28]

Solid State Relay (SSR) merupakan modul saklar elektronik yang dapat mengaktifkan atau memutuskan aliran listrik ketika tegangan eksternal diterapkan pada terminal kontrolnya. SSR berbeda dengan *relay* elektromekanis pada umumnya, SSR tidak meiliki komponen gerak dan hanya mengandalkan elemen semikonduktor, seperti *triac*, *dioda*, *thyristor*, atau transistor untuk melakukan operasi *switching*. SSR biasanya terdiri dari sensor yang dapat merespons sinyal kontrol, perangkat *switching* elektronik yang dapat mengalihkan daya ke sirkuit beban, dan mekanisme kopling yang memungkinkan sinyal kontrol mengaktifkan

switch tanpa bagian mekanis. SSR dianggap lebih unggul karena memiliki kecepatan *switching* yang relatif tinggi dibandingkan *relay* elektromekanis, memiliki umur yang lebih lama, dapat beroperasi dengan senyap, dan memiliki operasi yang terbebas dari bouncing kontak. Namun SSR memiliki keterbatasan dalam menahan lonjakan beban dan biasanya dilengkapi dengan resistensi "*on*" yang lebih tinggi [29].

2.8 Metode Pengembangan Sistem



Gambar 2.4 Metodologi *Waterfall* [30]

Metodologi pengembangan sistem merupakan pendekatan yang digunakan oleh pengembang dalam merancang, mengembangkan, atau memelihara sistem perangkat lunak atau produk lainnya. Metodologi umumnya mencakup prinsip, teknik, dan proses yang mengatur jalannya proyek pengembangan sistem untuk memastikan hasil yang maksimal. Metodologi pengembangan sendiri dapat dibagi menjadi beberapa jenis, seperti metode tradisional, iteratif, dan *agile*. Tentunya masing-masing metode ini memiliki fokus yang berbeda, baik dari segi tahapan maupun fleksibilitas [31].

Metodologi yang akan digunakan dalam Tugas Akhir ini, yakni metodologi *Waterfall*. Metodologi *Waterfall* bersifat linier dan terstruktur, di mana artinya setiap tahap pengembangan harus diselesaikan terlebih dahulu sebelum lanjut ke tahap berikutnya. *Waterfall* sendiri terdiri dari 6 tahapan, yaitu Analisis, Perancangan, Implementasi, Pengujian, Perilisan, dan Pemeliharaan. Berikut ini adalah penjelasan singkat seputar langkah-langkah metodologi *Waterfall*:

1. Tahapan analisis umumnya berisikan pengumpulan dan identifikasi kebutuhan sistem dari pengguna, membahas tentang target pengguna, pengumpulan data, persyaratan sistem dan lain-lain.
2. Tahapan perancangan umumnya berisikan rancangan arsitektur sistem, desain *database*, alur data, rancangan antar muka, dan lain-lain.
3. Tahapan implementasi umumnya berisikan proses pengembangan berdasarkan desain sistem yang telah dibuat sebelumnya.
4. Tahapan pengujian umumnya berisikan hasil integrasi dan pengujian untuk memastikan sistem dapat berjalan sesuai dengan kebutuhan.
5. Tahapan penerapan dimulai ketika sistem sudah lulus pengujian dan mulai digunakan oleh masyarakat.
6. Tahapan pemeliharaan umumnya berisikan kegiatan, seperti perbaikan *bug*, peningkatan performa, dan lain-lain.

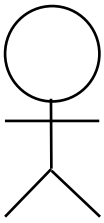
Keunggulan dari metode *Waterfall*, yakni strukturnya yang jelas dan mudah dipahami sehingga dapat membantu mengatur pengelolaan proyek dengan lebih mudah. Kelemahan *Waterfall* terdapat pada fleksibilitasnya, Metodologi *Waterfall* kurang baik dalam merespons terhadap perubahan [30].

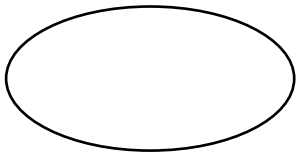
2.9 Kakas Pemodelan

Selama pengembangan Tugas Akhir ini akan digunakan kakas pemodelan *Activity Diagram* untuk menggambarkan alur kerja dari sistem yang dirancang, *Use Case Diagram* untuk menggambarkan modul dari aplikasi yang dirancang dan *Class Diagram* untuk menggambarkan hubungan antar kelas.

2.9.1 *Use Case Diagram*

Tabel 2.1 Simbol *Use Case Diagram* [32]

Model	Keterangan
	***Actor*** Model yang digunakan untuk menggambarkan pengguna, sistem atau alat.

	***Use Case*** Model yang digunaka untuk menggambarkan interaksi antara sistem dengan aktor.
Model	**Keterangan**
	Association Model yang digunakan sebagai penghubung antara aktor dan *use case* .

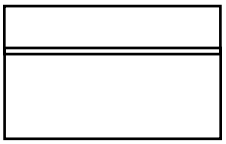
Use Case Diagram merupakan salah satu diagram *Unified Modelling Language* (UML) yang umumnya digunakan untuk menunjukkan bagaimana pengguna atau aktor akan berinteraksi dengan sistem melalui serangkaian skenario penggunaan. *Use Case Diagram* memberikan gambaran tentang fungsionalitas sistem dari sudut pandang pengguna atau aktor dengan menampilkan berbagai proses yang dikenal sebagai *use-case*.

Komponen utama dalam *Use Case Diagram* adalah aktor, kasus pengujian, dan hubungan antar elemen. Disini aktor merepresentasikan entitas luar yang berinteraksi dengan sistem, misalnya pengguna, perangkat keras, ataupun sistem yang lain. Sedangkan kasus pengujian, sesuai dengan namanya merupakan deskripsi fungsi spesifik dari sistem yang disediakan untuk aktor. Hubungan antar elemen menggambarkan keterlibatan aktor dengan fungsi-fungsi tertentu [33].

Use Case Diagram sering digunakan karena kemampuannya untuk mengidentifikasi kebutuhan pengguna dan fungsionalitas yang diharapkan oleh sistem. Dengan memvisualisasikan interaksi antara sistem dengan pengguna, pengembang dapat dengan lebih mudah memahami apa saja yang dibutuhkan oleh pengguna [33].

2.9.2 *Class Diagram*

Tabel 2.2 Simbol *Class Diagram* [34]

Model	Keterangan
	***Class*** Berisikan objek-objek yang berbagi atribut dan operasi yang sama

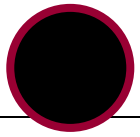
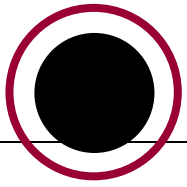
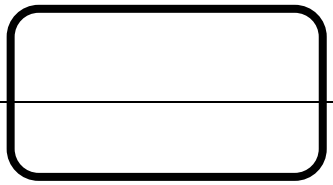
Model	Keterangan
	Generalisasi Hubungan antar objek yang berbagi perilaku dan struktur data

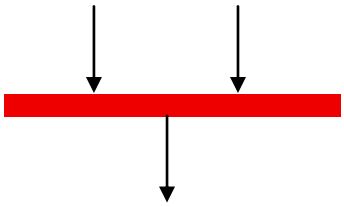
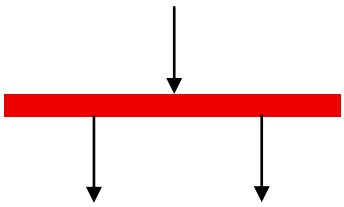
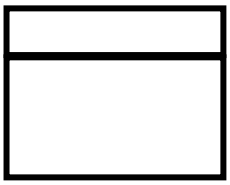
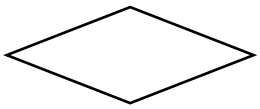
Class Diagram merupakan salah satu komponen utama dalam *Unified Modelling Language* (UML). *Class Diagram* digunakan untuk memodelkan struktur sistem yang berorientasi objek, dengan menggambarkan kelas-kelas dalam sistem, atribut, dan metode yang dimilikinya, serta hubungan antar kelas. *Class Diagram* memungkinkan pengembang untuk lebih memahami dan merancang struktur sistem dengan cara yang lebih terstruktur dan sistematis [34].

Dalam *Class Diagram*, setiap kelas digambarkan sebagai kotak dengan tiga bagian. Nama kelas ditampilkan dibagian atas, atribut ditampilkan di bagian tengah, dan metode ditampilkan dibagian bawah. Hubungan antar kelas, seperti asosiasi, generalisasi, dan ketergantungan direpresentasikan dengan garis penghubung dengan notasi khusus. Penggunaan *Class Diagram* sangat berguna selama tahap desain karena dapat membantu pengembang lebih memahami sistem yang akan dibangun dengan menggambarkan struktur data, dan logika yang ada dalam program [34].

2.9.3 *Activity Diagram*

Tabel 2.3 Simbol *Activity Diagram* [35]

Model	Keterangan
	***Start Node*** Model yang digunakan untuk menandakan titik awal dari sebuah alur
	***End Node*** Model yang digunakan untuk menandakan titik akhir dari sebuah alur
	***Activity*** Model yang digunakan untuk

Model	Keterangan
	menggambarkan aktivitas dalam sebuah alur
	***Join*** Model yang digunakan untuk menggabungkan beberapa aktivitas menjadi satu
	***Fork*** Model yang digunakan untuk memodelkan alur yang berjalan secara paralel
	***Swimlane*** Model yang digunakan untuk memisahkan antar entitas yang bertanggung jawab atas sebuah aktivitas
	***Decision*** Model yang digunakan untuk pengambilan keputusan
	Transition Model yang digunakan untuk menghubungkan aktivitas sebelumnya dan aktivitas selanjutnya

Activity Digram merupakan salah satu diagram dari *Unified Modelling Langugage* (UML). *Activity Diagram* digunakan karena kemampuannya untuk menggambarkan alur kerja atau proses dalam suatu sistem atau aplikasi, baik itu alur operasional aplikasi ataupun proses bisnis perusahaan. *Activity Diagram* memberikan representasi visual dari serangkaian aktivitas yang terjadi dalam suatu sistem, mulai dari awal hingga akhir. Setiap aktivitas dalam diagram ini direpresentasikan dengan bentuk oval atau lingkaran, garis panah yang menggambarkan urutan serta arah dari alur aktivitas dan juga *decision points*, yang dapat mempengaruhi alur aktivitas berdasarkan kondisi tertentu [35].

Activity Diagram umumnya digunakan untuk memodelkan proses bisnis yang kompleks, dimana terlibat banyak interaksi dari berbagai komponen atau aktor. Contohnya, seperti proses bisnis yang melibatkan berbagai departemen atau tim, *Activity Diagram* dapat membantu menggambarkan bagaimana tugas-tugas tersebut terdistribusi, bagaimana keputusan dapat diambil dari berbagai titik dalam pengerjaan, dan lain sebagainya [35].

2.10 *Black Box* dan *White Box Testing*

Black Box Testing merupakan salah satu metode pengujian perangkat lunak yang berfokus pada evaluasi fungsionalitas sistem tanpa pertimbangan terhadap struktur internal sistem. Dengan menggunakan *Black Box Testing* penguji hanya mengetahui *input* yang diberikan dan *output* yang diharapkan, tanpa informasi bagaimana proses internal sistem bekerja. Hal ini dilakukan agar penguji dapat fokus pada *output* yang dihasilkan guna memastikan sistem barjalan dangan baik. *Black Box Testing* digunakan karena dapat membantu mengidentifikasi kesalahan pada antarmuka aplikasi, alur sistem, bahkan integrasi sistem secara keseluruhan [36].

Di lain sisi, *White Box Testing* merupakan metode pengujian yang melihat langsung ke struktur internal sistem. *White Box Testing* bertujuan untuk memeriksa logika program, alur sistem, integrasi algoritma, dan lain lain. Kedua metode ini sering digunakan secara bersamaan untuk memastikan kualitas dari sistem yang dirancang secara menyeluruh, di mana *Black Box Testing* fokus pada penilaian kesesuaian fungsi dengan kebutuhan pengguna, sedangkan *White Box*

Testing berfokus untuk memastikan keandalan dan kemanan dari sisi kode internal [36].

2.11 Penelitian Terkait

Terdapat beberapa penelitian terdahulu yang digunakan sebagai panduan untuk pengembangan Tugas Akhir ini. Berikut ini adalah beberapa penelitian terdahulu yang digunakan:

1. Penelitian karya Zefanya Rumpesak yang membahas tentang sistem pemantauan dan prediksi konsumsi daya listrik menggunakan *Internet of Things* dan juga menerapkan algoritma *K-Nearest Neighbor* (KNN) untuk memprediksi konsumsi energi listrik berdasarkan data historis, dengan menggunakan *Mean Absolute Error* (MAE) dan *Mean Absolute Percentage Error* (MAPE) [6]. Relevansi penelitian ini dengan *Smart Electrical System* yang dikembangkan, yaitu penggunaan pendekatan IoT namun, dengan menerapkan algoritma yang berbeda, yaitu *Random Forest*. *Random Forest* dipilih karena kemampuannya untuk mengolah data dalam skala yang besar serta lebih baik dalam memproses data progresif.
2. Penelitian karya Ergi Putra, dkk dengan judul *Forecasting* Produksi Energi *Photovoltaic* Menggunakan Algoritma *Random Forest Classification*. Penelitian ini membahas tentang prediksi produksi energi listrik yang dihasilkan dari panel surya (*Photovoltaic*) menggunakan algoritma *Random Forest*. Tujuan dari penelitian ini, yaitu untuk meningkatkan akurasi prediksi produksi energi listrik berdasarkan berbagai faktor seperti suhu lingkungan, kondisi cuaca, dan intentsitas cahaya matahari. *Random Forest* dipilih karena kemampuannya untuk menangani data yang kompleks dan mengurangi risiko *overfitting* sehingga dapat menghasilkan data prediksi yang lebih stabil dibandingkan algoritma lain, seperti *Linear Regression* atau *K-Nearest Neighbor* (KNN) [8]. Relevansi penelitian ini, yaitu penggunaan algoritma yang sama namun, daripada memprediksi total produksi, SES memprediksi total konsumsi energi listrik pada akhir bulan.
3. Penelitian karya Annaya, dkk dengan judul *Intrusion Detection System Internet of Things* Pada Sistem Pemantauan Pemakaian Energi Listrik

Berbasis *Machine Learning*, membahas tentang penerapan sistem deteksi intrusi (*Intrusion Detection System*/IDS) dengan memanfaatkan teknologi *Internet of Things* (IoT) guna meningkatkan keamanan pada sistem pemantauan konsumsi energi listrik. Beberapa perangkat yang digunakan merupakan ESP32 dan PZEM-004T [7]. Relevansi dengan SES terletak pada aspek pemantauan dan keamanan menggunakan sistem IoT. Jika penelitian ini lebih fokus pada serangan *cyber* pada jaringan listrik, SES lebih menitikberatkan pengamanan atas kenjanggalan arus listrik.

Berikut ini adalah hasil dari ulasan yang dilakukan terkait penelitian terdahulu yang menjadi referensi penulisa Tugas Akhir ini.

Tabel 2.4 Tabel Perbandingan

Kriteria Perbandingan	**Penelitian 1** [6]	**Penelitian 2** [8]	**Penelitian 3** [7]
Permasalahan	Permasalahan yang diangkat dalam penelitian ini, yaitu borosnya penggunaan listrik di rumah tangga yang umumnya disebabkan oleh kelalaian pengguna. Contohnya seperti membiarkan alat elektronik tetap menyala meskipun tidak digunakan.	Penelitian ini berfokus pada prediksi besarnya energi listrik yang dihasilkan oleh sebuah Sistem Pembangkit Listrik Tenaga Surya (PLTS) berdasarkan kondisi tertentu, seperti suhu, intensitas sinar matahari, dan kelembapan.	Permasalahan yang diangkat dalam penelitian ini merupakan konsumsi energi listrik yang tidak terkontrol dengan baik sehingga dapat menyebabkan kerugian, terutama dalam hal biaya. Karena itulah, dibangun sebuah sistem pemantauan konsumsi energi listrik berbasis IoT untuk memudahkan pemantauan dan pengontrolan penggunaan energi.

Kriteria Perbandingan	**Penelitian 1** [6]	**Penelitian 2** [8]	**Penelitian 3** [7]
Tujuan	Membangun sebuah sistem yang dapat memantau dan memprediksi konsumsi energi listrik menggunakan algoritma KNN.	Membangun model prediksi energi PLTS menggunakan algoritma *Random Forest* yang dapat memproyeksikan besar-kecilnya energi listrik yang dihasilkan berdasarkan data historis lingkungan.	Tujuan dari penelitian ini tentunya untuk membangun sebuah sistem pemantauan konsumsi energi listrik berbasis Internet of Things (IoT), yang dapat memonitor penggunaan energi listrik dan memberikan peringatan apabila terdeteksi anomali pada penggunaan listrik.
Objek Prediksi	Konsumsi Energi Listrik.	Hasil produksi energi listrik PLTS.	Pola penggunaan energi listrik untuk mengidentifikasi anomali.
Komponen IoT yang digunakan	1. Mechanical Switch Relay 2. PZEM-004T 3. ESP8266	Tidak Dicantumkan	1. *Mechanical Switch Relay* 2. PZEM-004T 3. ESP32 4. I2C LCD Display
Fitur yang ditawarkan	Memberikan informasi pemakaian energi listrik dan memungkinkan pengguna untuk mematikan serta	Prediksi daya yang dihasilkan oleh PLTS berdasarkan Nilai Radiasi, Nilai Kelembapan, dan Nilai Suhu.	Mendeteksi intrusi pada aliran listrik dan memberikan peringatan.

Kriteria Perbandingan	**Penelitian 1** [6]	**Penelitian 2** [8]	**Penelitian 3** [7]
	menyalakan perangkat. Serta memberikan prediksi konsumsi energi listrk menggunakan KNN untuk tanggal yang dipilih.		
Data yang digunakan	Data konsumsi energi listrik yang telah dikumpulkan.	Data sekunder yang dari sistem SCADA PLTS PLN 7 MWp di Desa Sengkol.	Data yang diperoleh dari PZEM-004T yang dilatih menggunakan model *Intrusion Detection System* (IDS).
Algoritma	*K-Nearest Neighbor* (KNN).	*Random Forest.*	*Random Forest.*
Metode Evaluasi Performa	MAE dan MAPE.	*Confusion Matrix.*	*Black Box Testing.*

Sebagai sumber referensi penelitian ini, terdapat kesamaan dan juga perbedaan dengan masing masing penelitian terdahulu. *Smart Electrical System* (SES) dirancang dengan tujuan untuk membantu pengguna mengawasi penggunaan listrik mereka dengan menampilkan konsumsi *realtime* dan juga data historis konsumsi energi listrik. Berbeda dengan penelitian karya Zefanya Rumpesak, prediksi konsumsi dilakukan menggunakan algoritma *K-Nearest Neighbor* (KNN), SES melakukan prediksi menggunakan algoritma *Random Forest*. SES juga dilengkapi dengan *solid state relay* (SSR) yang dapat menyalakan atau mematikan perangkat dengan mudah berbeda dengan penelitian sebelumnya yang dominan menggunakan *mechanical switch relay* (MSR). Data yang digunakan untuk prediksi merupakan data perolehan dari sensor PZEM-

004T selama bulan berjalan untuk memprediksi total konsumsi di akhir bulan tersebut.

BAB III
ANALISIS

Pada bagian ini akan dijabarkan analisis pembangunan sistem, di mana analisis merupakan langkah pertama dari metodologi *Waterfall*.

3.1 Analisis Target Pengguna

Target pengguna dari sistem ini merupakan masyarakat umum Indonesia. *Smart Electrical System* (SES) dirancang untuk membantu pengguna mengontrol konsumsi energi listrik mereka. Selain itu SES juga dilengkapi dengan algoritma yang memungkinkan SES memprediksi total konsumsi energi listrik pengguna untuk bulan berjalan.

3.2 Sumber Data

Data yang digunakan selama pengembangan Tugas Akhir ini berasal dari konsumsi energi listrik harian penulis. Proses pengumpulan data berlangsung selama 3 bulan mulai dari bulan Maret sampai dengan bulan Mei. Berdasarkan informasi tersebut, dilakukanlah pengolahan data untuk menampilkan pola penggunaan, konsumsi sementara listrik pengguna dan memprediksi total konsumsi pada akhir bulan berjalan.

3.3 Identifikasi Masalah dan Kesempatan

Berikut ini merupakan hasil identifikasi masalah dan solusi yang ditawarkan.

Tabel 3.1 Identifikasi Masalah dan Kesempatan

Masalah	Kesempatan
Sebagian besar masyarakat tidak menyadari seberapa besar konsumsi energi listrik harian mereka sehingga kerap kali terjadi pemborosan konsumsi listrik rumah.	Mengembangkan sebuah sistem pemantauan berbasis *Internet of Things* (IoT) yang mencatat konsumsi energi listrik pengguna, dan menampilkan grafik penggunaan berdasarkan filter yang dipilih.

Masalah	Kesempatan
Bagi sebagian masyarakat yang menggunakan tagihan listrik pascabayar kerap kali mengalami kesulitan untuk memperkirakan biaya energi listrik yang digunakan.	Mengembangkan sebuah sistem yang dilengkapi algoritma yang dapat memprediksi total konsumi energi pengguna untuk akhir bulan berjalan.
Sering kali terjadi anomali pada perangkat elektronik pengguna yang tidak disadari oleh masyarakat.	Mengembangkan sebuah sistem yang bisa memberikan informasi kepada pengguna apabila ada kemungkinan masalah pada perangkat elektronik atau aliran listrik.

3.4 Persyaratan dan Preferensi Sistem Baru

Pada Tugas Akhir ini akan dirancang sebuah sistem yang dapat membantu masyarakat selaku pengguna untuk memantau konsumsi energi listrik sehari-hari mereka. Sistem yang dibangun diberi nama *Smart Electrical System* (SES). Dalam satu perangkat SES terdapat sebuah mikrokontroler, ESP32 yang akan dihubungkan dengan *database online* Firebase, via Internet. SES juga menggunakan PZEM-004T yang berfungsi sebagai sensor untuk membaca daya listrik dari sebuah perangkat elektronik dan *Solid State Relay* (SSR) yang berfungsi untuk memutuskan atau menyambungkan aliran listrik ke perangkat elektronik. Pada aplikasi yang dibangun digunakan algoritma *Random Forest* untuk memprediksi total konsumsi energi listrik pengguna untuk bulan berjalan. Prediksi ini akan disimpulkan dari data konsumsi energi listrik sehari-hari pengguna selama bulan berjalan. Berdasarkan uraian di atas, berikut ini adalah fitur yang akan tersedia dalam sebuah perangkat SES:

1. SES dapat menampilkan pola penggunaan energi listrik *realtime* dalam bentuk grafik.
2. SES dapat menampilkan data konsumsi energi listrik pengguna berdasarkan durasi yang ditentukan oleh pengguna berupa harian, mingguan, atau bulanan.
3. SES dapat memprediksi total konsumsi energi listrik pengguna pada akhir bulan berjalan berdasarkan data konsumsi energi listrik pengguna sehari-hari selama bulan berjalan.

4. SES dapat memberikan informasi apabila ada kemungkinan terjadi gangguan pada perangkat elektronik atau aliran listrik.

3.5 Manajemen Risiko

Pada bagian ini akan dijabarkan hasil analisis mengenai risiko yang dapat terjadi selama pengembangan Tugas Akhir.

Tabel 3.2 Manajemen Risiko

NO	Risiko	Penyebab	Efek
1.	Terjadi arus pendek pada perangkat SES.	• *Human error.* • Kesalahan dalam perakitan.	Terjadi kerusakan pada perangkat SES.
2.	ESP32 tidak dapat terhubung dengan Internet.	• Terjadi kesalahan pada bagian kode. • Jaringan yang kurang mumpuni.	Perangkat SES tidak dapat berjalan dengan optimal.
3.	Aplikasi tidak dapat terhubung dengan Internet	• Jaringan yang kurang mumpuni. • Aplikasi tidak memiliki izin untuk mengakses Internet.	Sistem SES tidak dapat berjalan dengan optimal.
4.	Aplikasi mengalami *forced closed* saat digunakan.	• Terdapat kesalahan pada kode aplikasi. • Perangkat yang digunakan kurang mumpuni.	Aplikasi tidak dapat dijalankan.
5.	Aplikasi dan perangkat SES tidak dapat terhubung.	• Jaringan yang kurang mumpuni. • Terjadi kesalahan di pengaturan jaringan.	Sistem SES tidak dapat digunakan dengan optimal.

3.6 Analisis Teknologi

Berikut ini merupakan perangkat lunak dan perangkat keras yang digunakan selama pengembangan Tugas Akhir ini.

Tabel 3.3 Daftar Perangkat Yang Digunakan

I. Perangkat Lunak	
Android Studio IDE (vGirrafe 2022.3.1)	Digunakan sebagai *text editor* dan *compiler* utama dalam pengembangan aplikasi.
Kotlin (v2.1.0)	Bahasa pemrograman yang digunakan dalam pengembangan aplikasi.
Firebase (v14.0.0)	Digunakan sebagai basis data utama untuk menyimpan data dan informasi.
Draw.IO (v27.0.9)	Digunakan untuk pemodelan.
Figma (v124.4.7)	Digunakan untuk merancang antarmuka aplikasi.
II. Perangkat Keras	
ESP 32	Mikrokontroler untuk mengatur perangkat SES.
PZEM-004T	Sensor tegangan untuk membaca arus tegangan listrik.
Solid State Relay	Untuk memutuskan dan menyalurkan aliran listrik ke perangkat elektronik.
Laptop	HP Pavillion (*Processor*: *Intel Core* I7 Gen 10, RAM: 16 Gb, ROM: NvMe *Solid State Drive* 512 Gb.)

3.7 Analisis Penerapan Algoritma *Random Forest*

Untuk melakukan prediksi total konsumsi energi listrik pada akhir bulan, akan digunakan algoritma *Random Forest Regression* dengan teknik *Bootstrap Sampling*. Prediksi akan dilakukan berdasarkan data yang diperoleh dari hasil konsumsi harian pengguna selama bulan berjalan. Berikut ini adalah simulasi alur penerapan algoritma *Random Forest*:

Langkah pertama yang dilakukan merupakan proses analisis data konsumsi pengguna selama bulan berjalan.

1. Analisis data konsumsi pengguna di bulan berjalan

Tabel 3.4 Data Konsumsi Harian

NO.	Hari / Tanggal	Energi listrik yang digunakan (kWh)
1.	Minggu / 1 Juni 2025	0,00098
2.	Senin / 2 Juni 2025	0,00121
3.	Selasa / 3 Juni 2025	0,00078
4.	Rabu / 4 Juni 2025	0,00078
5.	Kamis / 5 Juni 2025	0,00089

Total Konsumsi	0,00464

2. *Sampling* data historis dari 3 bulan terakhir:

Tabel dibawah ini berisikan data konsumsi energi listrik pengguna selama 3 bulan terakhir mulai dari bulan maret hingga bulan mei.

Tabel 3.5 Data Konsumsi Historis

Tanggal	Maret	April	Mei
1	0,00081	0,00081	0,00081
2	0,00085	0,00082	0,00071
3	0,00082	0,00111	0,00091
4	0,00084	0,00081	0,00041
5	0,00081	0,00032	0,00121
6	0,00089	0,00091	0,00056
7	0,00088	0,00081	0,00081
8	0,00082	0,00099	0,00071
9	0,00083	0,00091	0,00067
10	0,00087	0,00081	0,00091
11	0,00088	0,00089	0,00051
12	0,00089	0,00089	0,00085
13	0,00088	0,00111	0,00071
14	0,00081	0,00092	0,00071
15	0,00085	0,00041	0,00091
16	0,00084	0,00111	0,00081
17	0,00084	0,00091	0,00064
18	0,00085	0,00091	0,00091
19	0,00087	0,00088	0,00123
20	0,00082	0,00101	0,00043
21	0,00082	0,00091	0,00081
22	0,00084	0,00089	0,00088
23	0,00084	0,00089	0,00088
24	0,00089	0,00091	0,00091
25	0,00081	0,00098	0,00081
26	0,00086	0,00099	0
27	0,00087	0,00081	0,00012

Tanggal	Maret	April	Mei
28	0,00089	0,00098	0,00089
29	0,00089	0,00087	0,00089
30	0,00098	0,00089	0,00089
31	0,00089	-	0,00091
Total Konsumsi	0,02466	0,02646	0,02341
Rata-rata Konsumsi 3 Bulan		0,00083	

Tabel dibawah ini merupakan data yang di sampel yang diambil dari data konsumsi historis pengguna di Tabel 3.5:

Tabel 3.6 Sampel Data Konsumsi Historis

Bulan	Data Konsumsi 7 Hari Pertama	Total Konsumsi 15 Hari	Total Konsumsi Bulanan
Maret (31 Hari)	[0,00081, 0,00085, 0,00082, 0,00084, 0,00081, 0,00089, 0,00089]	0,01273 kWh	0,02466 kWh
April (30 Hari)	[0,00081, 0,00082, 0,00111, 0,00081, 0,00032, 0,00091, 0,00081]	0,01252 kWh	0,02646 kWh
Mei (31 Hari)	[0,00081, 0,00071, 0,00091, 0,00041, 0,00121, 0,00056, 0,00081]	0,0114 kWh	0,02341 kWh
Rata-rata konsumsi 15 hari		0,01221	

3. Ekstraksi fitur untuk prediksi:

a. Fitur 1: Total energi yang digunakan: 0,00464 kWh

Total energi yang digunakan diperoleh dari hasil penjumlahan data konsumsi energi listrik harian pengguna. Data bisa dilihat di Tabel 3.4.

$$Total\ Energy\ Used: 0{,}00098 + 0{,}00121 + 0{,}00078 + 0{,}00078 + 0{,}00089 = 0{,}00464\ kWh$$

b. Fitur 2: Rata-rata konsumsi harian: 0,00092 kWh

Rata-rata konsumsi harian diperoleh dari hasil pembagian antara total konsumsi energi listrik pengguna di bulan berjalan dengan jumlah hari yang digunakan untuk prediksi. Data bisa dilihat di Tabel 3.4.

$$Average\ Daily\ Usage: 0{,}00464 \div 5 = 0{,}000928 \text{ kWh}$$

c. Fitur 3: Kemajuan bulan: 16,7%

Kemajuan bulan diperoleh dari hasil pembagian antara jumlah data dalam bulan berjalan dibagi jumlah hari dalam 1 bulan. Data bisa dilihat di Tabel 3.4.

$$Month's\ Progress: 5 \div 30 = 0{,}167 \times 100 = 16{,}7\%$$

d. Fitur 4: Sisa hari: 25 Hari

Sisa hari diperoleh dari hasil pengurangan jumlah hari dalam 1 bulan dengan jumlah data prediksi.

$$Remaining\ Days: 30 - 5 = 25 \text{ Hari}$$

e. Fitur 5: Standar Deviasi: 0,0000174

Standar deviasi diperoleh dari hasil perhitungan varians, yang digunakan untuk mengukur penyebaran data dari nilai rata-rata. Data bisa dilihat di Tabel 3.4 dan hasil ekstraksi fitur 2.

$s = \sqrt{[(0{,}00098 - 0{,}000928)^2 + (0{,}00121 - 0{,}000928)^2 + (0{,}00078 - 0{,}000928)^2 + (0{,}00078 - 0{,}000928)^2 + (0{,}00089 - 0{,}000928)^2] / 5} = 0{,}0000174$

f. Fitur 6: Posisi bulan: 1.0

Posisi bulan diperoleh dari hasil perbandingan antara tanggal dilakukannya prediksi dengan rules yang telah ditentukan.

Tanggal prediksi <= 10 = 1.0

Tanggal prediksi >10 - < 20 = 2.0

Tanggal prediksi >= 20 = 3.0

g. Fitur 7: Perkembangan energi: - 0,254

Perkembangan energi diperoleh dengan membandingkan total konsumsi pada awal data dan akhir data.

Rata-rata 2 hari pertama: [0,00098, 0,00121] = 0,00109

Rata-rata 3 hari terakhir: [0,00078, 0,00078, 0,00089] = 0,00081

Perkembangan energi = (0,00081 – 0,00109) / 0,00109 = -0,254

h. Hasil ekstraksi dari fitur 1-7: [0,00464, 0,00092, 0,167, 25.0, 0,0000174, 1.0, - 0,254]

4. Latihan menggunakan sampel data

Dari data historis akan dibuat sampel data untuk latihan. Tabel 3.7 berisikan gambaran sampel yang dipilih secara acak sebanyak 70% dari total data historis.

Tabel 3.7 Daftar Sampel

Sampel	Hari / Tanggal
Sampel 1	Jumat, 7 Maret 2025
Sampel 2	Sabtu, 15 Maret 2025
Sampel 3	Senin, 7 April 2025
Sampel 4	Selasa, 15 April 2025
Sampel 5	Rabu, 7 Mei 2025
Dan seterusnya...	

Tabel 3.8 Hasil Ekstraksi Fitur Sampel

Fitur	Sampel 1	Sampel 2	Sampel 3	Sampel 4	Sampel 5
Fitur 1	0,00591	0,01273	0,00559	0,01252	0,00542
Fitur 2	0,00084	0,00050	0,00079	0,00083	0,00077
Fitur 3	0,226	0,484	0,233	0,5	0,226
Fitur 4	24.0	16.0	24.0	16.0	24.0
Fitur 5	0.000163	0,000168	0,00022	0,00049	-0,00013
Fitur 6	1.0	2.0	1.0	2.0	1.0
Fitur 7	0,05	0,02	0,05	0,02	0,05

5. Simulasi dengan *Decision Tree*

Selanjutkan akan dilakukan simulai dengan *Decision Tree* dengan sampel yang telah dipilih sebelumnya. Contoh sampel yang digunakan dapat dilihat di Tabel 3.8.

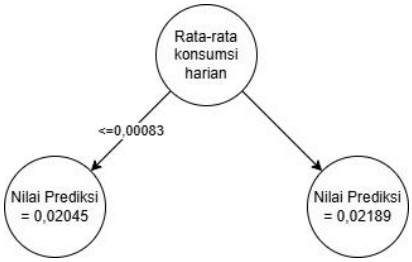
a. Pohon 1 (Rata-rata konsumsi harian)

Dalam simulasi ini pohon satu akan melakukan prediksi menggunakan salah satu fitur yaitu rata-rata konsumsi harian pengguna.

Rata-rata konsumsi harian untuk 3 bulan (Data dapat dilihat di Tabel 3.5): 0,00083

i. Kiri (<= 0,00083) = Dengan menggunakan sampel secara acak akan diperoleh rata-rata target prediksi yaitu 0.02045
ii. Kanan (> 0,00083) = Dengan menggunakan sampel secara acak akan diperoleh rata-rata target prediksi yaitu 0,02189

Prediksi untuk bulan Juni = Rata-rata konsumsi harian = 0,00092 (Data dapat dilihat pada hasil ekstraksi fitur 2 pada halaman 14) maka jalur yang tepat merupakan Kanan (> 0,00083) sehingga prediksi total konsumsi untuk bulan Juni dari pohon 1 adalah 0,02189.



Gambar 3.1 Pohon 1

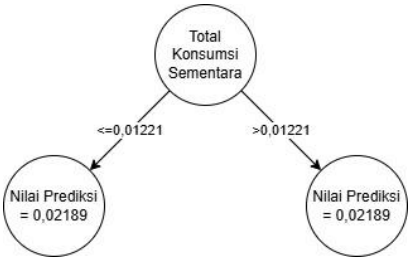
b. Pohon 2 (Total konsumsi sementara)

Dalam simulasi ini pohon dua akan melakukan prediksi menggunakan salah satu fitur yaitu total konsumsi sementara pengguna.

Rata-rata total konsumsi 15 hari untuk 3 bulan (Data dapat dilihat di Tabel 3.6): 0,01221

i. Kiri (<= 0,01221) = Dengan menggunakan sampel secara acak akan diperoleh rata-rata target prediksi yaitu 0.02189
ii. Kanan (> 0,01221) = Dengan menggunakan sampel secara acak akan diperoleh rata-rata target prediksi yaitu 0,02189

Prediksi untuk bulan Juni = Rata-rata total konsumsi sementara = 0,00464 (Data dapat dilihat pada hasil ekstraksi fitur 1 pada halaman 14) maka jalur yang tepat merupakan Kiri (<= 0,01) sehingga prediksi total konsumsi untuk bulan Juni dari pohon 2 adalah 0,02189.



Gambar 3.2 Pohon 2

Proses ini akan berlangsung sampai dengan jumlah pohon memenuhi jumlah yang ditentukan yaitu 20. Fitur dan sampel yang digunakan untuk masing-masing pohon akan dipilih secara acak oleh sistem.

6. Simulasi dengan *Random Forest* (20 Pohon)

Setelah pemprosesan dengan *Decision Tree* telah berjumlah 20 pohon selanjutkan akan dicari median dari 20 hasil prediksi tersebut untuk memperoleh hasil prediksi akhir untuk total konsumsi energi listrik pengguna di bulan Juni.

Tabel 3.9 Daftar Hasil Prediksi Setiap Pohon

Pohon	Hasil Prediksi
Pohon 1	0,02189
Pohon 2	0,02189
Dan seterusnya...	
Pohon 20	0,02189

7. Hasil prediksi akhir

Proses perolehan prediksi akhir akan berlangsung dengan pengambilan nilai tengah (median) dari rangkaian hasil simulasi dengan *Random Forest* pada Tabel 3.9. Dengan anggapan nilai median = 0,02189, maka dapat ditarik kesimpulan sebagai berikut.

$$\text{Prediksi Harian: } 0{,}02189 - 0{,}00464 = 0{,}01725 \div 25 = 0{,}00069\ kWh$$

$$\text{Total Prediksi} = 0{,}00069\ \times 30 = 0{,}02072\ kWh$$

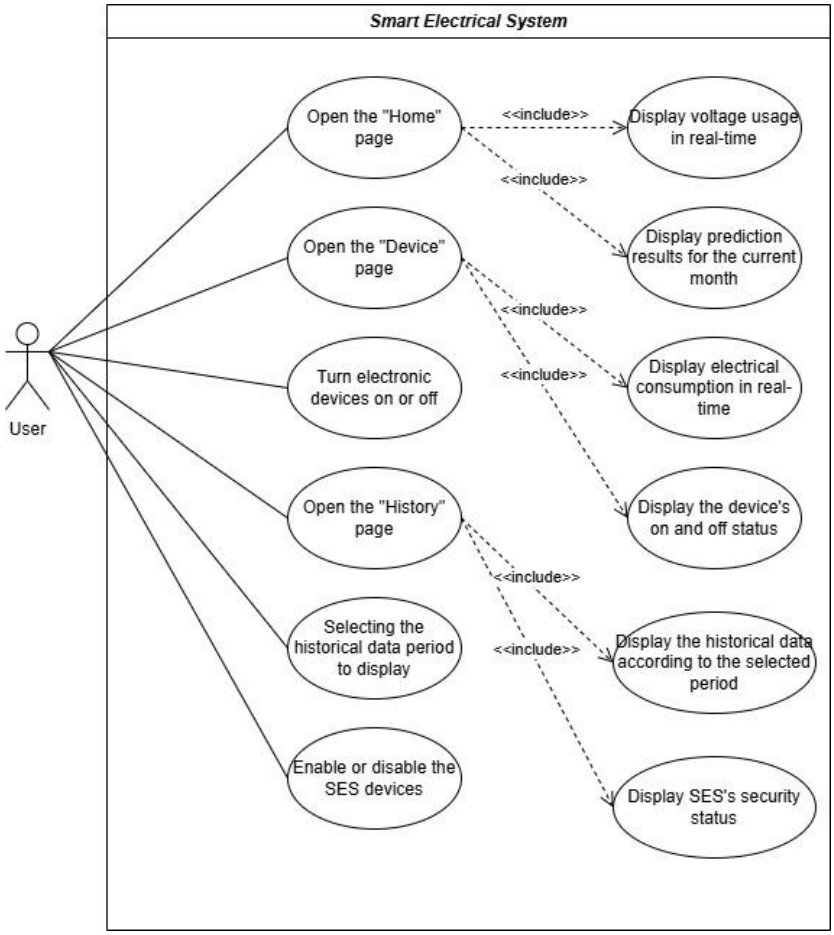
Dari hasil yang diperoleh dapat diketahui bahwa prediksi total konsumsi energi listrik pengguna selama bulan Juni adalah 0,02072 kWh.

BAB IV
PERANCANGAN

Pada bagian ini akan dijabarkan rancangan pembangunan sistem mulai dari *Use Case diagram, Class Diagram, Activity Diagram*, rancangan basis data, rancangan alat, dan rancangan tampilan antarmuka.

4.1 *Use Case Diagram* Sistem

Berikut ini merupakan gambaran dari fungsionalitas dari aplikasi yang dirancang, dibuat dalam bentuk *Use Case Diagram*.



Gambar 4.1 *Use Case Diagram* Aplikasi

Berikut ini merupakan *Use Case Table* yang berisikan informasi tentang modul yang dimiliki.

Tabel 4.1 *Use Case* Membuka Halaman "*Home*"

Nama *Use Case*	Membuka halaman "*Home*".	
Aktor	*User*	
Deskripsi	Pengguna membuka aplikasi SES yang telah terinstall pada *smartphone* pengguna.	
Pre-Condition	- Aplikasi SES telah terinstall pada *smartphone* pengguna.	
Normal Course	***User***	**Sistem**
	User menjalankan aplikasi SES.	Mengisialisasi aplikasi dan memulai fungsi pada halaman "*Home*".
Alternate Course	***User***	**Sistem**
	User tidak menjalankan aplikasi SES.	Sistem tidak menginisialisasi aplikasi.
Post-Condition	Aplikasi akan diinisialisasi dan fungsi pada halaman "*Home*" akan berjalan.	

Tabel 4.2 *Use Case* Menampilkan Data Penggunaan Voltase Secara *Realtime*

Nama *Use Case*	Menampilkan data penggunaan voltase secara *realtime*.	
Aktor	*User*	
Deskripsi	Sistem akan menampilkan data penggunaan voltase secara *realtime* dalam bentuk grafik pada antarmuka aplikasi.	
Pre-Condition	- Aplikasi SES telah terinstall pada *smartphone* pengguna. - Aplikasi berhasil terinisialisasi dengan baik. - Aplikasi memiliki akses Internet. - Aplikasi memiliki akses ke Firebase.	
Normal Course	***User***	**Sistem**
	User menjalankan aplikasi SES.	Menampilkan data penggunaan voltase secara *realtime* menggunakan grafik pada antarmuka aplikasi.
Alternate Course	***User***	**Sistem**
	User tidak menjalankan aplikasi SES.	Sistem tidak menjalankan fungsi untuk menampilkan nilai voltase secara *realtime.*
Post-Condition	Data penggunaan voltase perangkat akan ditampilkan pada	

	antarmuka aplikasi dalam bentuk grafik secara *realtime*.

Tabel 4.3 *Use Case* Menampilkan Total Prediksi Untuk Bulan Berjalan

Nama *Use Case*	Menampilkan total prediksi untuk bulan berjalan.	
Aktor	*User*	
Deskripsi	Sistem akan menampilkan hasil prediksi total konsumsi energi listrik untuk bulan berjalan menggunakan algoritma *Random Forest* pada antarmuka aplikasi.	
Pre-Condition	- Aplikasi SES telah terinstall pada *smartphone* pengguna. - Aplikasi berhasil terinisialisasi dengan baik. - Aplikasi memiliki akses Internet. - Apliakasi memiliki akses ke Firebase.	
Normal Course	***User***	**Sistem**
	User menjalankan aplikasi SES.	Memproses nilai prediksi total konsumsi energi listrik untuk bulan berjalan menggunakan data konsumsi harian pengguna dan menampilkan hasilnya pada antarmuka aplikasi.
Alternate Course	***User***	**Sistem**
	User tidak menjalankan aplikasi SES.	Sistem tidak menjalankan fungsi untuk memprediksi total konsumsi energi listrik untuk bulan berjalan.
Post-Condition	Hasil prediksi total konsumsi energi listrik pengguna untuk bulan berjalan menggunakan *Random Forest* tertampil pada antarmuka aplikasi.	

Tabel 4.4 *Use Case* Membuka Halaman "*Device*"

Nama *Use Case*	Membuka halaman "*Device*".
Aktor	*User*
Deskripsi	Pengguna membuka aplikasi SES yang telah terinstall pada *smartphone* pengguna dan melakukan navigasi ke halaman "*Device*".
Pre-Condition	- Aplikasi SES telah terinstall pada *smartphone* pengguna.

	- Aplikasi berhasil terinisialisasi dengan baik.	
Normal Course	***User***	**Sistem**
	User menjalankan aplikasi SES dan melakukan navigasi ke halaman "*Device*".	Memulai fungsi pada halaman "*Device*".
Alternate Course	***User***	**Sistem**
	User tidak melakukan navigasi ke halaman "*Device*".	Sistem tidak menjalankan fungsi pada halaman "*Device*".
Post-Condition	Pengguna dibawa kehalaman "*Device*" dan fungsi pada halaman "*Device*" juga akan terekseskusi secara otomatis.	

Tabel 4.5 *Use Case* Menampilkan Konsumsi Listrik Perangkat Secara *Realtime*

Nama *Use Case*	Menampilkan konsumsi listrik perangkat secara *realtime*.	
Aktor	*User*	
Deskripsi	Sistem akan menampilkan data konsumsi listrik yang digunakan perangkat elektronik seperti *Voltage*, *Current*, *Ampere*, dan *Energy* secara *realtime*.	
Pre-Condition	- Pengguna telah menjalankan aplikasi SES. - Pengguna melakukan navigasi ke halaman "*Device*".	
Normal Course	***User***	**Sistem**
	User menjalankan aplikasi SES dan melakukan navigasi ke halaman "*Device*".	Memulai fungsi untuk menampilkan data konsumsi energi listrik secara *realtime* untuk masing-masing perangkat.
Alternate Course	***User***	**Sistem**
	User tidak melakukan navigasi ke halaman "*Device*".	Sistem tidak menjalankan fungsi untuk menampilkan data konsumsi energi listrik.
Post-Condition	Sistem menampilkan data konsumsi energi listrik untuk masing-masing perangkat SES.	

Tabel 4.6 *Use Case* Menampilkan Status Nyala dan Matinya Perangkat

Nama *Use Case*	Menampilkan status nyala dan matinya perangkat.
Aktor	*User*
Deskripsi	Sistem akan menampilkan status aliran listrik dari *relay* ke

	perangkat elektronik. Apabila pengguna memilih status nyala, maka *relay* akan menyalurkan aliran listrik ke perangkat elektronik, namun jika pengguna memilih status mati, maka *relay* akan membatasi aliran listrik ke perangkat elektronik.	
Pre-Condition	- Pengguna telah menjalankan aplikasi SES. - Pengguna melakukan navigasi ke halaman "*Device*".	
Normal Course	***User***	**Sistem**
	User menjalankan aplikasi SES dan melakukan navigasi ke halaman "*Device*".	Sistem menampilkan status relay pada masing-masing perangkat SES yang terhubung.
Alternate Course	***User***	**Sistem**
	User tidak melakukan navigasi ke halaman "*Device*".	Sistem tidak menjalankan fungsi untuk menampilkan status relay dari masing-masing perangkat.
Post-Condition	Pengguna dibawa kehalaman "*Device*" dan fungsi pada halaman "*Device*" juga akan terekseskusi secara otomatis.	

Tabel 4.7 *Use Case* Menyalakan atau Mematikan Perangkat Elektronik

Nama *Use Case*	Menyalakan atau mematikan perangkat elektronik	
Aktor	*User*	
Deskripsi	Pengguna dapat memilih untuk menyalakan atau mematikan perangkat elektronik.	
Pre-Condition	- Pengguna telah menjalankan aplikasi SES. - Pengguna melakukan navigasi ke halaman "*Device*". - Aplikasi terhubung dengan Internet. - Aplikasi terhubung dengan Firebase. - Perangkat SES terhubung dengan jaringan Internet.	
Normal Course	***User***	**Sistem**
	User memilih untuk menyalakan atau mematikan perangkat elektronik.	Sistem memperbarui status *relay* di Firebase untuk menyalakan atau mematikan perangkat elektronik.
Alternate Course	***User***	**Sistem**

	User memilih untuk tidak menyalakan atau mematikan perangkat elektronik	Sistem tidak menjalankan fungsi untuk menampilkan status *relay* dari masing-masing perangkat.
Post-Condition	Status perangkat SES di Firebase disesuaikan dengan status pada aplikasi untuk menyalakan atau mematikan perangkat elektronik.	

Tabel 4.8 *Use Case* Membuka Halaman "*History*"

Nama *Use Case*	Membuka halaman "*History*"	
Aktor	*User*	
Deskripsi	Pengguna membuka aplikasi SES yang telah terinstall pada *smartphone* pengguna dan melakukan navigasi ke halaman "*History*".	
Pre-Condition	- Pengguna telah menjalankan aplikasi SES. - Pengguna melakukan navigasi ke halaman "*History*".	
Normal Course	***User***	**Sistem**
	User menjalankan aplikasi SES dan melakukan navigasi ke halaman "*History*".	Memulai fungsi pada halaman "*Device*".
Alternate Course	***User***	**Sistem**
	User memilih untuk tidak melakukan navigasi ke halaman "*History*".	Sistem tidak menjalankan fungsi pada halaman "*History*".
Post-Condition	Pengguna dibawa kehalaman "*History*" dan fungsi pada halaman "*History*" juga akan terekseskusi secara otomatis.	

Tabel 4.9 *Use Case* Menampilkan Data Historis Konsumsi Energi Listrik Sesuai Periode Yang Dipilih

Nama *Use Case*	Menampilkan data historis konsumsi energi listrik sesuai periode yang dipilih
Aktor	*User*
Deskripsi	Sistem akan menampilkan data historis konsumsi energi listrik pengguna berdasarkan periode waktu yang dipilih.
Pre-Condition	- Pengguna telah menjalankan aplikasi SES. - Pengguna melakukan navigasi ke halaman "*History*". - Aplikasi terhubung dengan Internet.

	- Aplikasi terhubung dengan Firebase.	
Normal Course	***User***	**Sistem**
	User menjalankan aplikasi SES dan melakukan navigasi ke halaman "*History*".	Menampilkan data historis konsumsi energi listrik pengguna berdasarkan periode waktu yang dipilih.
Alternate Course	***User***	**Sistem**
	User memilih untuk tidak melakukan navigasi ke halaman "*History*".	Sistem tidak menjalankan fungsi untuk menampilkan data historis konsumsi energi listrik pengguna.
Post-Condition	Data historis konsumsi energi listrik pengguna ditampilkan dalam bentuk grafik pada antarmuka aplikasi sesuai dengan periode waktu yang dipilih.	

Tabel 4.10 *Use Case* Menampilkan Status Keamanan Perangkat

Nama *Use Case*	Menampilkan status keamanan perangkat	
Aktor	*User*	
Deskripsi	Pengguna membuka aplikasi SES yang telah terinstall pada *smartphone* pengguna dan melakukan navigasi ke halaman "*History*".	
Pre-Condition	- Pengguna telah menjalankan aplikasi SES. - Pengguna melakukan navigasi ke halaman "*History*". - Aplikasi terhubung dengan Internet. - Aplikasi terhubung dengan Firebase.	
Normal Course	***User***	**Sistem**
	User menjalankan aplikasi SES dan melakukan navigasi ke halaman "*History*".	Menampilkan status keamanan perangkat SES.
Alternate Course	***User***	**Sistem**
	User memilih untuk tidak melakukan navigasi ke halaman "*History*".	Sistem tidak menjalankan fungsi untuk menampilkan status keamanan perangkat SES.
Post-Condition	Pengguna dibawa kehalaman "*History*" dan fungsi pada halaman	

	"*History*" juga akan terekseskusi secara otomatis.

Tabel 4.11 *Use Case* Memilih Periode Waktu Data Historis Untuk Ditampilkan

Nama *Use Case*	Memilih periode waktu dari data historis yang ingin ditampilkan.	
Aktor	*User*	
Deskripsi	Pengguna akan memilih periode waktu dari data historis yang ingin ditampilkan, baik itu harian, mingguan, maupun bulanan, serta tanggal spesifik yang ingin ditampilkan.	
Pre-Condition	- Pengguna telah menjalankan aplikasi SES. - Pengguna melakukan navigasi ke halaman "*History*". - Aplikasi terhubung dengan jaringan Internet. - Aplikasi terhubung dengan Firebase.	
Normal Course	***User***	**Sistem**
	User menjalankan aplikasi SES dan melakukan navigasi ke halaman "*History*".	Menampilkan data historis konsumsi energi listrik pengguna berdasarkan periode waktu yang dipilih.
Alternate Course	***User***	**Sistem**
	User memilih untuk tidak melakukan navigasi ke halaman "*History*".	Sistem tidak menjalankan fungsi untuk menampilkan data historis konsumsi energi listrik pengguna.
Post-Condition	Data konsumsi energi listrik akan ditampilkan pada halaman antarmuka aplikasi sesuai dengan periode waktu yang dipilih pengguna.	

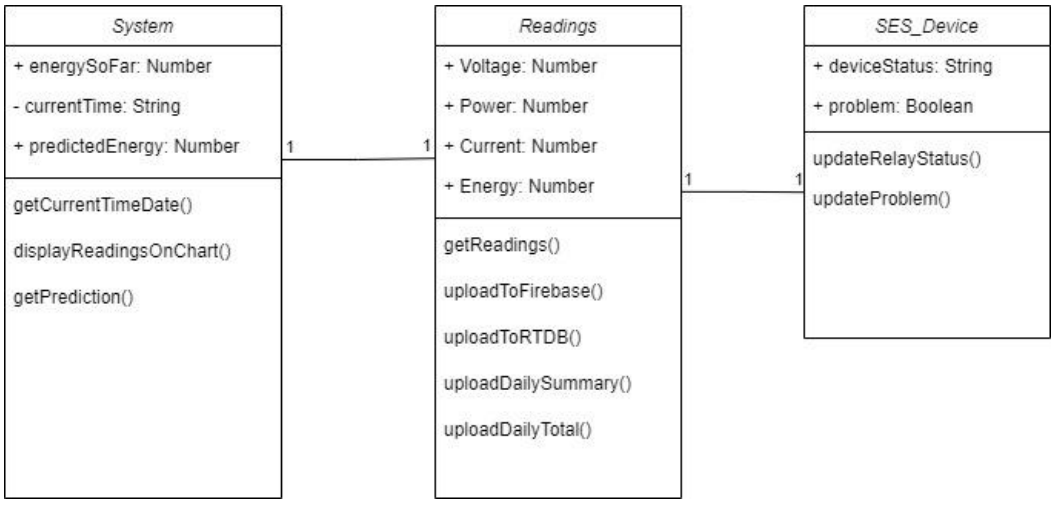
Tabel 4.12 *Use Case* Mengaktifkan atau Menonaktifkan Perangkat SES

Nama *Use Case*	Mengaktifkan atau menonaktifkan perangkat SES	
Aktor	*User*	
Deskripsi	Pengguna dapat memilih untuk menaktifkan atau menonaktifkan perangkat SES dari aplikasi. Dengan menonaktifkan perangkat SES, perangkat SES akan tetap dalam kondisi mati meskipun status perangkat bertuliskan "*ONLINE*".	
Normal Course	***User***	**Sistem**
	*User m*enekan *box* yang berisikan informasi masalah	Memperbarui status masalah pada perangkat SES di

	pada perangkat SES kemudian menekan *button* "*Click to Turn Off!*" untuk menonaktifkan perangkat SES atau "*Click to Turn On!*" untuk mengaktifkan perangkat SES.	Firebase.
Pre-Condition	- Pengguna telah menjalankan aplikasi SES. - Pengguna melakukan navigasi ke halaman "*History*". - Aplikasi terhubung dengan jaringan Internet. - Aplikasi terhubung dengan Firebase. - Perangkat SES terhubung dengan jaringan Internet.	
Post-Condition	Memperbarui status masalah pada perangkat SES di Firebase sesuai dengan pilihan pengguna.	

4.2 *Class Diagram* Sistem

Berikut ini merupakan rancangan pemodelan dari modul kelas yang digunakan menggunakan diagram UML, yaitu *Class Diagram*. Terdapat 3 kelas, yaitu *System*, *Readings*, dan *SES_Device*.

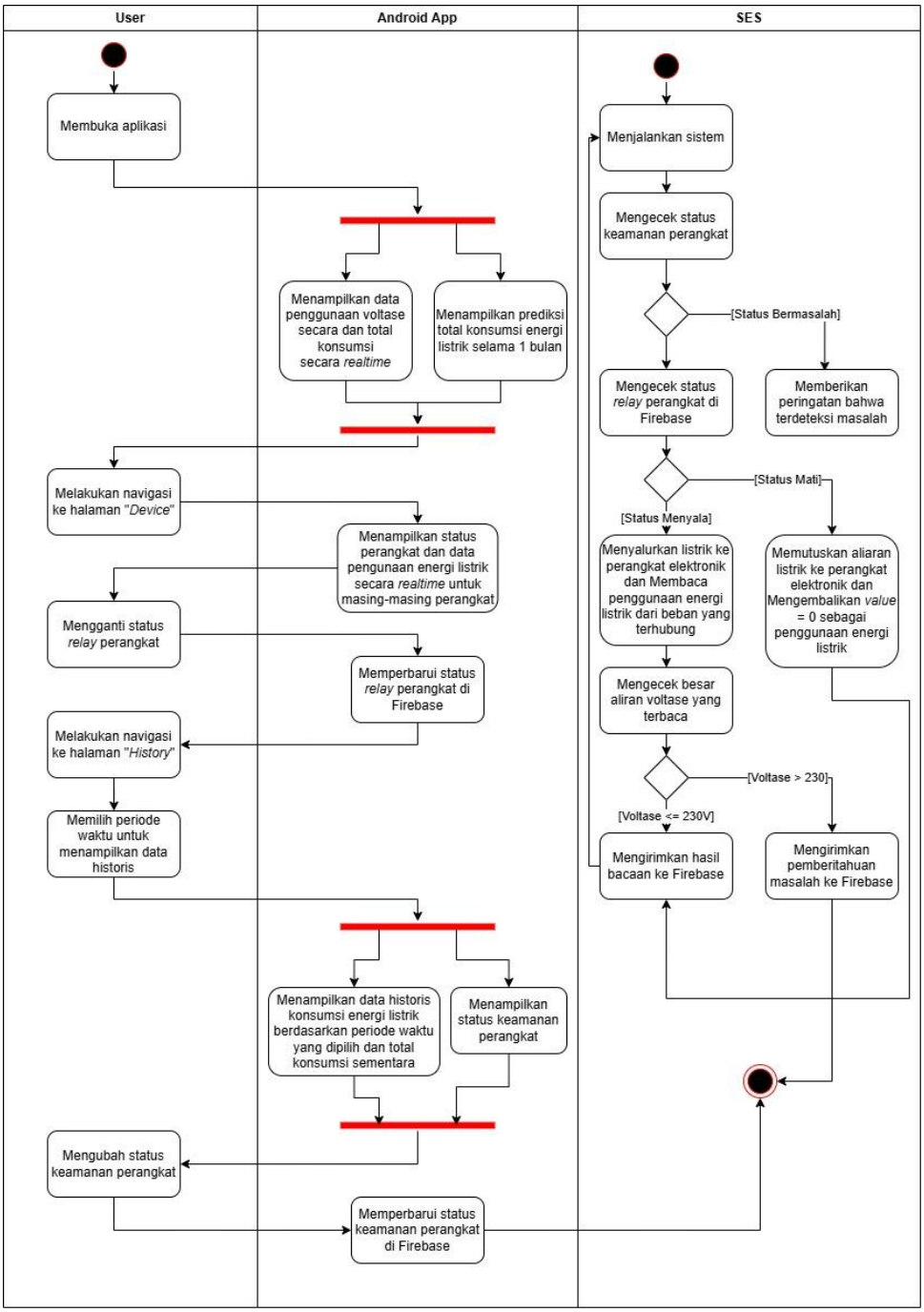


Gambar 4.2 *Class Diagram* Sistem

4.3 *Activity Diagram* Sistem

Gambar 4.3 merupakan gambaran dari *flow* dari sistem yang dirancang. Dimulai dari *user* menghubungkan perangkat SES ke aliran listrik dan menambahkan beban pada perangkat SES. Perangkat SES kemudian akan membaca tegangan arus listrik secara *realtime* yang akan dikirimkan ke *Realtime*

Database (RTDB) oleh Firebase. Aplikasi Android kemudian akan mengambil data dari RTDB yang kemudian ditampilkan pada antarmuka aplikasi.



Gambar 4.3 *Activity Diagram* Sistem

4.4 Kamus Data

Tabel 4.13 merupakan kamus data yang dibuat untuk menggambarkan data yang digunakan dalam Tugas Akhir ini. Firebase *Database* oleh Google merupakan basis data yang berfungsi untuk menyimpan data, terkhusus *Firestore* dan *Realtime Database*. *Firestore* merupakan sebuah NoSQL *database* dari

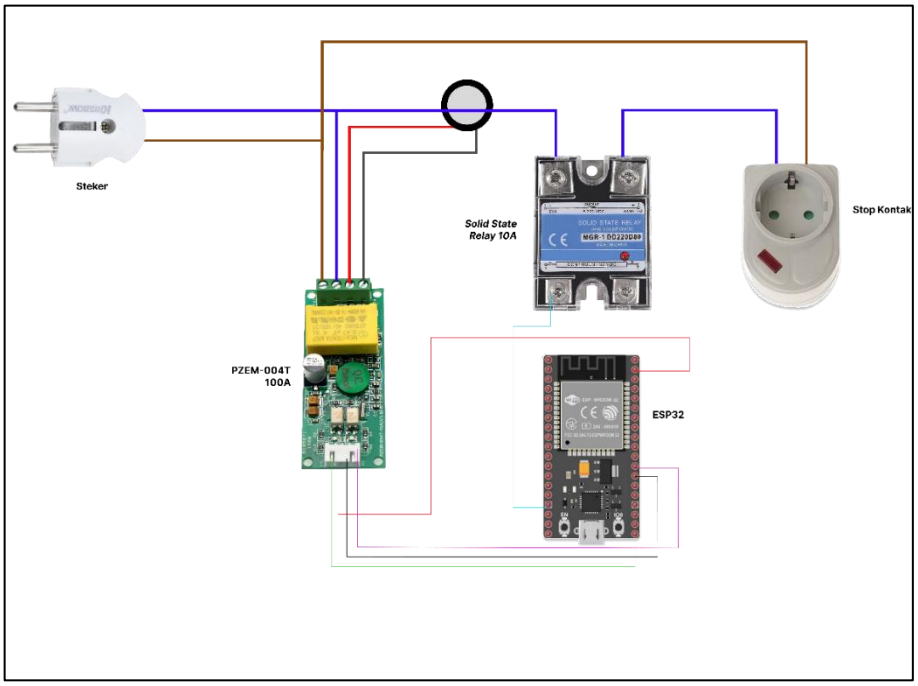
ekosistem Firebase berbasis *cloud* yang menyimpan data dalam bentuk koleksi yang memungkinkan pemodelan yang fleksibel. *Realtime Database* juga merupakan NoSQL *database* berbasis *cloud* dari ekosistem Firebase yang sesuai namanya memungkinkan pengguna untuk menyimpan dan menyinkronkan data secara *realtime* dengan setiap klien yang terhubung.

Tabel 4.13 Kamus Data

No	Tabel	Kolom	Tipe Data	Keterangan
1	*Realtime_data*	*Current*	*Number*	Berisikan nilai arus *realtime*
2	*Realtime_data*	*Energy*	*Number*	Berisikan nilai energi kumulatif
3	*Realtime_data*	*PeriodEnergy*	*Number*	Berisikan nilai energi *realtime*
4	*Realtime_data*	*Power*	*Number*	Berisikan nilai tenaga *realtime*
5	*Realtime_data*	*Problem*	*String*	Berisikan status keamanan perangkat SES
6	*Realtime_data*	*Status*	*String*	Berisikan status aliran listrik perangkat SES
7	*Realtime_data*	*voltage*	*Number*	Berisikan nilai voltase *realtime*
8	*Historical_data*	*Duration_so_far*	*Number*	Berisikan nilai durasi nyalanya perangkat SES dalam satuan detik
9	*Historical_data*	*Energy_so_far*	*Number*	Berisikan total konsumsi energi listrik harian
10	*Historical_data*	*Is_final*	*Boolean*	Berisikan status data
11	*Historical_data*	*Timestamp*	*Timestamp*	Berisikan waktu dan tanggal data

4.5 Rancangan Alat

Berikut ini merupakan rancangan alat untuk perangkat SES yang terdiri dari 1 buah steker, 1 buah sensor PZEM-004T, 1 buah mikrokontroler ESP32 *Wifi*/BT, 1 buah *Solid State Relay* 10A, dan 1 buah stop kontak. Dihubungkan menggunakan kabel listrik merek Eterna untuk aliran listrik utama antara steker dan stop kontak dan juga kabel *jumper* untuk mengirimkan sinyal antara PZEM-004T ke ESP32 dan ESP32 ke *Solid State Relay*.



Gambar 4.4 Rancangan Perangkat SES

Berikut ini merupakan rangkaian koneksi pin antar komponen yang digunakan, yaitu stop kontak, Steker, ESP32, PZEM-004T, dan *Solid State Relay* (SSR).

Tabel 4.14 Koneksi Pin Antar Komponen

Koneksi Pin		
ESP32	**PZEM-004T**	**Solid State Relay**
GPIO 16	TX Pin	-
GPIO 17	RX Pin	-
GPIO 12	-	*Positive Control Pin*
GND	-	*Negative Control Pin*
GND 2	GND Pin	-

Koneksi Pin		
ESP32	**PZEM-004T**	**Solid State Relay**
3.3V	VCC Pin	-

Berikut ini merupakan rangkaian untuk aliran listrik antar komponen yang digunakan.

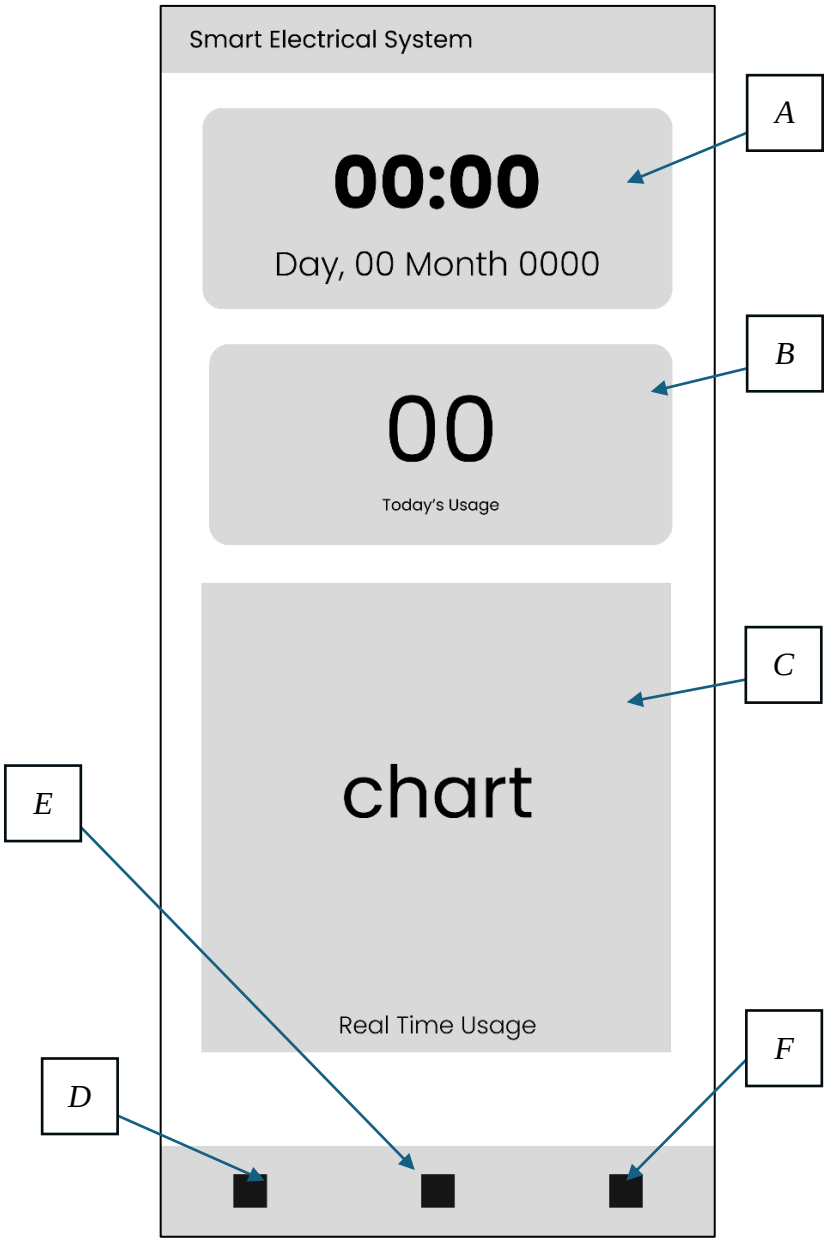
Tabel 4.15 Koneksi Aliran Listrik Antar Komponen

Rangkaian Aliran Listrik	
Steker *live wire*	PZEM-004T *live pin* & SSR *Negative Output Pin*
SSR *live wire*	Stop kontak *positive point*
Steker *neutral wire*	PZEM-004T *neutral pin* & Stop kontak *negative point*

Adapun *Current Transformer* (CT) merupakan salah satu komponen dari PZEM-004T. CT berfungsi untuk membaca tegangan arus listrik dan berfungsi dengan *live wire* yang diletakkan melalui CT tersebut. CT kemudian terhubung dengan PZEM-004T pada 2 pin khusus yang telah disediakan untuk CT.

4.6 Rancangan Antarmuka Aplikasi

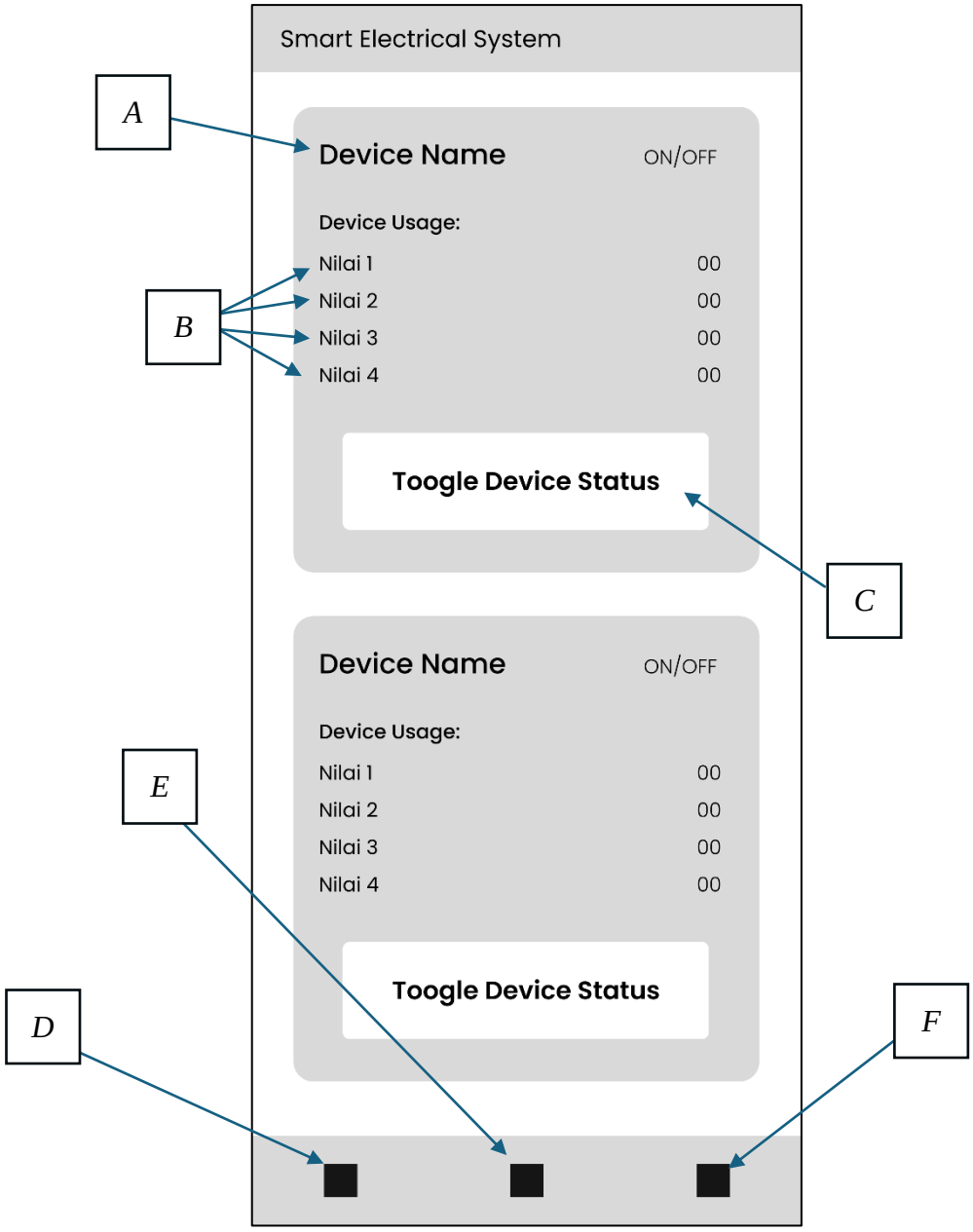
Berikut ini merupakan rancangan antarmuka dari aplikasi yang terdiri dari halaman *Home*, halaman *Device*, halaman *History,* dan *Pop Up* Masalah.



Gambar 4.5 Halaman *Home*

Keterangan:

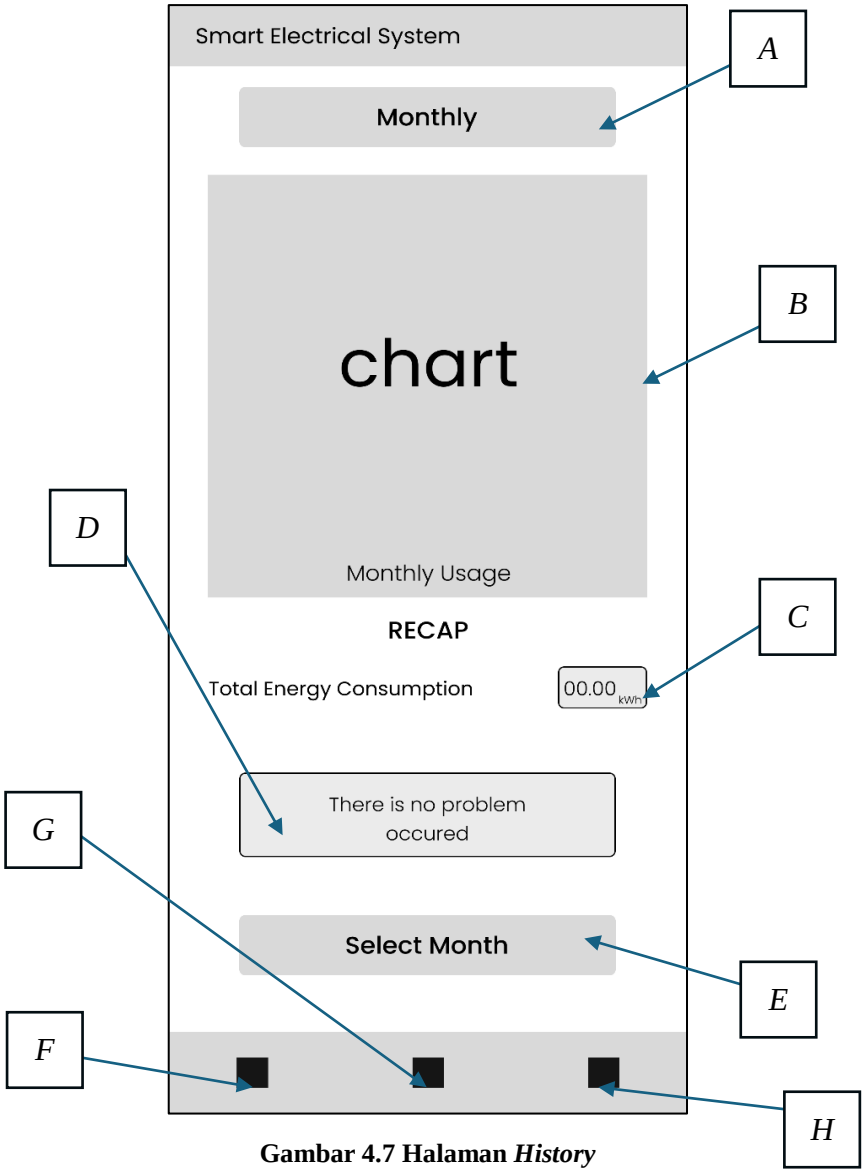
A. *Box* yang menggambarkan waktu dan tanggal secara *realtime*.
B. *Box* yang menggambarkan total konsumsi energi listrik untuk hari berjalan.
C. *Graph* yang menggambarkan penggunaan energi listrik secara *realtime*.
D. *Placeholder ImageButton* untuk ke halaman *Home*.
E. *Placeholder ImageButton* untuk ke halaman *Device*.
F. *Placeholder ImageButton* untuk ke halaman *History*.



Gambar 4.6 Halaman *Device*

Keterangan:

A. *Box* yang menggambarkan detail perangkat berupa nama dan status.
B. Nilai *realtime* yang diperoleh dari perangkat SES.
C. *Button* untuk men-*toggle* status aliran listrik.
D. *Placeholder ImageButton* untuk ke halaman *Home*.
E. *Placeholder ImageButton* untuk ke halaman *Device*.
F. *Placeholder ImageButton* untuk ke halaman *History*.

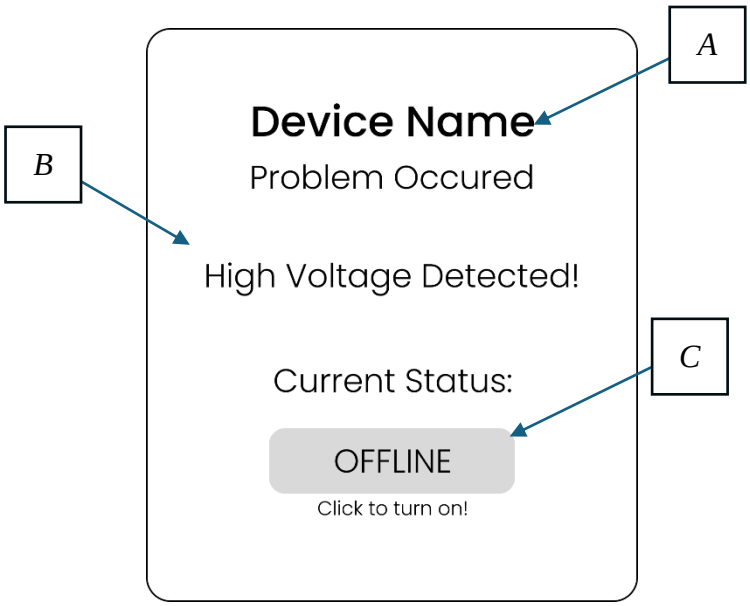


Gambar 4.7 Halaman *History*

Keterangan:

A. *Button* untuk memilih periode waktu yang akan ditampilkan.
B. *Graph* yang menggambarkan konsumsi energi listrik sesuai dengan periode waktu yang dipilih.
C. *Box* yang menunjukkan total konsumsi energi listrik.
D. *Box* yang berisikan informasi tentang anomali pada perangkat.
E. *Button* untuk memilih waktu yang ingin ditampilkan.
F. *Placeholder ImageButton* untuk ke halaman *Home*.
G. *Placeholder ImageButton* untuk ke halaman *Device*.

H. *Placeholder ImageButton* untuk ke halaman *History*.



Gambar 4.8 Tampilan *Pop Up*

Keterangan:

A. Nama perangkat yang mengalami masalah.
B. Keterangan masalah yang dialami perangkat.
C. *Button* untuk menghilangkan status masalah pada perangkat.

BAB V
IMPLEMENTASI

Bagian ini berisi proses implementasi dari rancangan alat dan aplikasi yang telah dibuat sebelumnya.

5.1 Lingkungan Implementasi

Berikut ini merupakan lingkungan implementasi yang dilakukan untuk perangkat keras SES dan juga aplikasi yang dikembangkan.

Tabel 5.1 Lingkungan Implementasi

NO	**Perangkat Lunak**	
1.	*Android Studio* IDE	*Compiler* dan *text editor* untuk membangun aplikasi berbasis Android.
2.	Firebase	Basis data yang menyimpan data penggunaan listrik.
NO	**Perangkat Keras**	
1.	ESP 32	Mikrokontroler untuk mengatur perangkat SES.
2.	PZEM-004T	Sensor yang digunakan untuk membaca arus tegangan listrik.
3.	*Solid State Relay*	Untuk memutuskan dan menyalurkan aliran listrik ke perangkat elektronik.

5.2 Aturan Implementasi

Berikut ini merupakan aturan implementasi yang terbagi menjadi 2, yaitu aturan implementasi pada alat dan aturan impelementasi pada aplikasi.

1. Aturan implementasi alat:

d. Alat yang dibuat harus terhubung pada aliran listrik aktif.
e. Alat elektronik yang ingin dipantau harus terhubung dengan alat yang dibuat.
f. Hubungan kabel *live* dan kabel *neutral* harus sesuai.
g. Alat harus terhubung dengan koneksi Internet aktif.

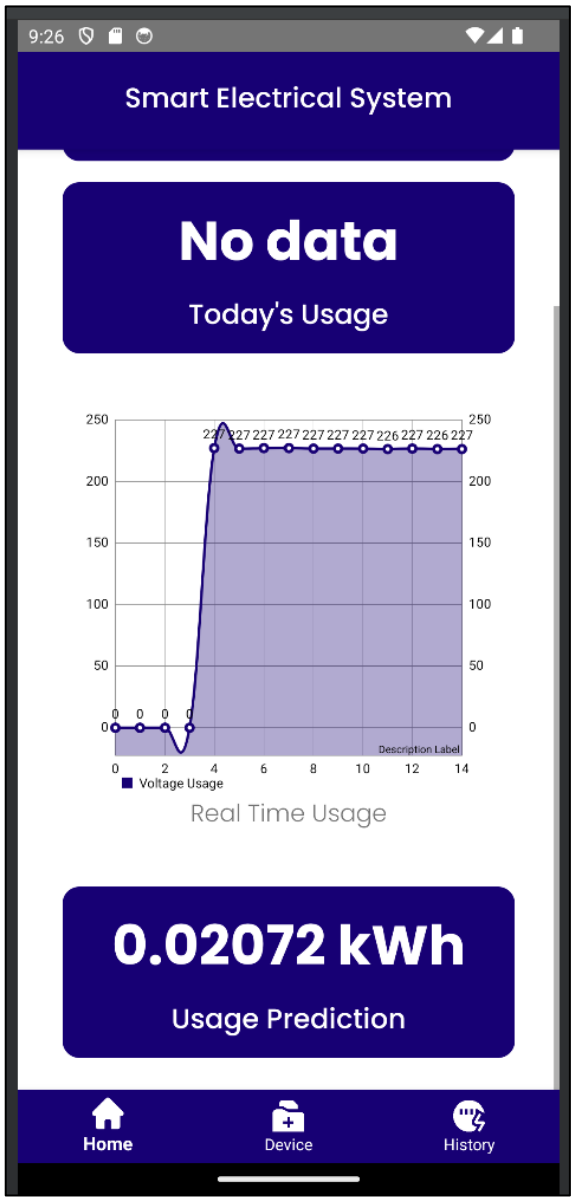
2. Aturan implementasi aplikasi:

   a. Perangkat harus terhubung dengan jaringan Internet yang stabil.

b. Aplikasi hanya dapat dijalankan pada perangkat Android versi 10 atau Android Q ke atas.

5.3 Implementasi Antarmuka Aplikasi

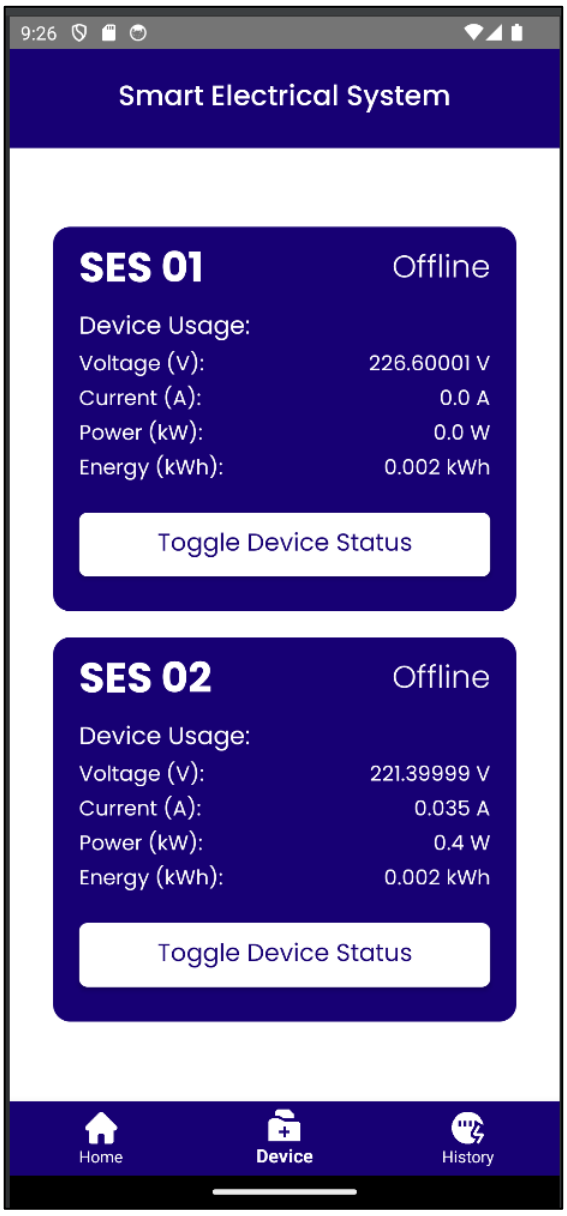
Berikut ini merupakan implementasi antarmuka dari aplikasi yang terdiri dari halaman *Home*, halaman *Device*, halaman *Report*, dan *pop up* Masalah



Gambar 5.1 Implementasi Halaman *Home*

Di halaman *Home* yang juga merupakan halaman utama dari aplikasi ini, pengguna dapat melihat data energi yang telah dikonsumsi sepanjang hari, dan

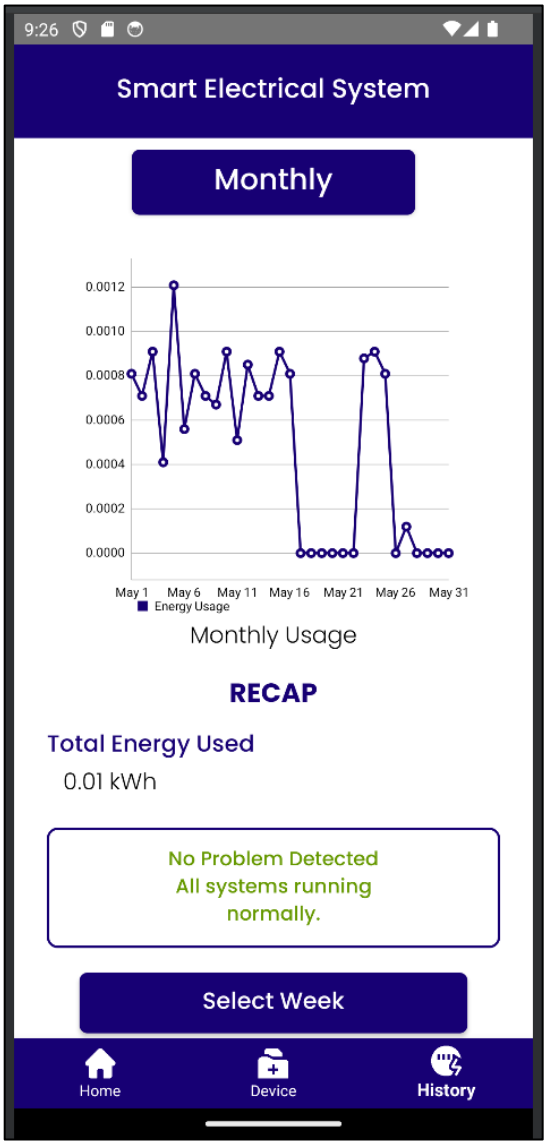
juga 15 nilai terakhir dari penggunaan voltase secara *realtime* yang ditampilkan dalam bentuk grafik. Pengguna juga dapat melihat nilai prediksi untuk perkiraan total konsumsi energi listrik pengguna selama sebulan.



Gambar 5.2 Implementasi Halaman *Device*

Di halaman *Device* pengguna dapat melihat data penggunaan energi listrik dari masing-masing perangkat SES. Terdapat nilai *Voltage* (V), *Current* (A), *Power* (kW), dan *Energy* (kWh). Terdapat juga status nyala dan matinya dari

masing-masing perangkat SES dan *Button "Toggle Device Status"* yang berfungsi untuk mengubah status nyala dan matinya perangkat SES.



Gambar 5.3 Implementasi Halaman *History*

Di halaman *History* pengguna dapat melihat data historis penggunaan energi listrik mereka untuk mingguan dan bulanan dan data penggunaan voltase untuk harian. Pengguna dapat menekan *button* bertuliskan "*Monthly*" untuk mengganti periode waktu yang ditampilkan. Data historis kemudian akan ditampilkan dalam bentuk grafik, nilai kumulatif energi listrik yang telah digunakan selama ini untuk mingguan dan bulanan, dan nilai rata-rata penggunaan voltase untuk harian. Selain itu akan ditampilkan status keamanan perangkat SES,

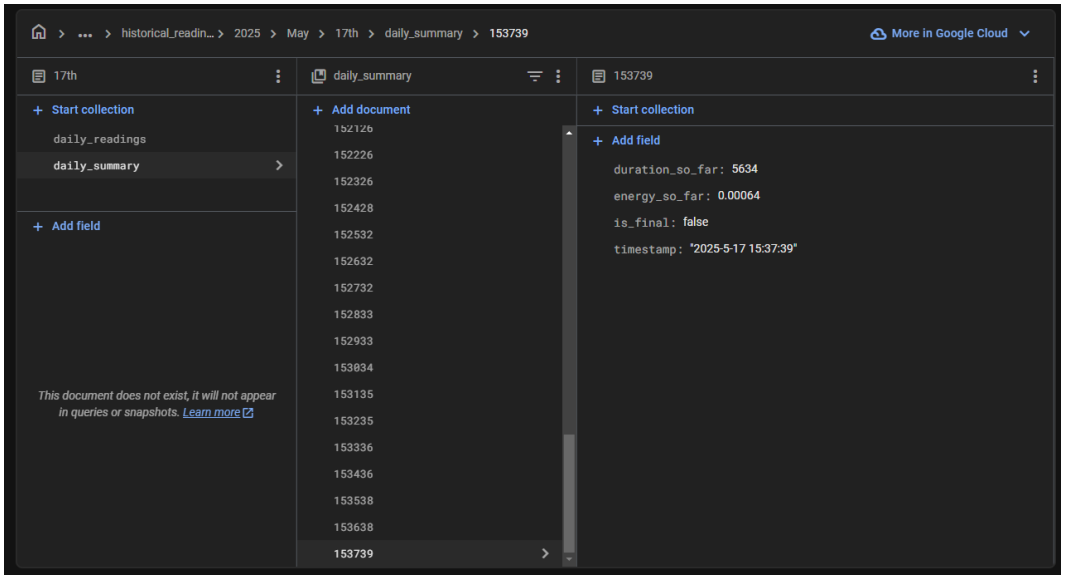
di mana pengguna dapat menekan *box* berisikan status keamanan perangkat untuk menampilkan sebuah *pop up window*. Terakhir dibagian bawah terdapat *button* untuk memilih tanggal yang ingin ditampilkan.

Gambar 5.4 *Pop Up Window* Masalah

Gambar 5.4 merupakan bentuk dari *pop up window* yang timbul ketika pengguna menekan *box* berisikan status keamanan perangkat SES. Di tampilan ini pengguna dapat melihat masalah dan status keamanan perangkat. Adapun *Button* bertuliskan "*Click to Turn Off!*" berfungsi untuk menonaktifkan perangkat SES. Dengan nonaktifnya perangkat SES, perangkat SES akan tetap dalam kondisi mati meskipun pengguna telah mengubah status perangkat di halaman *Device*. Pengguna harus mengaktifkan kembali perangkat SES dan mengubah status perangkat di halaman *Device* menjadi "*Online*" baru perangkat SES dapat menyala. Ada juga *button* bertuliskan "*Switch Device*" yang berfungsi untuk mengganti detail perangkat yang ditampilkan.

5.4 Implementasi Basis Data

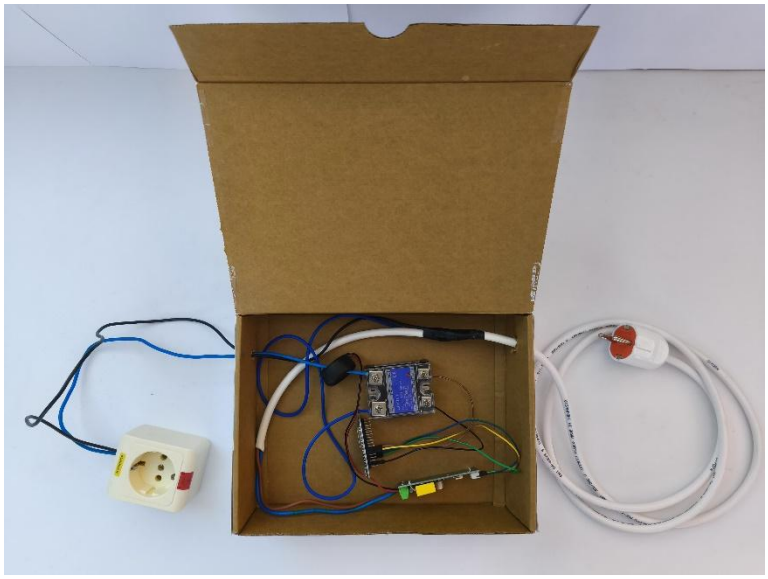
Berikut ini merupakan implementasi dari basis data yang digunakan selama pengembangan Tugas Akhir.



Gambar 5.5 Implementasi Basis Data

5.5 Implementasi Perangkat Keras

Berikut ini merupakan implementasi perangkat keras yang dirancang. Setiap perangkat SES terdiri dari 1 buah steker, 1 buah sensor PZEM-004T, 1 buah mikrokontroler ESP32 Wifi/BT, 1 buah *Solid State Relay* 10A, dan 1 buah stop kontak.



Gambar 5.6 Implementasi Perangkat Keras

5.6 Implementasi Modul Program

Terdapat 4 modul utama, yaitu menampilkan data konsumsi secara *realtime*, menampilkan data konsumsi historis pengguna berdasarkan periode waktu yang

dipilih, menyalakan dan mematikan perangkat via aplikasi, dan prediksi total konsumsi berdasarkan data penggunaan energi listrik sementara. Kode program dapat dilihat pada Lampiran A.

BAB VI
PENGUJIAN

Bagian ini berisikan proses dan hasil dari pengujian yang dilakukan, baik untuk perangkat SES maupun aplikasi yang dikembangkan. Bab ini membahas tentang tujuan pengujian, kriteria pengujian, proses pengujian yang dilakukan, yaitu *Black Box* dan *White Box Testing*, pengujian modul serta analisis hasil pengujian.

6.1 Tujuan Pengujian

Tujuan dari dilakukannya pengujian, yaitu untuk memastikan sistem yang dirancang dapat berfungsi sesuai dengan spesifikasi dan kebutuhan yang telah ditetapkan sebelumnya. Dengan menggunakan *Black Box* dan *White Box Testing*, akan dicari kesalahan atau cacat sedini mungkin sehingga dapat diperbaiki sebelum produk digunakan oleh pengguna akhir. Pengujian juga dilakukan untuk menilai kualitas dan performa sesuai dengan standar yang berlaku.

6.2 Kriteria Pengujian

Kriteria pengujian merupakan nilai-nilai dasar yang harus dipenuhi oleh sistem yang dirancang. Terdapat beberapa kriteria pengujian yang telah ditetapkan terkait hasil akhir Tugas Akhir ini, yaitu sebagai berikut:

1. Perangkat SES dapat berfungsi dengan baik.
2. Aplikasi yang dirancang dapat berjalan dengan baik.
3. Perangkat SES dan aplikasi dapat terintegrasi dengan Firebase.
4. Perangkat SES dapat membaca arus tegangan listrik dan mengirimkan data ke Firebase.
5. Aplikasi yang dirancang dapat mengakses data penggunaan arus listrik dari Firebase, dan ditampilkan di halaman antarmuka aplikasi.
6. Aplikasi dapat memperbarui status perangkat SES di Firebase.
7. Perangkat SES dapat menyala dan mati sesuai dengan status perangkat di Firebase.

8. Aplikasi dapat memberikan prediksi total konsumsi energi listrik pengguna untuk 1 bulan berdasarkan data konsumsi sementara pengguna menggunakan *Random Forest*.
9. Aplikasi dapat meberikan informasi apabila terjadi kejanggalan pada perangkat SES.

6.3 *Black Box Testing*

Pada bagian ini akan dilakukan pengujian menggunakan *Black Box Testing* guna menguji alur dan fungsionalitas sistem yang dirancang baik itu perangkat SES maupun operasi aplikasi Android. Pengujian *Black Box Testing* akan dilakukan oleh salah satu kolega penulis.

Pengujian dilakukan menggunakan perangkat *Smartphone* Android Oppo Reno 5 4G dengan spesifikasi sebagai berikut.

Tabel 6.1 Spesifikasi Perangkat Pengujian

NO	Spesifikasi	Spesifikasi Perangkat Pengujian
1	*Operating System*	Android 13 *Color* OS
2	*Processor*	Qualcomm Snapdragon 720G *Octa-core*
3	RAM	8Gb *Physical* RAM dan *up to* 6Gb *Virtual* RAM (*Extended* RAM)
4	ROM	128Gb *Internal Storage*
5	Jaringan	Telkomsel 4G

Berikut ini merupakan hasil dari pengujian yang dilakukan.

Tabel 6.2 Hasil Pengujian *Black Box Testing*

NO	Aktivitas Pengujian	Hasil yang diharapkan	Hasil pengujian	Kesimpulan
1.	Menjalankan dan operasi navigasi aplikasi Android	Aplikasi dapat diluncurkan dan navigasi antar halaman dapat beroperasi dengan baik.	Aplikasi berhasil diluncurkan dan navigasi antar halaman juga beroperasi dengan baik	[x] Diterima [] Ditolak

NO	Aktivitas Pengujian	Hasil yang diharapkan	Hasil pengujian	Kesimpulan
2.	Menampilkan data penggunaan voltase secara *realtime* menggunakan grafik di aplikasi	Aplikasi menampilkan data penggunaan voltase secara *realtime* menggunakan grafik	Aplikasi berhasil menampilkan data penggunaan voltase secara *realtime* menggunakan grafik	[x] Diterima [] Ditolak
3.	Menampilkan hasil prediksi total konsumsi energi untuk bulan berjalan berdasarkan data konsumsi energi sementara pengguna	Aplikasi menampilkan hasil prediksi total konsumsi energi untuk bulan berjalan	Aplikasi berhasil menampilkan hasil prediksi total konsumsi energi listrik untuk bulan berjalan	[x] Diterima [] Ditolak
4.	Menampilkan data bacaan tegangan listrik berupa *Voltage* (V), *Current* (A), *Power* (kW), dan *Energy* (kWh) untuk masing-masing perangkat SES	Aplikasi menampilkan data tegangan listrik untuk masing-masing perangkat SES	Aplikasi berhasil menampilkan data tegangan listrik untuk masing-masing perangkat SES	[x] Diterima [] Ditolak

NO	Aktivitas Pengujian	Hasil yang diharapkan	Hasil pengujian	Kesimpulan
5.	Memperbarui status perangkat SES dengan menekan *button* "*Toggle Device Status*"	Status perangkat SES diperbarui sesuai dengan pilihan pengguna	Status perangkat SES berhasil diperbarui sesuai dengan pilihan pengguna	[x] Diterima [] Ditolak
6.	Menampilkan data historis konsumsi energi listrik pengguna seusai dengan periode waktu yang dipilih	Aplikasi menampilkan data historis konsumsi energi listrik pengguna berdasarkan periode waktu yang dipilih	Aplikasi berhasil menampilkan data historis konsumsi energi listrik pengguna berdasarkan periode waktu yang dipilih	[x] Diterima [] Ditolak

6.4 *White Box Testing*

Berikut ini adalah hasil dari pengujian yang dilakukan menggunakan *White Box Testing* yang dilakukan oleh penulis.

Tabel 6.3 Hasil Pengujian *White Box Testing*

NO	Kasus Pengujian	Input	Hasil Pengujian
1.	Membaca arus tegangan listrik yang digunakan beban.	Terdapat beban yang terhubung.	Menampilkan data konsumsi *realtime*.
		Tidak ada beban yang terhubung.	Menampilkan data konsumsi terakhir.
2.	Prediksi total konsumsi energi pengguna selama 1 bulan.	Data yang dimiliki mencukupi standar penggunaan *Random*	Melakukan prediksi total konsumsi energi listrik

NO	Kasus Pengujian	Input	Hasil Pengujian
		Forest (Data historis 3 bulan terakhir & data konsumsi harian > 5 hari).	pengguna berdasarkan data konsumsi harian menggunakan *Random Forest*.
		Data yang dimiliki tidak mencukupi standar penggunaan *Random Forest* (Data historis 3 bulan terakhir & data konsumsi harian > 5 hari).	Melakukan prediksi total konsumsi energi listrik penguna berdasarkan data konsumsi harian menggunakan *Linear Regression*.

6.5 Pengujian Fungsi Sistem

Berikut merupakan beberapa kasus pengujian yang dilakukan terkait aplikasi dan perangkat yang dibangun.

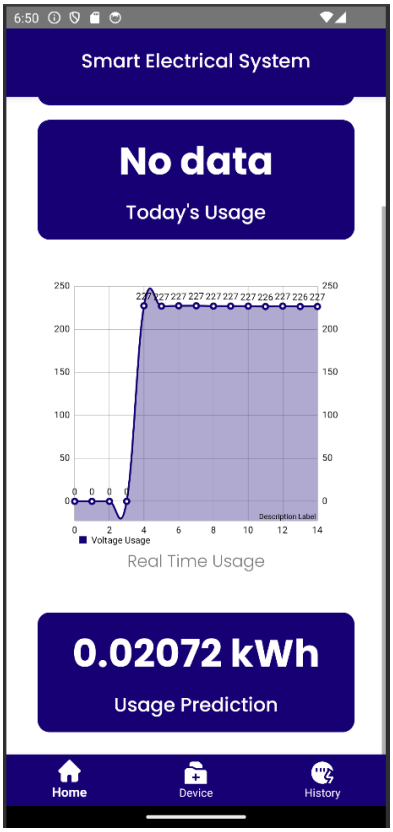
1. Pengujian Prediksi Menggunakan Random Forest

Berikut ini merupakan hasil pengujian untuk prediksi dengan *Random Forest*. Proses prediksi dapat dilihat di BAB III pada halaman 12. Simulasi ini akan dilakukan dengan ketentuan sebagai berikut.

Tabel 6.4 Ketentuan Simulasi Prediksi

NO	Ketentuan simulasi	Nilai yang digunakan
1.	Jumlah hari dengan data	5 Hari (1 Juni – 5 Juni)
2.	Jumlah hari yang diprediksi	25 Hari (6 Juni – 30 Juni)
3.	Jumlah pohon yang digunakan	20 Pohon
4.	Nilai persentase data *Bootstrap*	70%
5.	Tanggal pengujian	6 Juni 2025

Berikut ini merupakan tampilan antarmuka aplikasi yang menampilkan nilai prediksi untuk bulan Juni.



Gambar 6.1 Tampilan Antarmuka Prediksi

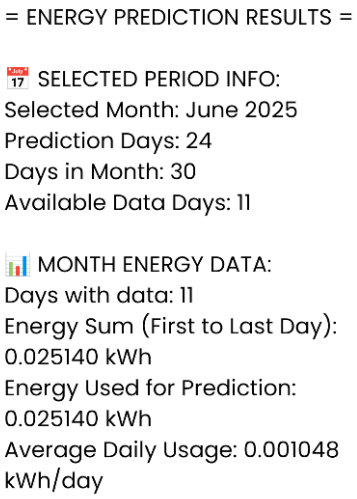
2. Pengujian Prediksi Dengan *Missing Data*

Berikut ini merupakan hasil pengujian prediksi menggunakan *Random Forest* dengan missing data. Simulasi ini akan dilakukan dengan ketentuan sebagai berikut.

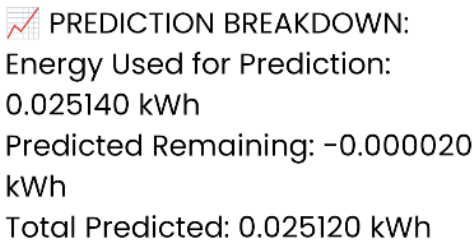
Tabel 6.5 Ketentuan Simulai Prediksi

NO	Ketentuan simulasi	Nilai yang digunakan
1.	Jumlah hari dengan data	11 Hari (1 Juni – 11 Juni)
2.	Jumlah hari yang diprediksi	6 Hari (25 Juni – 30 Juni)
3.	Jumlah hari dengan data kosong	13 Hari (12 Juni – 24 Juni)
4.	Tanggal pengujian	24 Juni 2025

Berikut ini merupakan gambar dari tampilan antarmuka aplikasi yang menampilkan nilai prediksi untuk bulan Juni dengan *missing data*.



Gambar 6.2 Tampilan Antarmuka Ketentuan Prediksi



Gambar 6.3 Tampilan Antarmuka Hasil Prediksi

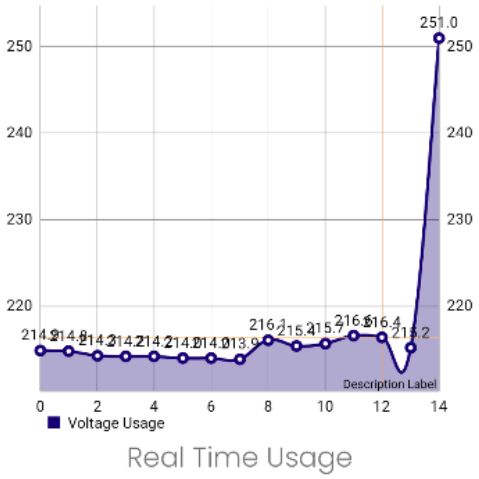
3. Pengujian Pemutusan Aliran Listrik Otomatis

Berikut ini merupakan hasil pengujian pemutusan aliran listrik otomatis ketika nilai voltase aliran listrik melebihi batas umum yaitu 220-230V.

Tabel 6.6 Ketentuan Simulasi Anomali

NO	Ketentuan simulasi	Nilai yang digunakan
1.	Nilai voltase yang digunakan	251 Volt
2.	Batas toleransi nilai voltase	>= 230 Volt

Berikut ini merupakan gambar dari tampilan antarmuka aplikasi yang menampilkan nilai voltase yang dibaca dan status keamanan perangkat SES.



Gambar 6.4 Tampilan Grafik Voltase *Realtime*

Gambar 6.5 Tampilkan Status Keamanan Perangkat

6.6 Analisis Hasil Pengujian

Berdasarkan hasil pengujian yang dilakukan, baik itu menggunakan *Black Box Testing* maupun *White Box Testing* dapat dilihat bahwa sistem yang dirancang dapat berjalan dengan baik dan dapat memenuhi persyaratan awal yang telah ditentukan.

BAB VII
KESIMPULAN DAN SARAN

7.1 Kesimpulan

Berdasarkan hasil perancangan, pengembangan, dan juga pengujian yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut:

1. *Smart Electrical System* dapat beroperasi dengan baik dan mampu untuk menampilkan data penggunaan secara *realtime*.
2. *Smart Electrical System* dapat menggunakan algoritma *Random Forest* untuk memprediksi total konsumsi energi listrik pengguna selama 1 bulan berdasarkan konsumsi harian pengguna.
3. *Smart Electrical System* dapat membantu pengguna mengontrol konsumsi energi listrik mereka dan mengurangi risiko terjadinya pemborosan listrik dan memberikan infromasi serta menonaktifkan perangkat SES apabila terdeteksi anomali pada perangkat SES.

7.2 Saran

Berikut ini merupakan saran untuk pengembangan *Smart Electrical System* untuk ke depannya agar dapat beroperasi lebih optimal dan dapat bermanfaat bagi masyarakat luas.

1. Aplikasi dapat dibuat dengan dinamis agar pengguna dapat dengan bebas mengatur jumlah perangkat SES yang ingin digunakan.
2. Mempertimbangkan faktor eksternal seperti suhu, cuaca, dan lain sebagainya sebagai fitur prediksi.
3. Rangkaian perangkat SES dapat dibuat lebih ringkas agar dapat digunakan dengan lebih praktis.
4. Dapat menerapkan algoritma *Intrusion Detection System* (IDS) untuk menganalisis lebih detail mengenai permasalahan pada aliran listrik.

DAFTAR PUSTAKA

[1] A. Prionggo dan V. Puspawardani, AYO HEMAT LISTRIK DAN DUKUNG ENERGI TERBARUKAN, Jakarta: Koaksi Indonesia, 2018.

[2] A. P. O. Amane, *Internet of Things* di berbagai bidang, Jambi: SONPEDIA, 2023.

[3] R. H. Nugraha, E. Yuwono, L. Prasetyohadi, Y. A. B dan H. Patria, "Analisis Konsumsi Energi Listrik Pelanggan dan Biaya Pokok Produksi Penyediaan Energi Listrik dengan *Machine Learning*," *Jurnal Sains Komputer & Informatika (J-SAKTI),* vol. 6, no. 1, pp. 47-56, 2022.

[4] Ardianto, A. B. Raharjo dan D. Purwitasari, "*Random Forest Regression* Untuk Prediksi Produksi Daya Pembangkit Listrik Tenaga Surya," *Briliant Jurnal Riset dan Konseptual,* vol. 7, no. 4, pp. 1058-1075, 2022.

[5] A. W. Aditya, N. R. Alham, R. M. Utomo dan Hilmansyah, "Sistem Pemantauan Konsumsi Energi Listrik Berbasis Web Sebagai Upaya Konservasi Energi," *Techno.Com,* vol. 22, no. 1, pp. 36-44, 2023.

[6] Z. M. P. Rumpesak, "Sistem Pemantauan dan Prediksi Pemakaian Daya Listrik Berbasis *Internet of Things* Menggunakan *Algoritma K-Nearest Neighbor*," Manado, 2022.

[7] S. A. Havid, "*Intrusion Detection System Internet of Things* Pada Sistem Pemantauan Pemakaian Energi Listrik Berbasis *Machine Learning*," Universitas Pendidikan Indonesia, Bandung, 2024.

[8] E. P. Febtiawan, L. A. S. I. Akbar dan A. S. Rachman, "*Forecasting* Produksi Energi *Photovoltaic* Menggunakan Algoritma *Random Forest Classification*," *Journal of Information System Research (JOSH),* vol. 5, no. 4, pp. 1053-1062, 2024.

[9] G. A. M. Ashfania, T. Prahasto, A. Widodo dan T. Warsokusumo, "Penggunaan Algoritma *Random Forest* untuk Klasifikasi berbasis Kinerja Efisiensi Energi pada Sistem Pembangkit Daya," *ROTASI,* vol. 24, no. 3, pp. 14-21, 2022.

[10] H. Bevrani, M. Wanatabe dan Y. Mitani, *Power System Monitoring and Control,* Japan: WILEY, 2014.

[11] S. Borlase, *Smart Grids: Advanced Technologies and Solutions*, New York: CRC Press, 2017.

[12] S. Piu, A. Arifin dan M. Rizal, "Optimasi Penggunaan Energi Listrik Bagi Pelanggan Rumah Tangga Berbasis *Machine Learning* dan *Internet of Things*," *MALCOM: Indonesian Journal of Machine Learning and Computer Science,* vol. 5, no. 1, pp. 84-92, 2025.

[13] Ridwan, S. B. Mulia dan A. I. Rosid, "Sistem Pemantauan Penggunaan Listrik Rumah Tangga Dengan *Website* Berbasis IoT," *Journal of Energy and Electrical Engineering (JEEE),* vol. 4, no. 2, pp. 125-131, 2023.

[14] C. Cekdin, Distribusi Daya Listrik: Teori dan Praktik, Yogyakarta: Penerbit ANDI, 2021.

[15] N. F. Hidayat, W. Sya'roni dan U. Habibah, *Random Forest*: Teori, Metode, dan Studi Kasus, Probolinggo: Pustaka Nurja, 2024.

[16] R. Novrian, T. Agustiani, M. Fikri dan M. F. Hikmatulloh, "Penerapan Algoritma *Random Forest* dalam Prediksi Status Penerima PIP pada Siswa: Studi Kasus pada SMK Amaliah 1," *Karimah Tauhid,* vol. 3, no. 2, pp. 1791-1799, 2024.

[17] I. P. Astuti dan G. A. Buntoro, Pemrograman Android Dasar, Ponorogo: UNMUH Ponorogo Press, 2023.

[18] A. Syahfitri, "*Internet of Things*, Sejarah, Teknologi, dan Penerapannya," *Uranus: Jurnal Ilmiah Teknik Elektro, Sains dan Informatika,* vol. 3, no. 1, pp. 113-120, 2025.

[19] K. P. Seng, L. M. Ang dan E. Ngharamike, "*Artificial intelligence Internet of Things: A new paradigm of distributed sensor networks*," *International Journal of Distributed Sensor Networks,* vol. 18, no. 3, pp. 1-27, 2022.

[20] S. Suryono dan Hardiansah, Panduan Praktis Membuat Aplikasi Android dengan Android Studio, Depok: PT Lauwba Techno Indonesia, 2020.

[21] J. J. Sanjaya dan J. Susilo, "Perbandingan Performa Kotlin vs Java dalam Pengembangan Android dengan Metode Iterasi While," *Bit-Tech (Binary Digital - Technology),* vol. 7, no. 2, pp. 545-553, 2024.

[22] Purnomo, F. Rosyana, O. Purbo dan R. Aziz, Firebase: Membangun Aplikasi Berbasis Android, Yogyakarta: Percetakan CV, Andi Offset, 2021.

[23] P. Dutson, *Android Development Patterns: Best Practices for Professional Developers*, Boston: Addison-Wesley Professional, 2016.

[24] ESPRESSIF SYSTEM, "Hardware: ESP32," [Online]. Available: https://www.espressif.com/en/products/socs/esp32#:~:text=ESP32%20is%20highly%2Dintegrated%20with,Circuit%20Board%20(PCB)%20requirements.. [Diakses 16 March 2025].

[25] V. Barral, O. Campos, T. Dominguez-Bolano, C. J. Escudero dan J. A. Garcia-Naya, "*Fine Time Measurement for the Internet of Things: A Practical Aproach Using ESP32*," *IEEE Internet of Things Journal,* vol. 9, no. 19, pp. 1-14, 2022.

[26] NN DIGITAL, "NN DIGITAL: Mengenal PZEM-004T Modul Elektronik Untuk Alat Pengukur Listrik," 10 July 2019. [Online]. Available: https://www.nn-digital.com/blog/2019/07/10/mengenal-pzem-004t-modul-elektronik-untuk-alat-pengukuran-listrik/. [Diakses 16 March 2025].

[27] S. Anwar, T. Artono, Nasrul, Dasrul dan A. Fadli, "Pengukuran Energi Listrik Berbasis PZEM-004T," *Proceeding Seminar Nasional Politeknik Negeri Lhokseumawe,* vol. 3, no. 1, pp. 272-276, 2019.

[28] E. A. Prastyo, "EdukasiElektronika: Edukasi: Komponen: Pengertian dan Penjelasan tentang *Solid State Relay* (SSR)," 2023. [Online]. Available:

https://www.edukasielektronika.com/2023/05/pengertian-dan-penjelasan-tentang-solid-state-relay.html. [Diakses 16 March 2025].

[29] Suryono, Supriyati dan S. Kusumastuti, "Rancang Bangun Sensor Gesture Sebagai Pengganti Saklar Pengontrol Lampu Tanpa Sentuhan," *ORBITH*, vol. 18, no. 1, pp. 53-63, 2022.

[30] E. Pawan, Thamrin, Rosiyati, P. Hasan, S. Bei dan P. Matu, "*Using Waterfall Method to Design Information System of* SPMI STIMIK Sepuluh Nopember Jayapura," *International Journal of Computer and Information System (UCIS)*, vol. 2, no. 2, pp. 34-39, 2021.

[31] Y. E. Achyani dan S. Saumi, "Penerapan Metode *Waterfall* Pada Sistem Informasi Manajemen Buku Perpustakaan Berbasis WEB," *STMIK Palangka Raya*, pp. 83-94, 2017.

[32] E. Endaryati, Sistem Informasi Akuntansi, Semarang: Yayasan Prima Agus Teknik, 2021.

[33] H. Podeswa, *UML for the IT Business Analyst, Second Edition: A Practical Guide to Requirements Gathering Using the Unified Modeling Language*, Boston: Course Technology, 2015.

[34] D. Rosca dan L. Domingues, "*A Systematic Comparison of Roundtrip Software Engineering Approaches applied to UML Class Diagram*," *Elsevier Science Direct*, vol. 181, pp. 861-868, 2021.

[35] D. A dan W. B, *System Analysis and Design: An Object-Oriented Approach with UML*, Virginia: John Wiley & Sons, 2015.

[36] T. D. Capote, "*A Comparative Study Of Black Box And White Box Testing Techniques In Modern Software Development*," *Frontiers In Engineering and Technology (FET)*, vol. 5, no. 1, pp. 1-7, 2024.

[37] T. H. Cormen, C. E. Leiserson, R. L. Rivest dan C. Stein, *Intoduction to Algorithms*, 4rd ed, Cambridge: MIT, 2022.

LAMPIRAN A

KODE PROGRAM

1. Menampilkan Data Konsumsi *Realtime*

Berikut ini merupakan sebagian kode yang digunakan untuk menampilkan data penggunaan *voltage* pengguna.

```
private fun fetchLast15ReadingsAndUpdateChart() {
    val readingsRef = db.collection("historical_data")
        .document("SES01")
        .collection("readings")
        .orderBy("timestamp", Query.Direction.DESCENDING)
        .limit(15)

    readingsRef.get().addOnSuccessListener { result ->
        val documents = result.documents

        entries.clear()
        for (i in documents.indices.reversed()) {
            val document = documents[i]
            val energy = document.getDouble("voltage") ?:
0.0
            val xPosition = (documents.size - 1 -
i).toFloat()
            entries.add(Entry(xPosition, energy.toFloat()))
        }

        if (::dataSet.isInitialized) {
            dataSet.values = entries
            dataSet.notifyDataSetChanged()
        } else {
            dataSet = LineDataSet(entries, "Voltage
Usage").apply {
                color =
ContextCompat.getColor(requireContext(), R.color.navy)

setCircleColor(ContextCompat.getColor(requireContext(),
R.color.navy))
                lineWidth = 2f
                circleRadius = 4f
                setDrawCircleHole(true)
                valueTextSize = 10f
                setDrawFilled(true)
                fillColor =
ContextCompat.getColor(requireContext(), R.color.navy)
                mode = LineDataSet.Mode.CUBIC_BEZIER
            }
        }

        val lineData = LineData(dataSet)
        lineChart.data = lineData
```

```
        lineChart.xAxis.position =
XAxis.XAxisPosition.BOTTOM
        lineChart.xAxis.isGranularityEnabled = true
        lineChart.xAxis.granularity = 1f
        lineChart.xAxis.axisMinimum = 0f
        lineChart.xAxis.axisMaximum = (entries.size -
1).toFloat()

        lineChart.invalidate()
        lineChart.animateX(1000)
        lineChart.animateY(1000)
    }.addOnFailureListener { exception ->
        exception.printStackTrace()
    }
}
```

2. Menampilkan Data Konsumsi Historis

Berikut ini merupakan sebagian kode yang digunakan untuk menampilkan data historis konsumsi energi pengguna berdasarkan periode waktu yang dipilih.

```
private fun fetchDataDaily(onResult: (Map<String, Double>) -
> Unit) {
    val year = calendar.get(Calendar.YEAR).toString()
    val month = calendar.getDisplayName(Calendar.MONTH,
Calendar.LONG, Locale.getDefault()) ?: return
    val dayWithSuffix = getDayWithSuffix()

    val dailyReadingsRef = db.collection("historical_data")
        .document("SES01")
        .collection("historical_readings")
        .document(year)
        .collection(month)
        .document(dayWithSuffix)
        .collection("daily_readings")

    dailyReadingsRef.get()
        .addOnSuccessListener { snapshot ->
            Log.d("DailyReadings", "Documents count:
${snapshot.size()}")
            if (snapshot.isEmpty) {
                Log.d("DailyReadings", "No documents found")
                onResult(emptyMap())
                return@addOnSuccessListener
```

```
            }

            val timeFormatter =
DateTimeFormatter.ofPattern("H:m:s")

            val latestDoc = snapshot.documents.maxByOrNull {
doc ->
                val readings = doc.get("readings") as?
List<Map<String, Any>>
                readings?.mapNotNull {
it["timestamp"]?.toString()?.substringAfter(" ") }
                    ?.mapNotNull { runCatching {
LocalTime.parse(it, timeFormatter) }.getOrNull() }
                    ?.maxOrNull() ?: LocalTime.MIN
            }

            if (latestDoc != null) {
                val readings = latestDoc.get("readings") as?
List<Map<String, Any>>
                Log.d("DailyReadings", "Using document:
${latestDoc.id}, readings size: ${readings?.size ?: 0}")

                if (!readings.isNullOrEmpty()) {
                    val resultMap = readings.takeLast(15)
                        .asReversed() // ← Flip the list
                        .mapNotNull { reading ->
                            val timestamp =
reading["timestamp"]?.toString()
                            val voltage =
(reading["voltage"] as? Number)?.toDouble()
                            if (timestamp != null && voltage
!= null) timestamp to voltage else null
                        }.toMap()


                    Log.d("DailyReadings", "Mapped result
size: ${resultMap.size}")
                    onResult(resultMap)
                } else {
```

```
                    Log.d("DailyReadings", "No readings
array found in latest document")
                    onResult(emptyMap())
                }
            } else {
                Log.d("DailyReadings", "No valid documents
with timestamps")
                onResult(emptyMap())
            }
        }
        .addOnFailureListener { e ->
            Log.e("DailyReadings", "Error fetching data", e)
            onResult(emptyMap())
        }
}

private fun fetchDataMonth(
    onResult: (Map<String, Double>) -> Unit
) {
    val db = FirebaseFirestore.getInstance()
    val year = calendar.get(Calendar.YEAR).toString()
    val month = calendar.getDisplayName(Calendar.MONTH,
Calendar.LONG, Locale.getDefault()) ?: return
    val baseRef = db.collection("historical_data")
        .document("SES01")
        .collection("historical_readings")
        .document(year)
        .collection(month)

    baseRef.get().addOnSuccessListener { dayDocs ->
        val dayTasks = dayDocs.documents.map { dayDoc ->
            val dayName = dayDoc.id

            baseRef.document(dayName)
                .collection("daily_summary")
                .get()
                .continueWith { task ->
                    val snapshot = task.result
```

```
                    val finalDoc =
snapshot?.documents?.firstOrNull {
                        it.getBoolean("is_final") == true
                    } ?: snapshot?.documents?.maxByOrNull {
doc -> doc.id.toIntOrNull() ?: Int.MIN_VALUE }

                    val energy =
(finalDoc?.get("energy_so_far") as? Number)?.toDouble() ?:
0.0

                    dayName to energy
                }
            }

            Tasks.whenAllSuccess<Pair<String, Double>>(dayTasks)
                .addOnSuccessListener { results ->
                    val resultMap = results.toMap()

                    resultMap.toList()
                        .sortedBy { (day, _) -> day.filter {
it.isDigit() }.toIntOrNull() ?: Int.MAX_VALUE }
                        .forEach { (day, energy) ->
                            Log.d("MonthlySummary", "Day: $day →
Energy: ${"%.5f".format(energy)}")
                        }

                    onResult(resultMap)
                }
                .addOnFailureListener { e ->
                    Log.e("MonthlySummary", "Failed to load
summaries", e)
                    onResult(emptyMap())
                }

        }.addOnFailureListener { e ->
            Log.e("MonthlySummary", "Failed to load day
documents", e)
            onResult(emptyMap())
        }
```

```
}

private fun fetchDataWeek(
    onResult: (Map<String, Double>) -> Unit
) {
    val year = calendar.get(Calendar.YEAR).toString()
    val month = calendar.getDisplayName(Calendar.MONTH,
Calendar.LONG, Locale.getDefault()) ?: return
    val baseRef = db.collection("historical_data")
        .document("SES01")
        .collection("historical_readings")
        .document(year)
        .collection(month)

    val dateList = (6 downTo 0).map { offset ->
        Calendar.getInstance().apply {
            time = calendar.time  // Use selected date
            add(Calendar.DAY_OF_MONTH, -offset)
            set(Calendar.HOUR_OF_DAY, 0)
            set(Calendar.MINUTE, 0)
            set(Calendar.SECOND, 0)
            set(Calendar.MILLISECOND, 0)
        }
    }

    val formattedDays = dateList.map {
        val day = it.get(Calendar.DAY_OF_MONTH)
        "$day${getDaySuffix(day)}"
    }

    val dayTasks = formattedDays.map { dayName ->
        baseRef.document(dayName)
            .collection("daily_summary")
            .get()
            .continueWith { task ->
                val snapshot = task.result

                val finalDoc =
snapshot?.documents?.firstOrNull {
```

```
                    it.getBoolean("is_final") == true
                } ?: snapshot?.documents?.maxByOrNull { doc
-> doc.id.toIntOrNull() ?: Int.MIN_VALUE }

                val energy = (finalDoc?.get("energy_so_far")
as? Number)?.toDouble() ?: 0.0
                dayName to energy
            }
    }

    Tasks.whenAllSuccess<Pair<String, Double>>(dayTasks)
        .addOnSuccessListener { results ->
            val resultMap = results.toMap()

            resultMap.toList()
                .sortedBy { (day, _) -> day.filter {
it.isDigit() }.toIntOrNull() ?: Int.MAX_VALUE }
                .forEach { (day, energy) ->
                    Log.d("WeeklySummary", "Day: $day →
Energy: ${"%.5f".format(energy)}")
                }

            onResult(resultMap)
        }
        .addOnFailureListener { e ->
            Log.e("WeeklySummary", "Failed to load weekly
summaries", e)
            onResult(emptyMap())
        }
}
```

3. Menyalakan dan Mematikan Perangkat

Berikut ini merupakan sebagian kode yang digunakan untuk memberikan perintah untuk mematikan atau menyalakan perangkat.

```
private fun toggleDeviceStatusSES01() {
    isDeviceOnline = !isDeviceOnline
    val newStatus = if (isDeviceOnline) "online" else
"offline"
```

```
database.child("status").setValue(newStatus).addOnSuccessLis
tener {
        updateDeviceStatusUI()
        Toast.makeText(activity, "Device status updated to
$newStatus", Toast.LENGTH_SHORT).show()
    }.addOnFailureListener {
        Toast.makeText(activity, "Failed to update device
status", Toast.LENGTH_SHORT).show()
    }
}

private fun toggleDeviceStatusSES02() {
    isDeviceOnline2 = !isDeviceOnline2
    val newStatus = if (isDeviceOnline2) "online" else
"offline"


database2.child("status").setValue(newStatus).addOnSuccessLi
stener {
        updateDeviceStatusUI()
        Toast.makeText(activity, "Device status updated to
$newStatus", Toast.LENGTH_SHORT).show()
    }.addOnFailureListener {
        Toast.makeText(activity, "Failed to update device
status", Toast.LENGTH_SHORT).show()
    }
}

private fun updateDeviceStatusUI() {
    val statusText = if (isDeviceOnline) "Online" else
"Offline"
    ses01_deviceStatus.text = "$statusText"

    val statusText2 = if (isDeviceOnline2) "Online" else
"Offline"
    ses02_deviceStatus.text = "$statusText2"
}
```

4. Prediksi Total Konsumsi Berdasarkan Data Penggunaan Energi Listrik Pengguna

Berikut ini merupakan sebagian kode yang digunakan untuk memprediksi total konsumsi energi listrik pengguna berdasarkan data konsumsi harian pengguna.

```
class EnergyPredictor {

    private val db = FirebaseFirestore.getInstance()
    private var predictionValue: Double = 0.0
    private var isUpdated = false
    private var lastPredictionUpdate: Long = 0
    private val UPDATE_INTERVAL = 3600000
    private val randomForest = RandomForest()
```

```
    fun predictEndOfMonthUsage(onPredictionComplete: (Double) ->
Unit) {
        val currentTime = System.currentTimeMillis()
        if (isUpdated && currentTime - lastPredictionUpdate <
UPDATE_INTERVAL) {
            onPredictionComplete(predictionValue)
            return
        }

        val calendar = Calendar.getInstance()
        val currentYear = calendar.get(Calendar.YEAR).toString()
        val currentMonth = calendar.getDisplayName(Calendar.MONTH,
Calendar.LONG, Locale.getDefault()) ?: return
        val currentDay = calendar.get(Calendar.DAY_OF_MONTH)
        val daysInMonth =
calendar.getActualMaximum(Calendar.DAY_OF_MONTH)

        collectHistoricalData { historicalData ->
            collectCurrentMonthData(currentYear, currentMonth) {
currentMonthData ->
                if (currentMonthData.isEmpty()) {
                    onPredictionComplete(0.0)
                    return@collectCurrentMonthData
                }

                if (historicalData.isNotEmpty()) {
                    randomForest.train(historicalData)
                }

                val features = prepareFeatures(currentMonthData,
currentDay, daysInMonth)

                val totalPredictedEnergy = if
(historicalData.isNotEmpty()) {
                    randomForest.predict(features)
                } else {
                    fallbackLinearPrediction(currentMonthData,
currentDay, daysInMonth)
                }

                updatePrediction(totalPredictedEnergy)
                onPredictionComplete(totalPredictedEnergy)
            }
        }
    }

    private fun collectHistoricalData(callback:
(List<TrainingData>) -> Unit) {
        val trainingDataList = mutableListOf<TrainingData>()
        val calendar = Calendar.getInstance()
        val currentYear = calendar.get(Calendar.YEAR)
        val currentMonth = calendar.get(Calendar.MONTH)

        var processedMonths = 0
        val monthsToProcess = 6

        for (i in 1..monthsToProcess) {
            val targetCalendar = Calendar.getInstance()
            targetCalendar.add(Calendar.MONTH, -i)
```

```
            val year =
targetCalendar.get(Calendar.YEAR).toString()
            val month =
targetCalendar.getDisplayName(Calendar.MONTH, Calendar.LONG,
Locale.getDefault()) ?: continue
            val daysInMonth =
targetCalendar.getActualMaximum(Calendar.DAY_OF_MONTH)

            collectMonthData(year, month, daysInMonth) { monthData
->
                if (monthData.isNotEmpty()) {
                    for (day in 7..daysInMonth) {
                        val currentData = monthData.filter {
it.day <= day }
                        if (currentData.size >= 7) {
                            val features =
prepareFeatures(currentData, day, daysInMonth)
                            val actualTotal =
monthData.maxOfOrNull { it.energySoFar } ?: 0.0

trainingDataList.add(TrainingData(features, actualTotal))
                        }
                    }
                }

                processedMonths++
                if (processedMonths >= monthsToProcess) {
                    callback(trainingDataList)
                }
            }
        }
    }

    private fun collectMonthData(year: String, month: String,
daysInMonth: Int, callback: (List<DailyEnergyData>) -> Unit) {
        val dailyDataList = mutableListOf<DailyEnergyData>()
        var processedDays = 0

        for (day in 1..daysInMonth) {
            val dayWithSuffix = day.toString() + getDaySuffix(day)
            db.collection("historical_data")
                .document("SES01")
                .collection("historical_readings")
                .document(year)
                .collection(month)
                .document(dayWithSuffix)
                .collection("daily_summary")
                .whereEqualTo("is_final", true)
                .limit(1)
                .get()
                .addOnSuccessListener { documents ->
                    if (!documents.isEmpty) {
                        val doc = documents.documents[0]
                        val energySoFar =
doc.getDouble("energy_so_far") ?: 0.0
                        dailyDataList.add(DailyEnergyData(day,
energySoFar))
                    }
```

```
                    processedDays++
                    if (processedDays >= daysInMonth) {
                        callback(dailyDataList.sortedBy { it.day
})
                    }
                }
                .addOnFailureListener {
                    processedDays++
                    if (processedDays >= daysInMonth) {
                        callback(dailyDataList.sortedBy { it.day
})
                    }
                }
        }
    }

    private fun collectCurrentMonthData(year: String, month:
String, callback: (List<DailyEnergyData>) -> Unit) {
        val dailyDataList = mutableListOf<DailyEnergyData>()
        val currentDay =
Calendar.getInstance().get(Calendar.DAY_OF_MONTH)

        var processedDays = 0

        for (day in 1..currentDay) {
            val dayWithSuffix = day.toString() + getDaySuffix(day)
            db.collection("historical_data")
                .document("SES01")
                .collection("historical_readings")
                .document(year)
                .collection(month)
                .document(dayWithSuffix)
                .collection("daily_summary")
                .whereEqualTo("is_final", true)
                .limit(1)
                .get()
                .addOnSuccessListener { documents ->
                    if (!documents.isEmpty) {
                        val doc = documents.documents[0]
                        val energySoFar =
doc.getDouble("energy_so_far") ?: 0.0
                        dailyDataList.add(DailyEnergyData(day,
energySoFar))
                    }

                    processedDays++
                    if (processedDays >= currentDay) {
                        callback(dailyDataList.sortedBy { it.day
})
                    }
                }
                .addOnFailureListener {
                    processedDays++
                    if (processedDays >= currentDay) {
                        callback(dailyDataList.sortedBy { it.day
})
                    }
                }
```

```
        }
    }

    private fun prepareFeatures(dailyData: List<DailyEnergyData>,
currentDay: Int, daysInMonth: Int): DoubleArray {
        val features = mutableListOf<Double>()

        val totalEnergyUsed = dailyData.sumOf { it.energySoFar }
        features.add(totalEnergyUsed)

        val avgDailyUsage = totalEnergyUsed / currentDay
        features.add(avgDailyUsage)

        val monthProgress = currentDay.toDouble() / daysInMonth
        features.add(monthProgress)

        val daysRemaining = daysInMonth - currentDay
        features.add(daysRemaining.toDouble())

        val recentData = dailyData.takeLast(minOf(3,
dailyData.size))
        val recentAvg = if (recentData.size > 1) {
            val recentDailyUsages = recentData.zipWithNext { a, b
-> b.energySoFar - a.energySoFar }
            recentDailyUsages.average()
        } else avgDailyUsage

        val trendRatio = if (avgDailyUsage > 0) recentAvg /
avgDailyUsage else 1.0
        features.add(trendRatio)

        features.add((currentDay % 7).toDouble())

        val dailyUsages = dailyData.zipWithNext { a, b ->
b.energySoFar - a.energySoFar }
        val variance = if (dailyUsages.size > 1) {
            val mean = dailyUsages.average()
            dailyUsages.map { (it - mean).pow(2) }.average()
        } else 0.0
        features.add(sqrt(variance))

        return features.toDoubleArray()
    }

    private fun fallbackLinearPrediction(currentMonthData:
List<DailyEnergyData>, currentDay: Int, daysInMonth: Int): Double
{
        val totalEnergyUsedSoFar = currentMonthData.sumOf {
it.energySoFar }
        val averageDailyUsage = totalEnergyUsedSoFar / currentDay
        val remainingDays = daysInMonth - currentDay
        val predictedRemainingEnergy = averageDailyUsage *
remainingDays
        return totalEnergyUsedSoFar + predictedRemainingEnergy
    }

    private fun updatePrediction(prediction: Double) {
        predictionValue = prediction
        isUpdated = true
```

```
        lastPredictionUpdate = System.currentTimeMillis()
    }

    private fun getDaySuffix(day: Int): String {
        return when {
            day in 11..13 -> "th"
            day % 10 == 1 -> "st"
            day % 10 == 2 -> "nd"
            day % 10 == 3 -> "rd"
            else -> "th"
        }
    }

    data class DailyEnergyData(val day: Int, val energySoFar:
Double)
    data class TrainingData(val features: DoubleArray, val target:
Double)

    class RandomForest {
        private val trees = mutableListOf<DecisionTree>()
        private val numTrees = 10
        private val maxDepth = 8
        private val minSamplesLeaf = 3

        fun train(trainingData: List<TrainingData>) {
            trees.clear()

            for (i in 0 until numTrees) {
                val bootstrapSample = (0 until
trainingData.size).map {

trainingData[Random.nextInt(trainingData.size)]
                }

                val tree = DecisionTree(maxDepth, minSamplesLeaf)
                tree.train(bootstrapSample)
                trees.add(tree)
            }
        }

        fun predict(features: DoubleArray): Double {
            if (trees.isEmpty()) return 0.0

            val predictions = trees.map { it.predict(features) }
            return predictions.average()
        }
    }

    class DecisionTree(private val maxDepth: Int, private val
minSamplesLeaf: Int) {
        private var root: TreeNode? = null

        fun train(data: List<TrainingData>) {
            root = buildTree(data, 0)
        }

        fun predict(features: DoubleArray): Double {
            return root?.predict(features) ?: 0.0
        }
```

```
        private fun buildTree(data: List<TrainingData>, depth:
Int): TreeNode {
            if (depth >= maxDepth || data.size <= minSamplesLeaf)
{
                val avgTarget = data.map { it.target }.average()
                return TreeNode.LeafNode(avgTarget)
            }

            var bestSplit: Split? = null
            var bestScore = Double.MAX_VALUE

            val numFeatures = data.firstOrNull()?.features?.size
?: 0

            val featuresToTry = (0 until
numFeatures).shuffled().take(sqrt(numFeatures.toDouble()).toInt().
coerceAtLeast(1))

            for (featureIndex in featuresToTry) {
                val values = data.map { it.features[featureIndex]
}.distinct().sorted()

                for (i in 0 until values.size - 1) {
                    val threshold = (values[i] + values[i + 1]) /
2
                    val (left, right) = data.partition {
it.features[featureIndex] <= threshold }

                    if (left.size < minSamplesLeaf || right.size <
minSamplesLeaf) continue

                    val score = calculateMSE(left) * left.size +
calculateMSE(right) * right.size

                    if (score < bestScore) {
                        bestScore = score
                        bestSplit = Split(featureIndex, threshold,
left, right)
                    }
                }
            }

            return if (bestSplit != null) {
                val leftChild = buildTree(bestSplit.left, depth +
1)
                val rightChild = buildTree(bestSplit.right, depth
+ 1)
                TreeNode.InternalNode(bestSplit.featureIndex,
bestSplit.threshold, leftChild, rightChild)
            } else {
                val avgTarget = data.map { it.target }.average()
                TreeNode.LeafNode(avgTarget)
            }
        }

        private fun calculateMSE(data: List<TrainingData>): Double
{
            if (data.isEmpty()) return 0.0
```

```
            val mean = data.map { it.target }.average()
            return data.map { (it.target - mean).pow(2)
}.average()
        }

        private data class Split(
            val featureIndex: Int,
            val threshold: Double,
            val left: List<TrainingData>,
            val right: List<TrainingData>
        )

        sealed class TreeNode {
            abstract fun predict(features: DoubleArray): Double

            data class LeafNode(val value: Double) : TreeNode() {
                override fun predict(features: DoubleArray):
Double = value
            }

            data class InternalNode(
                val featureIndex: Int,
                val threshold: Double,
                val left: TreeNode,
                val right: TreeNode
            ) : TreeNode() {
                override fun predict(features: DoubleArray):
Double {
                    return if (features[featureIndex] <=
threshold) {
                        left.predict(features)
                    } else {
                        right.predict(features)
                    }
                }
            }
        }
    }
}
```