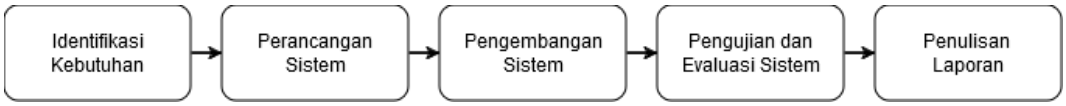
BAB III
METODE PENELITIAN

3.1 Metode Penelitian

Penelitian ini menggunakan metode *Design and Development* (D&D), yang merujuk pada model penelitian yang dikembangkan oleh Richey dan Klein (2009). Model penelitian D&D memiliki tujuan utama untuk menyediakan informasi bagi *instructional designer* (ID) bahwa suatu masalah telah ditemukan dan diselesaikan secara empiris dan sistematis melalui serangkaian penelitian yang meliputi proses desain, pengembangan, dan evaluasi (Ellis & Levy, 2010). Pendekatan ini memungkinkan pengembangan solusi berbasis teknologi yang efektif dan sesuai dengan kebutuhan.

Metode ini diterapkan untuk merancang dan mengembangkan aplikasi *mobile* berbasis *Machine Learning* guna mendeteksi *Malware* pada File APK yang diterima melalui pesan WhatsApp. Fokus dari metode ini adalah menghasilkan produk teknologi yang inovatif dan berfungsi dengan baik melalui pendekatan sistematis. Proses ini melibatkan serangkaian langkah yang mencakup desain, pengembangan, dan evaluasi secara iteratif (Solihatini dkk., 2020). Gambar 3.1 menunjukkan Proses yang dilakukan mencakup tahap-tahap metode penelitian yang digunakan.



Gambar 3.1 Tahapan pada Metode Penelitian yang digunakan

3.1.1 Identifikasi Kebutuhan

Tahap Identifikasi Kebutuhan merupakan langkah awal dalam penelitian ini yang bertujuan memahami secara mendalam permasalahan utama yang diangkat dalam latar belakang. Ancaman *Malware* yang semakin meningkat, khususnya yang menyebar melalui File APK di aplikasi WhatsApp, menjadi isu penting. File APK sering menjadi medium rentan yang digunakan untuk menyisipkan program berbahaya, yang dapat mencuri data pribadi, mengakses perangkat tanpa izin, atau

merusak sistem operasi Android. Kondisi ini menciptakan risiko keamanan yang signifikan bagi pengguna perangkat Android. Kajian literatur terkait pola penyebaran *Malware*, analisis penelitian sebelumnya tentang metode deteksi *Malware* berbasis *Machine Learning*, serta peninjauan teknologi terkini dalam mendeteksi dan mencegah ancaman menjadi langkah penting dalam merumuskan solusi. Hasil dari kajian ini mendukung pengembangan aplikasi *mobile* berbasis Android yang dirancang untuk mendeteksi *Malware* pada File APK secara efektif.

Aplikasi ini dirancang untuk memindai File APK yang diterima atau diunduh pengguna melalui WhatsApp. Deteksi dilakukan secara otomatis dengan cara membaca File APK dan menganalisisnya menggunakan model *Machine Learning*. Jika File tersebut teridentifikasi sebagai *Malware*, aplikasi akan segera memberikan notifikasi kepada pengguna sebagai bentuk peringatan.

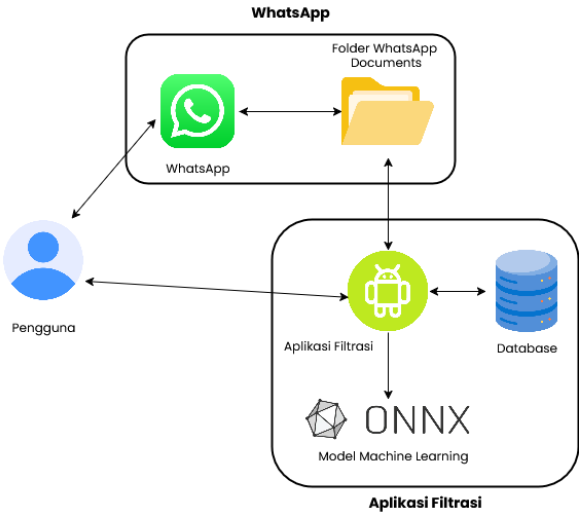
Algoritma *Random Forest* dipilih sebagai inti proses analisis karena keandalannya dalam mengelola *database* yang kompleks serta menghasilkan klasifikasi yang akurat. Proses analisis dilakukan dengan memanfaatkan gabungan prediksi dari beberapa *decision tree*, yang meningkatkan akurasi hasil deteksi. Berbagai fitur File APK, seperti izin aplikasi, struktur File, dan metadata, menjadi objek analisis untuk menentukan apakah File tersebut tergolong *Malware*.

Identifikasi kebutuhan ini memastikan bahwa aplikasi yang dirancang tidak hanya mampu mendeteksi ancaman *Malware* secara akurat, tetapi juga memberikan pengalaman pengguna yang responsif dan praktis. Dengan pendekatan ini, aplikasi diharapkan dapat menjadi solusi yang efektif untuk meningkatkan keamanan perangkat Android.

3.1.2 Perancangan Sistem

Pada tahap Perancangan Sistem, dilakukan perencanaan mendalam mengenai arsitektur aplikasi yang akan dikembangkan, dengan mempertimbangkan kebutuhan fungsional dan non-fungsional yang telah diidentifikasi pada tahap sebelumnya. Penelitian ini bertujuan untuk merancang dan mengembangkan aplikasi filtrasi konten pesan WhatsApp, khususnya untuk mendeteksi File APK yang berpotensi mengandung *Malware*. Aplikasi ini akan menggunakan model

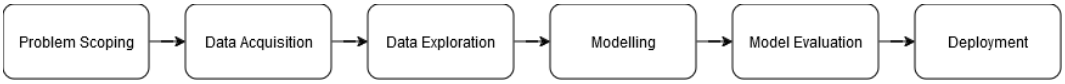
Random Forest sebagai algoritma utama untuk menganalisis dan mengklasifikasikan File APK yang diterima melalui WhatsApp. Setelah tahap pengembangan model dan aplikasi selesai, langkah berikutnya adalah implementasi model ke dalam aplikasi yang telah dikembangkan. Diagram sistem utama dari aplikasi yang dikembangkan dapat dilihat pada gambar 3.2.



Gambar 3.2 Diagram Arsitektur Sistem Aplikasi Filtrasi *Malware*

3.1.2.1 Perancangan Model Fitrasi *Malware*

Dalam perancangan model filtrasi *Malware* pada aplikasi ini, digunakan metode AI *Project Cycle* yang dikemukakan oleh Daswin De Silva dan Damminda Alahakoon (2022), yang mencakup seluruh tahapan pengembangan model kecerdasan buatan (AI), mulai dari konsepsi hingga implementasi dalam produksi. Siklus ini memungkinkan untuk merancang, mengembangkan, dan mengoptimalkan model deteksi *Malware* berbasis *Machine Learning* secara sistematis. Pendekatan AI *Project Cycle* terdiri dari beberapa tahapan yang terstruktur, yaitu *Problem Scoping*, *Data Acquisition*, *Data Exploration*, *Modelling*, *Model Evaluation*, dan *Deployment* (Aurelia & Prasetya, 2022). Pada gambar 3.3, diperlihatkan alur pengembangan model AI berdasarkan AI *Project Cycle* yang dirangkum oleh Aurelia.



Gambar 3.3 Metode Pengembangan AI *Project Cycle*

a. Problem Scoping

Tahap pertama adalah *Problem Scoping*, yang bertujuan mendefinisikan masalah yang ingin diselesaikan, yaitu mendeteksi *Malware* pada File APK yang diterima melalui WhatsApp. Masalah utama yang dihadapi adalah ancaman dari File APK berbahaya yang dapat merusak perangkat Android, sehingga aplikasi perlu dapat memindai File APK dan memberikan notifikasi jika terdeteksi *Malware*.

b. Data Acquisition

Tahap berikutnya dalam pengembangan model adalah *Data Acquisition*, di mana dataset yang relevan dikumpulkan untuk melatih model *Machine Learning*. CIC (*Canadian Institute for Cybersecurity*) adalah lembaga penelitian yang berfokus pada keamanan siber dan menyediakan berbagai Dataset yang digunakan dalam penelitian dan pengembangan sistem keamanan, termasuk deteksi *Malware*. Salah satu kontribusi penting dari CIC adalah penyediaan Dataset CICMalDroid 2020 dan CIC-AndMal2017, yang sangat relevan untuk deteksi *Malware* pada aplikasi Android.

Dataset CICMalDroid 2020 digunakan sebagai Dataset utama dalam proyek ini (Mahdavifar, Abdul Kadir, Fatemi, Alhadidi, & Ghorbani, 2020; Mahdavifar, Alhadidi, & Ghorbani, 2022), yang mencakup informasi tentang aplikasi Android yang diklasifikasikan berdasarkan apakah aplikasi tersebut aman atau berpotensi berbahaya. Dataset ini menyertakan berbagai fitur, seperti izin yang diminta, aktivitas jaringan, dan karakteristik lain yang relevan untuk mendeteksi *Malware*. Di sisi lain, CIC-AndMal2017 juga merupakan Dataset yang terdiri dari File APK yang telah dianalisis untuk mengidentifikasi apakah File tersebut terinfeksi *Malware*, lengkap dengan berbagai fitur yang menggambarkan perilaku aplikasi (Lashkari, Kadir, Taheri, & Ghorbani, 2018).

Selain kedua Dataset dari CIC tersebut, proyek ini juga memanfaatkan Dataset dari AndroZoo, sebuah proyek riset yang menyediakan koleksi besar File APK Android yang dikumpulkan dari berbagai sumber. Dataset AndroZoo sangat berguna karena menyediakan metadata yang lengkap, seperti hash File, informasi

package, izin, API calls, serta label keamanan dari berbagai antivirus engine. Data ini memberikan keragaman dan kompleksitas tambahan yang dapat memperkaya proses pelatihan model Machine Learning untuk mendeteksi aplikasi berbahaya (Allix, Bissyandé, Klein, & Le Traon, 2016).

Seluruh Dataset diperoleh langsung dari laman resmi seperti milik University of New Brunswick (UNB) untuk Dataset CIC, serta dari platform resmi AndroZoo. Dataset-Dataset ini kemudian digunakan untuk melatih model Machine Learning yang bertujuan untuk membedakan File APK yang aman dari yang berpotensi berbahaya, serta mengidentifikasi ciri-ciri aplikasi yang mengandung *Malware*.

Proses pembentukan Dataset dalam penelitian ini dimulai dengan tahapan pengumpulan File APK dari berbagai sumber yang telah ditentukan. Sumber tersebut mencakup Dataset CIC-AndMal2017, CICMalDroid2020, serta AndroZoo, yang masing-masing menyediakan beragam aplikasi baik yang tergolong *benign* maupun *Malware*. File-File APK dari ketiga sumber tersebut dikumpulkan dan disusun dalam struktur direktori berdasarkan klasifikasinya. Secara keseluruhan, Dataset yang berhasil dikumpulkan terdiri dari 12.731 aplikasi *Malware* dan 3.324 aplikasi *benign* yang berasal dari sumber CIC, serta 6.026 aplikasi *benign* tambahan yang diambil dari AndroZoo. Jumlah total data yang digunakan mencerminkan keberagaman dan keseimbangan data yang cukup baik untuk membangun model deteksi *Malware* yang andal.

Setelah File APK terkumpul, langkah selanjutnya adalah melakukan proses ekstraksi fitur dari masing-masing File. Ekstraksi ini bertujuan untuk memperoleh informasi statis dari struktur internal aplikasi Android seperti informasi yang terdapat dalam berkas AndroidManifest.xml, yang kemudian digunakan sebagai representasi numerik dalam pelatihan model *machine learning*. Proses ekstraksi dilakukan menggunakan dua pustaka utama dalam Python, yaitu *apkutils* dan *androguard*. Pustaka *apkutils* digunakan sebagai metode utama untuk membaca dan mengambil informasi dari File APK. Namun, apabila proses ekstraksi dengan pustaka ini gagal karena kendala kompatibilitas atau struktur File yang tidak standar, maka proses akan dialihkan secara otomatis menggunakan *androguard* sebagai metode alternatif.

Pendekatan komplementer ini bertujuan untuk memaksimalkan tingkat keberhasilan ekstraksi fitur, serta menjamin bahwa semua File APK dapat diproses tanpa kehilangan data penting. Fitur-fitur yang diekstraksi mencakup lima elemen utama dari struktur aplikasi Android, yaitu:

1) *Permissions* – hak akses yang diminta oleh aplikasi,
2) *Services* – layanan latar belakang,
3) *Activities* – antarmuka aktivitas pengguna,
4) *Receivers* – komponen penerima siaran atau notifikasi sistem, dan
5) *Providers* – penyedia data untuk komunikasi antar aplikasi.

Setiap fitur kemudian dikonversi ke dalam bentuk representasi biner: nilai 1 (*True*) diberikan jika fitur tersebut terdapat pada File APK, dan 0 (False) apabila tidak ditemukan. Label kelas (*Malware* atau *benign*) juga ditambahkan pada akhir setiap vektor fitur sebagai penanda untuk proses klasifikasi. Potongan kode Python yang digunakan dalam proses ekstraksi fitur ini ditampilkan pada Gambar 3.4.

```python
def extract_apk_features(apk_path):
    try:
        apk = APK.from_file(apk_path)
        manifest_xml = apk.get_manifest()
        root = ET.fromstring(manifest_xml)
        ns = {"android": "http://schemas.android.com/apk/res/android"}

        permissions = {elem.get(f"{{{ns['android']}}}name") for elem in root.findall("uses-permission", ns)}
        activities = {elem.get(f"{{{ns['android']}}}name") for elem in root.findall("application/activity", ns)}
        services = {elem.get(f"{{{ns['android']}}}name") for elem in root.findall("application/service", ns)}
        receivers = {elem.get(f"{{{ns['android']}}}name") for elem in root.findall("application/receiver", ns)}
        providers = {elem.get(f"{{{ns['android']}}}name") for elem in root.findall("application/provider", ns)}

        features = permissions | activities | services | receivers | providers
    except Exception:
        try:
            apk_andro = andro_apk.APK(apk_path)
            features = set(apk_andro.get_permissions()) | set(apk_andro.get_activities()) | \
                       set(apk_andro.get_services()) | set(apk_andro.get_receivers()) | \
                       set(apk_andro.get_providers())
        except Exception:
            print(f"❌ Gagal mengekstrak fitur dari {apk_path}")
            return None

    return features if features else None
```

Gambar 3.4 Ekstraksi Fitur dari *File* APK

Dataset *Malware* dalam penelitian ini mencakup beragam kategori serangan, antara lain: SMS *Malware*, *Banking Malware*, Adware, Scareware, Ransomware, dan Riskware. Masing-masing kategori *Malware* disimpan dalam direktori terpisah selama tahap pra-pemrosesan untuk memudahkan pengelolaan data. Namun, pada tahap akhir pembentukan dataset, seluruh kategori *Malware* tersebut digabungkan ke dalam satu label umum, yaitu *Malware*, guna menyederhanakan proses

klasifikasi dan memungkinkan model fokus dalam mendeteksi *Malware* secara umum.

Adapun struktur direktori Dataset yang digunakan dapat dijelaskan sebagai berikut:

1) Dataset dari CIC-AndMal2017 terdiri atas direktori *Benign*_2017, *Benign*_2016, *Benign*_2015, *Adware-APKs*, *SMSMalware*, *Scareware*, dan *Ransomware*.
2) Dataset dari CICMalDroid2020 mencakup direktori *Benign, Banking, SMS, Adware,* dan *Riskware.*
3) Dataset dari AndroZoo disimpan dalam direktori *Benign*_AndroZoo, yang berisi aplikasi *benign* hasil pengambilan acak dari repositori aplikasi Android terbesar tersebut.

Struktur direktori yang digunakan pada proses ini ditunjukkan pada Gambar 3.5, yang menampilkan susunan folder berdasarkan jenis Dataset dan kategori aplikasinya.

Gambar 3.5 Struktur Direktori Dataset *Malware* dan *Benign*

Penggabungan data dari berbagai sumber dan tahun ini memberikan keragaman pola fitur, yang pada akhirnya meningkatkan kemampuan generalisasi model dalam mendeteksi aplikasi berbahaya.

Hasil ekstraksi fitur menghasilkan Dataset berdimensi besar, baik dari segi jumlah fitur maupun jumlah File aplikasi yang diproses. Penyimpanan dalam format konvensional seperti .csv atau .xlsx dinilai kurang efisien, karena menghasilkan ukuran File yang sangat besar serta memperlambat proses pembacaan dan manipulasi data dalam analisis selanjutnya. Oleh karena itu, Dataset akhir disimpan menggunakan format .parquet.

Parquet merupakan format penyimpanan berbasis kolom (*columnar storage*) yang optimal untuk pemrosesan data berskala besar. Format ini memiliki keunggulan dalam hal kecepatan akses, efisiensi ruang penyimpanan, serta dukungan kompresi seperti Snappy yang memungkinkan pengurangan ukuran File tanpa mengorbankan integritas data. Proses konversi dan penyimpanan ke format .parquet menggunakan pustaka pandas dan pyarrow, sebagaimana ditunjukkan pada Gambar 3.6.

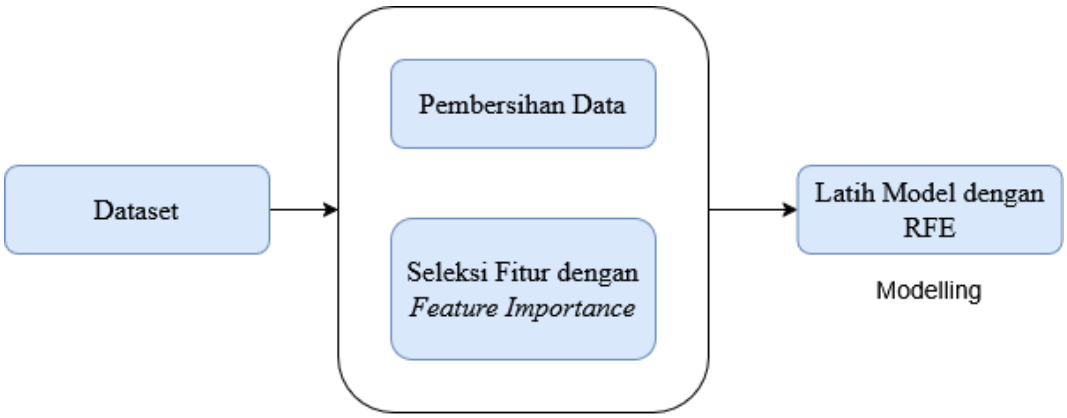
```python
# Gunakan PyArrow Table untuk optimasi penyimpanan
schema = pa.schema([pa.field("APK_Name", pa.string())] +
                   [pa.field(feature, pa.bool_()) for feature in all_features] +
                   [pa.field("Class", pa.dictionary(pa.int8(), pa.string()))])

table = pa.Table.from_pandas(pd.DataFrame(dataset, columns=["APK_Name"] + all_features + ["Class"]), schema=schema)

# Simpan sebagai Parquet dengan kompresi Snappy
pq.write_table(table, output_parquet, compression="snappy")

print(f"✅ Dataset berhasil disimpan di {output_parquet}")
```

Gambar 3.6 Kode Menyimpan Hasil Ekstraksi Fitur ke dalam Format .parquet

Dengan demikian, proses pembentukan Dataset dalam penelitian ini telah dilakukan secara menyeluruh dan sistematis, dimulai dari pengumpulan File APK dari berbagai sumber, ekstraksi fitur menggunakan pustaka Python, pelabelan kelas berdasarkan struktur direktori, hingga penyimpanan hasil akhir dalam format .parquet yang efisien dan siap digunakan untuk pelatihan serta pengujian model deteksi *Malware* berbasis machine learning.

c. Data Exploration

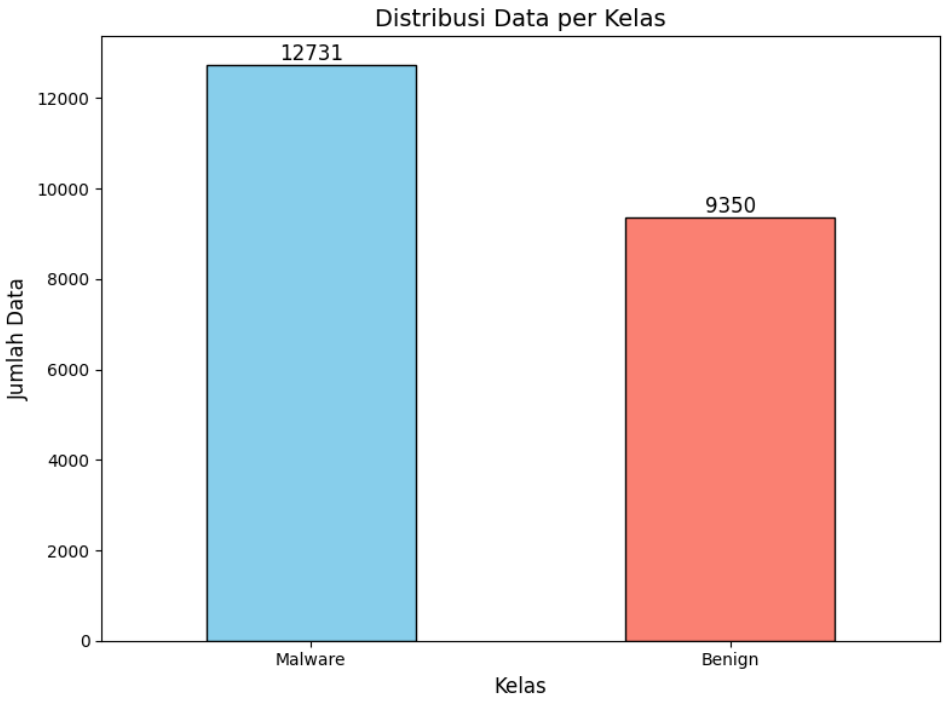
Pada tahap *Data Exploration*, dilakukan analisis yang mendalam untuk memahami pola dan hubungan dalam data yang telah dikumpulkan. Selain itu,



Gambar 3.7 Diagram Arsitektur *Data Exploration*

tahap ini juga melibatkan proses pembersihan data, dimana data yang tidak lengkap, duplikat, atau tidak relevan diidentifikasi dan dihapus agar hanya data yang bersih dan valid yang digunakan untuk pelatihan model. Pembersihan data ini penting untuk memastikan bahwa model yang dibangun tidak dipengaruhi oleh kesalahan atau ketidaksesuaian dalam data. Gambar 3.7 menunjukkan proses *Data Exploration* yang dilakukan.

Distribusi data per kelas menunjukkan jumlah sampel yang terkumpul untuk masing-masing kategori. Kelas "*Malware*" terdiri dari 12.731 sampel, sementara kelas "*Benign*" memiliki 9.350 sampel. Pembagian data ini menjadi dasar dalam proses pemilihan fitur dan pemodelan berikutnya. Hal ini dapat dilihat pada diagram batang yang ditampilkan pada Gambar 3.8.



Gambar 3.8 Distribusi Data per Kelas

Pada tahap ini juga dilakukan proses seleksi fitur (*feature selection*), yakni proses identifikasi dan pemilihan fitur-fitur yang paling relevan serta berpengaruh signifikan terhadap hasil prediksi terhadap ancaman *Malware*. Tujuan utama dari seleksi fitur ini adalah untuk menurunkan kompleksitas data, meningkatkan akurasi model prediktif, dan mempercepat waktu pelatihan model. Sebelum memasuki tahap pemodelan akhir yang akan digunakan untuk proses deployment, dilakukan

terlebih dahulu pemodelan awal guna memperoleh nilai *feature importance*. Dari lebih dari 100.000 fitur awal, disaring menjadi 1.000 fitur terpilih yang kemudian direduksi kembali menggunakan metode *Recursive Feature Elimination* (RFE) sebagai bagian dari tahap pemodelan akhir. Setelah nilai *feature importance* diperoleh, dilakukan proses pelatihan model (*train model*) menggunakan fitur hasil seleksi RFE, yang akan dibahas lebih lanjut pada bagian Pembahasan.

d. Modelling

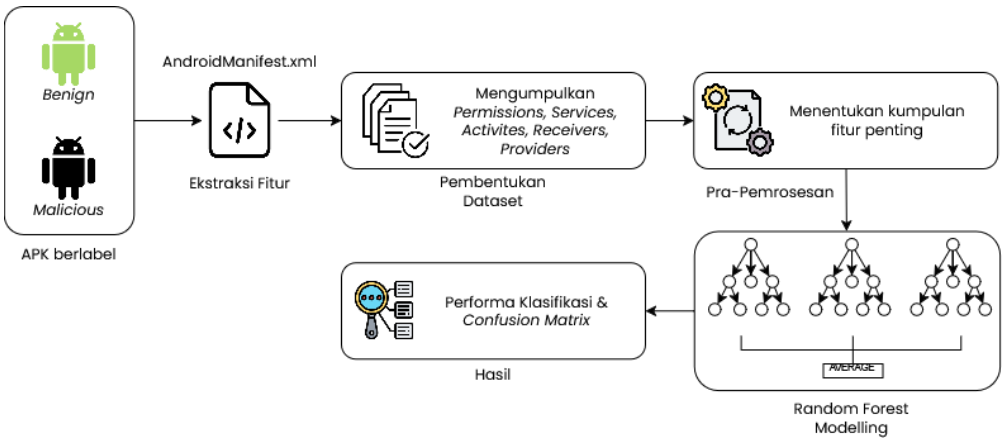
Setelah data dibersihkan dan fitur yang relevan dipilih, tahap berikutnya adalah *Modelling*. Pada tahap ini, algoritma *Random Forest* dipilih untuk mendeteksi *Malware* karena kemampuannya dalam menangani data yang kompleks dan memberikan prediksi yang akurat. *Random Forest* menggunakan teknik *ensemble learning*, di mana beberapa pohon keputusan (*decision trees*) digabungkan untuk memberikan klasifikasi yang lebih baik. Sebelum pelatihan model dilakukan, Dataset dibagi menjadi dua bagian: 80% untuk data pelatihan (*training set*) dan 20% untuk data pengujian (*testing set*). Pembagian ini penting untuk memastikan bahwa model dapat dilatih dengan data yang cukup, namun juga diuji dengan data yang belum pernah dilihat sebelumnya, sehingga dapat mengevaluasi kemampuan model dalam mendeteksi *Malware* pada data yang baru dan tidak dikenal.

e. Model Evalution

Setelah model dilatih, tahap *Model Evaluation* dilakukan untuk menilai kinerja model dengan menggunakan metrik yang dihitung melalui *confusion matrix*, seperti akurasi, presisi, recall, dan F1 score. Tahap terakhir dalam pengembangan model ini adalah *Deployment*, di mana model yang telah dilatih dan dievaluasi akan diimplementasikan ke dalam aplikasi *mobile* berbasis Android. Setelah model selesai dilatih, model tersebut akan dikonversi menjadi File dengan ekstensi .onnx (Open Neural Network Exchange), yang memungkinkan integrasi model *Machine Learning* ke dalam aplikasi Android. Ekstensi .onnx digunakan karena format ini mendukung interoperabilitas lintas platform, yang memungkinkan model yang dibangun dapat dijalankan pada berbagai perangkat, termasuk Android. Dengan menggunakan library ONNX, model *Machine Learning* dapat dikonversi ke dalam

format .onnx, yang kemudian dapat digunakan dalam aplikasi Android untuk mendeteksi *Malware* secara langsung.

Sebagai kelanjutan dari tahapan pengembangan model yang telah dijelaskan sebelumnya, proses perancangan sistem deteksi *Malware* dirangkum secara sistematis dalam bentuk diagram arsitektur, yang ditampilkan pada Gambar 3.9. Diagram ini menggambarkan alur kerja mulai dari ekstraksi fitur, pembersihan dan pemrosesan data, hingga evaluasi performa model menggunakan metrik seperti akurasi, presisi, recall, dan *confusion matrix*. Visualisasi ini membantu memahami tahapan-tahapan penting dalam membangun model deteksi *Malware* berbasis *Machine Learning* secara terstruktur dan efektif.

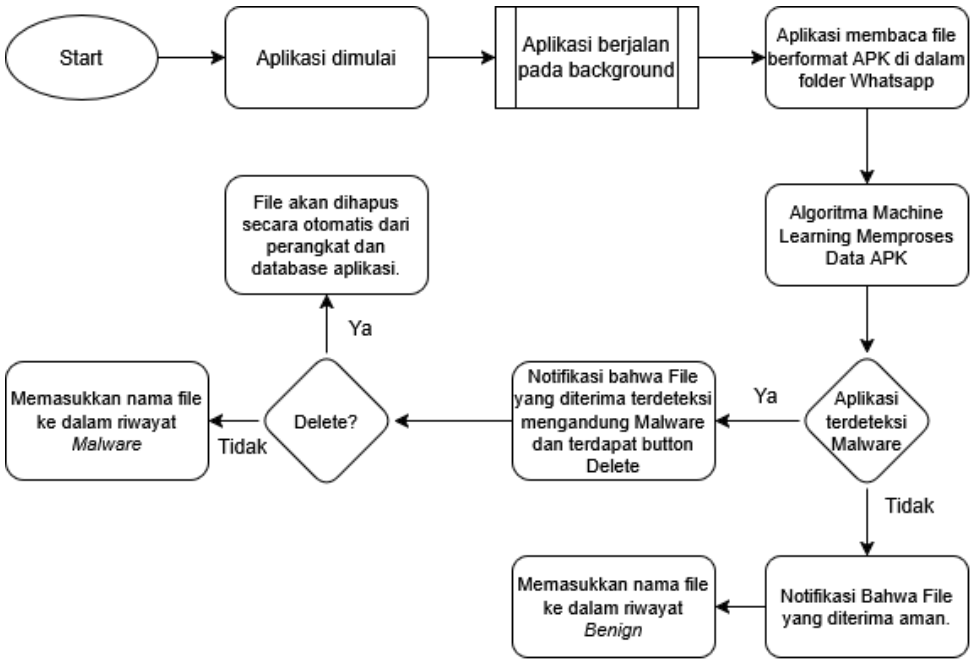


Gambar 3.9 Diagram Arsitektur Perancangan Model

3.1.2.2 Perancangan Aplikasi

Pada tahap perancangan aplikasi, peneliti merancang aplikasi Android berbasis *machine learning* yang diberi nama SAMAWA (*Scan & Alert for Malware on WhatsApp*). Aplikasi ini dirancang untuk mendeteksi malware pada File .apk yang diterima melalui pesan *WhatsApp*. Pada tahap perancangan aplikasi, aplikasi *mobile* berbasis Android ini dikembangkan menggunakan Android Studio, *Integrated Development Environment* (IDE) yang khusus dirancang untuk pengembangan aplikasi Android. Aplikasi ini dirancang untuk berfungsi sebagai alat filtrasi *Malware*, yang secara otomatis bekerja di *background process* untuk mendeteksi potensi ancaman *Malware* pada File APK yang diterima melalui WhatsApp.

Diagram alir dari perancangan aplikasi yang akan dibuat dapat dilihat pada gambar 3.10

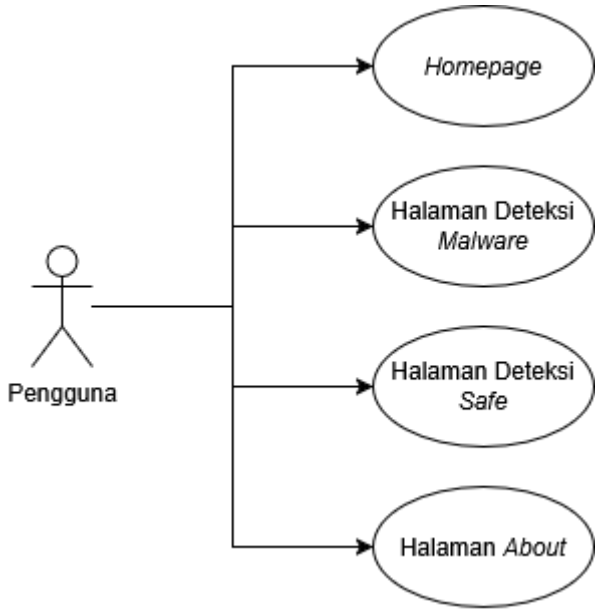


Gambar 3.10 Diagram Alir Aplikasi SAMAWA

Aplikasi ini secara khusus akan membaca File-File APK yang tersimpan dalam folder WhatsApp *Documents*, yang merupakan lokasi di mana File yang diterima melalui aplikasi WhatsApp biasanya disimpan. Setelah File APK terdeteksi, algoritma *Machine Learning* yang telah dilatih sebelumnya akan memproses data dari File APK tersebut untuk menganalisis fitur-fitur yang relevan, seperti izin aplikasi dan *services*.

Berdasarkan analisis tersebut, algoritma akan mengklasifikasikan File APK tersebut apakah termasuk kategori *Malware* (berbahaya) atau *benign* (jinak). Jika File APK terdeteksi sebagai *Malware*, aplikasi akan memberikan notifikasi kepada pengguna untuk memperingatkan adanya potensi ancaman pada File yang diterima. Selain itu, jika File APK terdeteksi sebagai *benign*, aplikasi juga akan memberikan notifikasi yang memberitahukan pengguna bahwa File tersebut aman dan tidak mengandung ancaman.

Seluruh proses ini berjalan di latar belakang tanpa melibatkan interaksi manual dari pengguna, sehingga pengguna tidak perlu khawatir memeriksa setiap File yang diterima. Aplikasi ini memastikan deteksi *Malware* dilakukan secara otomatis, mempermudah dan mempercepat proses tanpa mengganggu pengalaman pengguna, serta memberikan pemberitahuan yang jelas mengenai status File yang diterima, apakah itu berbahaya atau aman. Untuk menggambarkan hubungan antara aktor dan sistem dalam menjalankan fungsionalitas tersebut, digunakan *Use Case Diagram* yang memperlihatkan peran masing-masing aktor serta fitur-fitur utama yang tersedia pada aplikasi. Diagram ini membantu dalam memahami batasan tanggung jawab dan interaksi yang terjadi antara pengguna dan sistem. *Use Case Diagram* dapat dilihat pada Gambar 3.11.



Gambar 3.11 *Use Case Diagram* Aplikasi SAMAWA

Berdasarkan *Use Case Diagram* diatas, terdapat empat *use case* (halaman atau fitur) yang dapat diakses oleh pengguna:

1. Homepage

   Halaman awal aplikasi yang berfungsi sebagai pusat navigasi atau tampilan utama setelah pengguna membuka aplikasi. Dari sini, pengguna dapat memilih untuk menuju ke halaman-halaman lain yang tersedia.

2. Halaman Deteksi *Malware*

Halaman ini menampilkan daftar aplikasi yang terdeteksi sebagai malware oleh sistem. Pengguna dapat memeriksa aplikasi berbahaya yang teridentifikasi berdasarkan hasil analisis model machine learning.

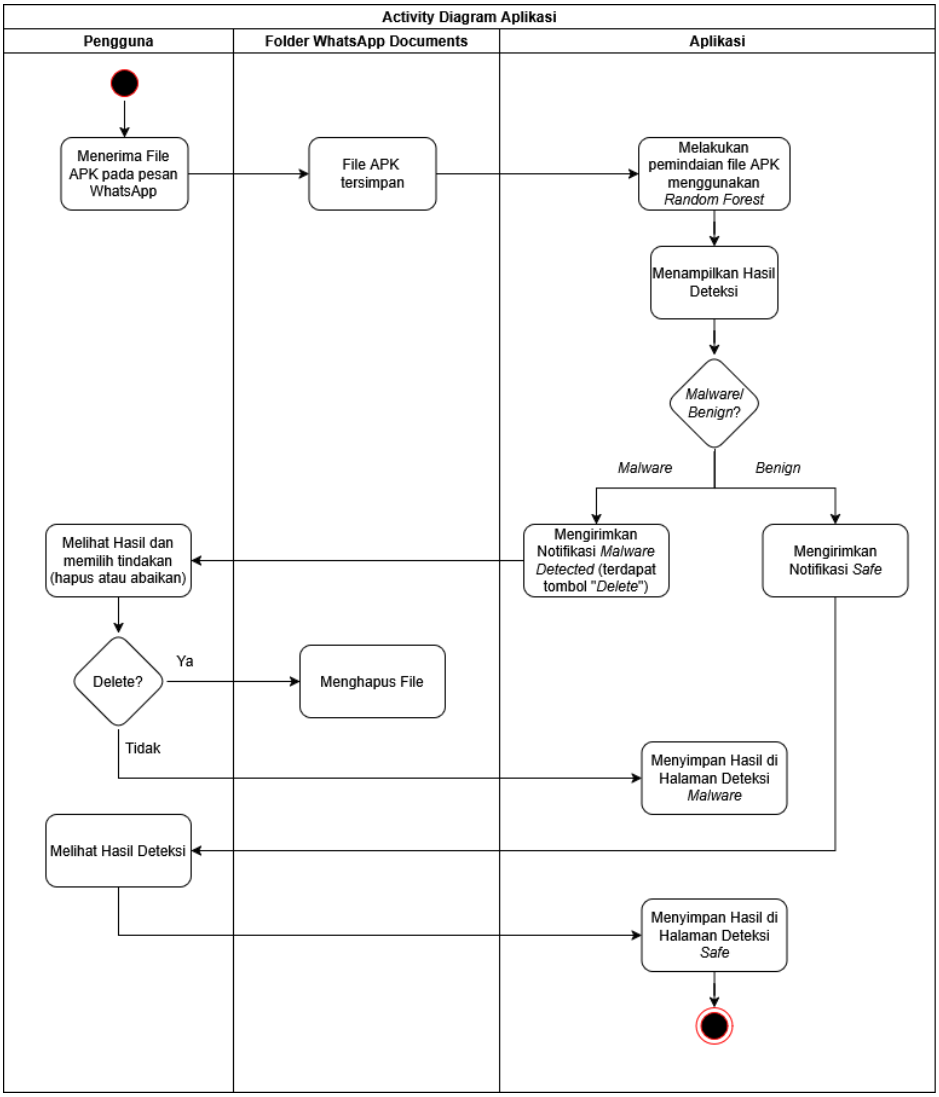
3. Halaman Deteksi *Benign*

   Halaman ini berisi daftar aplikasi yang diklasifikasikan sebagai aman (benign). Ini membantu pengguna mengetahui aplikasi mana saja yang tidak memiliki indikasi *malware*.

4. Halaman *About*

   Menyediakan informasi mengenai aplikasi, seperti tujuan pengembangan, teknologi yang digunakan, tim pengembang, atau cara kerja aplikasi dalam mendeteksi malware.

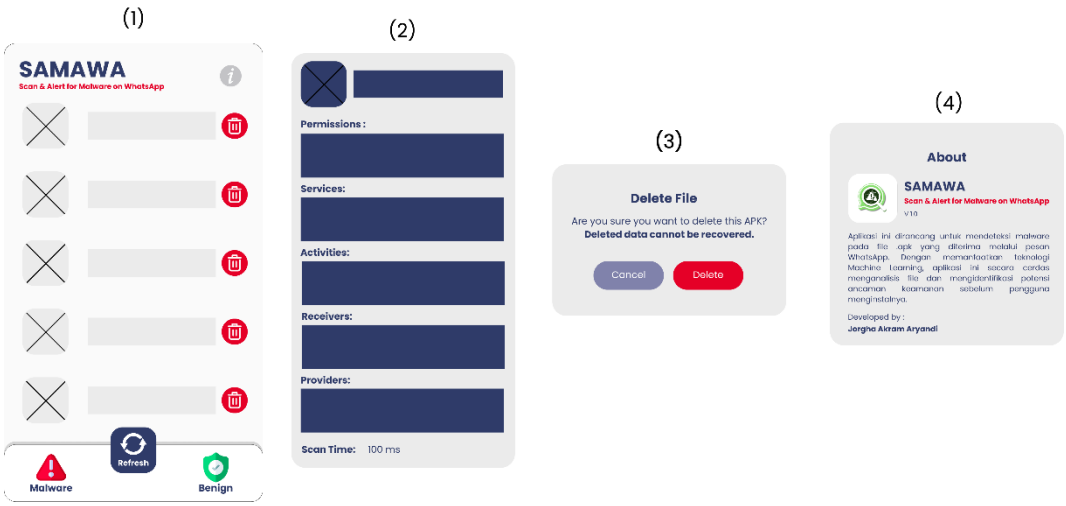
Selanjutnya, *Activity Diagram* digunakan untuk menjelaskan alur proses yang berjalan dalam sistem, termasuk tahapan-tahapan deteksi otomatis, pengambilan keputusan berdasarkan hasil analisis, dan pemberitahuan kepada pengguna. Diagram ini memberikan gambaran logis mengenai urutan aktivitas yang terjadi dalam sistem, baik secara internal maupun yang melibatkan pengguna.



Gambar 3.12 *Activity Diagram* Aplikasi SAMAWA

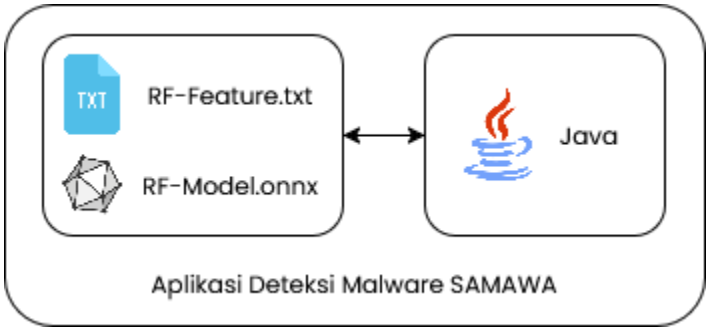
Gambar 3.13 adalah desain antarmuka pengguna aplikasi yang akan dikembangkan, yang disusun dengan mempertimbangkan kebutuhan aplikasi berdasarkan diagram sistem dan diagram alir yang telah dibuat, serta mendukung kenyamanan pengguna dalam mengakses informasi terkait status keamanan File secara intuitif.

Gambar 3.13 adalah desain antarmuka pengguna aplikasi yang akan dikembangkan memperhitungkan kebutuhan aplikasi berdasarkan diagram sistem dan diagram alir yang telah dibuat.



Gambar 3.13 Desain Antarmuka Aplikasi (1), Desain Dialog Detail (2), Desain *Delete Confirmation File* (3), Desain *About* Aplikasi (4)

Aplikasi android didesain dengan spesifikasi minimal android versi 9 dan target android versi 15 didesain menggunakan IDE Android Studio Ladybug 2024.2.1 Patch 2. Bahasa pemrograman yang digunakan java yang nantinya dapat mengakses model *machine learning* yang berformat ONNX. Adapun diagram yang menjelaskan hubungan antara model yang dibangun dengan aplikasi yang dibangun terdapat pada Gambar 3.14.



Gambar 3.14 Diagram Hubungan Model dengan Aplikasi Android

3.1.3 Pengembangan Sistem

Pada tahap pengembangan sistem, peneliti menggunakan metode pengembangan *Prototype*. Metode ini memungkinkan pengembang untuk membuat model awal (*prototyping*) dari sistem yang dapat diuji dan dinilai oleh pengguna sejak awal proses pengembangan. Menurut (Pricillia & Zulfachmi, 2021), model pengembangan Prototype lebih cocok untuk sistem atau perangkat lunak yang bersifat kustomisasi, yaitu perangkat lunak yang dirancang berdasarkan permintaan dan kebutuhan tertentu, termasuk situasi atau kondisi spesifik. Melalui pengembangan *prototype*, pengembang dan pengguna dapat berinteraksi secara langsung, memberikan *feedback*, serta melakukan penyesuaian secara fleksibel sesuai dengan kebutuhan pengguna. Hal ini bertujuan agar sistem yang dikembangkan tidak hanya sesuai dengan harapan pengguna, tetapi juga mampu mengimplementasikan metode atau algoritma tertentu pada kasus tertentu, sehingga dapat menghasilkan solusi yang lebih tepat sasaran

Prototipe sendiri adalah model awal dari sebuah sistem yang dapat digunakan untuk menggambarkan fitur atau tampilan sistem yang akan dikembangkan lebih lanjut. Prototipe ini memberikan gambaran yang lebih konkret tentang bagaimana sistem bekerja, serta mengidentifikasi potensi masalah dan kekurangan yang perlu diperbaiki sebelum pengembangan lebih lanjut. Dengan mengikuti metode pengembangan prototipe, peneliti dapat melakukan penyesuaian dan perubahan pada sistem secara lebih fleksibel berdasarkan umpan balik pengguna, yang pada akhirnya menghasilkan sistem yang lebih baik dan lebih sesuai dengan kebutuhan pengguna (Alda dkk., 2024).

3.1.3.1 Tahap *Communication*

Pada Tahap *Communication* dalam pengembangan sistem menggunakan metode *Prototyping*, pengembang berkomunikasi secara intensif dengan pengguna untuk menggali kebutuhan dan harapan mereka terhadap sistem yang akan dikembangkan. Selain berkomunikasi langsung dengan pengguna, tahap ini juga melibatkan kajian terhadap penelitian terdahulu yang relevan dengan topik pengembangan sistem. Hal ini penting agar pengembang dapat memahami

pendekatan dan solusi yang telah diuji sebelumnya serta mengidentifikasi potensi perbaikan atau inovasi dalam sistem yang akan dikembangkan.

Setelah melalui diskusi yang mendalam dan kajian literatur terkait, keputusan yang dihasilkan adalah pengembangan model algoritma *Random Forest* untuk mendeteksi *Malware* pada File APK yang diterima melalui aplikasi WhatsApp. Keputusan ini didukung oleh penelitian terdahulu yang menunjukkan bahwa *Random Forest* adalah algoritma pembelajaran mesin yang efektif untuk masalah klasifikasi, terutama dalam mendeteksi ancaman *Malware* pada berbagai format File, termasuk APK.

Random Forest dipilih karena kemampuannya dalam mengklasifikasikan data yang sangat kompleks dan kemampuannya untuk bekerja dengan berbagai jenis fitur (seperti izin aplikasi, struktur File, dll) yang relevan untuk mendeteksi potensi ancaman *Malware*.

Melalui kajian terhadap penelitian sebelumnya dan komunikasi dengan pengguna, keputusan ini memberikan arah yang jelas bagi pengembang untuk fokus pada implementasi dan pengujian model *Random Forest* dalam pengembangan prototipe sistem. Model ini akan diuji lebih lanjut pada tahap berikutnya, dengan mempertimbangkan hasil dan temuan dari penelitian sebelumnya untuk memastikan efektivitasnya dalam mendeteksi *Malware* pada File APK yang diterima melalui WhatsApp.

3.1.3.2 Tahap *Quick Plan*

Pada tahap *Quick Plan* dalam pengembangan sistem menggunakan metode Prototyping, pengembang merancang rencana awal untuk sistem yang akan dikembangkan. Pada tahap ini, pengembangan fokus pada dua hal utama: pengembangan model untuk filtrasi konten *Malware*, khususnya pada File APK, dan perancangan aplikasi berbasis Android.

Pengembangan Model untuk Filtrasi *Malware* Model untuk filtrasi *Malware* menggunakan algoritma klasifikasi *Random Forest* sebagai algoritma utama dalam mendeteksi *Malware* pada File APK. *Random Forest* dipilih karena kemampuannya yang terbukti efektif dalam menangani masalah klasifikasi yang

kompleks, terutama dalam analisis File APK. Data dari file APK akan diproses melalui ekstraksi fitur, yang mencakup analisis terhadap file *AndroidManifest.xml*, yaitu *permissions, services, activities, receivers*, dan *providers*.. Proses ekstraksi fitur ini bertujuan untuk mengekstrak informasi penting yang relevan untuk mendeteksi apakah File tersebut mengandung *Malware* atau tidak.

Setelah ekstraksi fitur dilakukan, data akan diproses melalui model *Random Forest* yang telah dilatih sebelumnya. Model ini akan mengklasifikasikan File APK sebagai *Malware* atau *benign* berdasarkan fitur yang diekstraksi. Proses ini akan berlangsung secara otomatis di latar belakang, tanpa memerlukan interaksi langsung dari pengguna, memastikan bahwa File APK yang diterima melalui WhatsApp dapat dianalisis dan diklasifikasikan dengan cepat.

Pengembangan Aplikasi Android Untuk pengembangan aplikasi Android, peneliti menggunakan desain aplikasi *Model-View-Controller* (MVC) sebagai pola desain utama aplikasi yang dikembangkan. MVC adalah salah satu pola desain perangkat lunak (*software design pattern*) yang sering digunakan untuk menciptakan kode yang lebih terorganisir dan mudah dikelola. MVC memisahkan aplikasi menjadi tiga komponen utama, yaitu:

a. *Model*: Komponen ini berfungsi untuk mengelola data dan logika aplikasi. Dalam konteks aplikasi ini, Model akan menangani proses ekstraksi fitur dan pengklasifikasian File APK menggunakan algoritma *Random Forest*.
b. *View*: Komponen ini berfungsi untuk menampilkan antarmuka pengguna (user interface/UI). View akan menampilkan hasil dari klasifikasi, seperti notifikasi yang memberitahukan pengguna jika File APK yang diterima terdeteksi sebagai *Malware* atau *benign*.
c. *Controller*: Komponen ini bertindak sebagai penghubung antara Model dan View. *Controller* akan menangani interaksi pengguna dan mengarahkan data ke Model untuk diproses, serta mengarahkan hasilnya ke View untuk ditampilkan kepada pengguna.

Pemilihan desain MVC didasarkan pada keuntungan yang ditawarkannya dalam mengorganisasi aplikasi menjadi bagian-bagian yang terpisah dan mudah dikelola. Dengan menggunakan MVC, kode aplikasi menjadi lebih terstruktur, memudahkan

pemeliharaan dan pengembangan lanjutan, serta memastikan bahwa perubahan pada satu bagian tidak berdampak besar pada bagian lainnya. Seperti yang dijelaskan oleh Fadhullah, dkk. (2023), penggunaan MVC dalam pengembangan perangkat lunak dapat menghasilkan aplikasi yang lebih mudah dipelihara dan lebih fleksibel dalam menghadapi perubahan atau penambahan fitur.

3.1.3.4 Tahap *Construction of Prototype*

Pada tahap *Construction of Prototype*, peneliti mulai membangun prototipe sistem berdasarkan rencana yang telah disusun sebelumnya pada tahap *Quick Plan*. Tujuan utama dari tahap ini adalah untuk menciptakan versi awal dari sistem yang dapat diuji dan digunakan oleh pengguna, memberikan gambaran mengenai bagaimana sistem akan bekerja pada tahap selanjutnya.

Pada tahap ini, pengembangan sistem dilakukan dengan fokus pada pembuatan fungsionalitas dasar yang mencakup algoritma deteksi *Malware* dengan model *Random Forest*, serta implementasi aplikasi berbasis Android. Pengembang akan mulai dengan implementasi algoritma *Random Forest* untuk memproses data File APK yang diterima melalui WhatsApp. Proses ekstraksi fitur, yang melibatkan analisis izin aplikasi, dan fitur statis lainnya, juga akan diintegrasikan ke dalam prototipe aplikasi.

Di sisi aplikasi Android, pengembang akan mulai membangun antarmuka pengguna (*View*) yang akan menampilkan hasil klasifikasi apakah File APK tersebut tergolong *Malware* atau *benign*. Komponen *Controller* juga akan dikembangkan untuk menghubungkan antara model deteksi *Malware* dan tampilan hasil yang diberikan kepada pengguna.

Prototipe ini akan mengintegrasikan seluruh komponen dasar dari aplikasi dan memungkinkan pengujian fungsi-fungsi utama seperti deteksi *Malware*, pemberitahuan kepada pengguna, dan kemampuan aplikasi untuk berjalan di latar belakang. Dengan adanya prototipe ini, pengembang dan pengguna dapat melakukan uji coba sistem dan mengidentifikasi area yang perlu diperbaiki atau disesuaikan.

3.1.3.5 Tahap *Development, Delivery & Feedback*

Pada tahap terakhir, pengujian dilakukan dengan menggunakan parameter yang telah ditetapkan untuk mengevaluasi keandalan model deteksi *Malware* dan kinerja aplikasi secara keseluruhan. Pengujian bertujuan untuk memastikan bahwa algoritma *Random Forest* yang digunakan dalam aplikasi dapat mengklasifikasikan File APK dengan akurat sebagai *Malware* atau *benign* (jinak). Selain itu, pengujian juga dilakukan untuk mengevaluasi hasil pengujian *black box*.

Setelah pengujian dan perbaikan dilakukan, aplikasi kemudian disiapkan untuk delivery ke pengguna. Aplikasi ini langsung diinstal pada perangkat yang digunakan untuk pengujian atau distribusi terbatas kepada pengguna tertentu, seperti pengguna dalam lingkungan uji coba atau kelompok pengguna yang relevan.

Pada tahap feedback, pengembang mengumpulkan masukan dari pengguna mengenai kinerja aplikasi, kemudahan penggunaan, serta masalah atau bug yang ditemukan selama penggunaan. Umpan balik ini sangat penting untuk menyempurnakan aplikasi dan memastikan deteksi *Malware* yang lebih akurat, serta memastikan bahwa aplikasi dapat berjalan dengan stabil dan memberikan manfaat maksimal bagi pengguna.

3.1.4 Pengujian dan Evaluasi Sistem

Pengujian dan evaluasi sistem sangat penting untuk menentukan apakah hasil rancangan prototipe memenuhi tujuan penelitian. Pengujian ini dilakukan untuk mengevaluasi kinerja dari sistem yang dikembangkan, terutama untuk memastikan bahwa aplikasi dapat mendeteksi *Malware* dengan akurat dan efisien. Pengujian dilakukan pada beberapa bagian sistem, salah satunya adalah Pengujian Model, yang bertujuan untuk mengukur kinerja algoritma deteksi *Malware* yang digunakan dalam aplikasi.

3.1.4.1 Pengujian Model

Pada pengujian model ini, *Confusion* Matrix digunakan untuk mengevaluasi hasil prediksi model algoritma *Random Forest* dalam mendeteksi *Malware* pada File APK. *Confusion* Matrix adalah matriks yang digunakan untuk menampilkan jumlah data uji yang diklasifikasikan dengan benar dan salah. Hal ini memudahkan

evaluasi akurasi sistem klasifikasi dan membantu dalam menganalisis kinerja model secara lebih detail. *Confusion* Matrix menggambarkan seberapa banyak prediksi yang benar-benar sesuai dengan label yang sebenarnya (*ground truth*) dan berapa banyak yang salah, sehingga memberikan gambaran yang lebih jelas mengenai kekuatan dan kelemahan model yang digunakan (Robi & Kania, 2023).

Dari *confusion matrix*, beberapa metrik kinerja yang penting dapat dihitung, antara lain:

a. Akurasi (*Accuracy*): Mengukur seberapa baik model dalam memprediksi dengan benar. Dihitung dengan rumus:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

di mana:

1) TP = *True Positives* (prediksi benar bahwa File adalah *Malware*)
2) TN = *True Negatives* (prediksi benar bahwa File adalah *benign*)
3) FP = *False Positives* (prediksi salah bahwa File adalah *Malware*, padahal *benign*)
4) FN = *False Negatives* (prediksi salah bahwa File adalah *benign*, padahal *Malware*)

b. Presisi (*Precision*): Mengukur seberapa banyak prediksi positif yang benar dibandingkan dengan semua prediksi positif. Dihitung dengan rumus:

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Presisi menunjukkan berapa banyak dari File yang diprediksi sebagai *Malware* yang benar-benar *Malware*, sehingga dapat memberikan indikasi mengenai tingkat kesalahan dalam memberi label "*Malware*".

c. Recall (Sensitivitas): Mengukur seberapa banyak File *Malware* yang berhasil ditemukan oleh model dibandingkan dengan semua File yang benar-benar *Malware*. Dihitung dengan rumus:

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

Recall penting untuk memastikan bahwa model tidak melewatkan *Malware* yang ada dalam data dan dapat mendeteksi sebanyak mungkin ancaman.

d. F1-Score: Merupakan rata-rata harmonis antara presisi dan recall, memberikan gambaran yang lebih jelas mengenai keseimbangan antara keduanya. Dihitung dengan rumus:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

F1-Score sangat berguna ketika kita perlu menyeimbangkan presisi dan recall, terutama pada Dataset yang tidak seimbang, di mana salah satu metrik bisa sangat tinggi sementara yang lainnya rendah

3.1.4.2 Pengujian Aplikasi

Pada tahap pengujian aplikasi, peneliti menggunakan metode pengujian *black box* untuk mengevaluasi fungsionalitas aplikasi tanpa melihat atau mengakses kode programnya. Pengujian ini berfokus pada input yang diberikan kepada aplikasi dan output yang dihasilkan, serta bagaimana aplikasi merespons berdasarkan fungsionalitas yang telah dirancang. Dalam pengujian *black box*, penguji tidak perlu mengetahui struktur internal aplikasi, karena pengujian berfokus pada pengujian fungsionalitas dan apakah aplikasi memenuhi tujuan yang telah ditentukan sebelumnya.

Pengujian Black Box adalah teknik pengujian perangkat lunak yang digunakan untuk menguji aplikasi dari sisi pengguna. Pengujian ini dilakukan dengan mengevaluasi apakah aplikasi memberikan hasil yang diharapkan saat diberikan input tertentu, tanpa memperhatikan bagaimana hasil tersebut dicapai di dalam kode. Pengujian ini umumnya dilakukan pada tingkat sistem atau antarmuka pengguna.

Beberapa parameter yang diuji pada metode black box mencakup:

a. Skenario Uji: Peneliti membuat skenario uji untuk memverifikasi apakah aplikasi berfungsi sesuai dengan tujuan. Skenario ini mencakup berbagai kondisi pengujian yang mewakili berbagai skenario penggunaan aplikasi oleh pengguna.
    1) Pengujian dilakukan sebanyak 15 pengujian menggunakan 1 perangkat smartphone berbasis Android sebagai penerima dan 1 buah komputer sebagai pengirim.
    2) Android yang dipakai pada perangkat smartphone menggunakan android versi 15 dengan 8 GB RAM, Prosessor Snapdragon 625, Memori 128 GB.
    3) Komputer yang digunakan menggunakan operating system windows dengan 16 GB RAM, Prosessor Intel Pentium, Harddisk 512 GB.
    4) Versi Aplikasi WhatsApp pada smartphone 2.25.12.74 dan versi aplikasi WhatsApp Web pada komputer yakni versi 2.3000.1022289489.
    5) Jaringan yang digunakan pada pengiriman apk dari komputer ke smartphone menggunakan jaringan wifi yang terkoneksi internet, baik perangkat smartphone maupun komputer terhubung dengan access point wifi yang sama.
    6) Fokus pengujian diarahkan pada dua fitur utama, yaitu Halaman Antarmuka Aplikasi dan File *Watcher Service*. Halaman Antarmuka Aplikasi mencakup pengujian terhadap seluruh fitur yang tersedia secara langsung pada aplikasi, untuk memastikan bahwa setiap fungsi dapat diakses dan berjalan dengan baik oleh pengguna. Sementara itu, File *Watcher Service* berperan sebagai layanan latar belakang (*background service*) yang mendeteksi keberadaan File APK yang diterima melalui pesan WhatsApp. Pengiriman File ap
b. *Expected Result* (Hasil yang Diharapkan): Setiap skenario uji memiliki hasil yang diharapkan, yang berfungsi sebagai acuan untuk membandingkan apakah aplikasi memberikan hasil yang benar. Misalnya, jika aplikasi menerima File APK yang telah diberi label sebagai *Malware*, maka hasil

yang diharapkan adalah aplikasi akan memberikan peringatan atau menandai File tersebut sebagai *Malware*.

c. Pengujian Fungsionalitas: Pada pengujian ini, fungsionalitas aplikasi diuji, seperti apakah aplikasi mampu memproses File APK, apakah deteksi *Malware* bekerja dengan baik, dan apakah sistem memberikan umpan balik yang tepat kepada pengguna.
d. Pengujian Antarmuka Pengguna: Pengujian ini melibatkan evaluasi antarmuka pengguna aplikasi untuk memastikan kemudahan penggunaan dan kelancaran interaksi pengguna dengan aplikasi.

Dalam penelitian ini, pengujian juga melibatkan pengujian aplikasi melalui pengiriman File APK menggunakan aplikasi WhatsApp untuk menguji integritas antara model deteksi *Malware* dan aplikasi. Peneliti akan mengirimkan File APK yang telah diberi label sebagai aman dan berbahaya melalui WhatsApp, untuk memastikan apakah aplikasi mampu mendeteksi File yang berbahaya dan memberikan umpan balik yang tepat kepada pengguna. Berikut adalah beberapa skenario tes yang digunakan:

a. Skenario 1: Pengiriman File APK yang aman melalui WhatsApp.
   Expected result: Aplikasi harus menandai File sebagai aman dan tidak memberikan peringatan.
b. Skenario 2: Pengiriman File APK yang berbahaya melalui WhatsApp. *Expected result*: Aplikasi harus mendeteksi File tersebut sebagai *Malware* dan memberikan peringatan kepada pengguna.

Pengujian ini memastikan bahwa integrasi antara model deteksi *Malware* dan aplikasi berjalan dengan baik, dan aplikasi dapat memberikan informasi yang akurat kepada pengguna berdasarkan hasil deteksi. Dengan demikian, pengujian black box ini memastikan bahwa aplikasi berfungsi sebagaimana mestinya dan dapat memenuhi harapan pengguna dalam konteks deteksi *Malware* melalui File yang dikirimkan di WhatsApp.

3.1.5 Penulisan Laporan

Tahap terakhir dalam penelitian ini adalah penulisan laporan yang akan disusun dalam bentuk skripsi. Laporan ini bertujuan untuk mendokumentasikan seluruh rangkaian proses penelitian secara sistematis. Skripsi akan mencakup pendahuluan yang menjelaskan latar belakang, tujuan, dan manfaat penelitian, serta tinjauan pustaka yang membahas teori-teori dasar, algoritma yang digunakan, dan teknologi yang diterapkan. Bagian metode penelitian akan menguraikan metode pengembangan sistem yang diterapkan, termasuk tahapan-tahapan dalam pengembangan aplikasi dan pengujian yang telah dilakukan. Hasil dan pembahasan akan menyajikan temuan-temuan dari pengujian sistem, sedangkan kesimpulan dan saran akan merangkum hasil penelitian dan memberikan rekomendasi untuk pengembangan lebih lanjut. Penulisan laporan ini diharapkan dapat memberikan pemahaman yang jelas mengenai proses dan kontribusi penelitian terhadap pengembangan aplikasi deteksi *Malware* berbasis Android.

3.2 Perangkat Penunjang Penelitian

Dalam penelitian ini, peneliti menggunakan perangkat keras dan perangkat lunak tertentu untuk menunjang proses pengembangan dan pengujian sistem. Perangkat keras yang digunakan adalah laptop dengan spesifikasi prosesor Intel Pentium 2020M, RAM sebesar 16 GB, dan penyimpanan internal yang mencukupi untuk pengolahan data dan pengembangan aplikasi. Laptop ini digunakan untuk pengembangan algoritma *Machine Learning* serta pembuatan prototipe aplikasi.

Untuk pengembangan algoritma *Machine Learning*, peneliti memanfaatkan Visual Studio Code sebagai editor kode. Visual Studio Code dipilih karena fleksibilitasnya dalam mendukung berbagai bahasa pemrograman dan integrasi dengan *library Machine Learning* yang relevan.

Dalam pembuatan aplikasi *mobile* berbasis Android, peneliti menggunakan Android Studio sebagai *Integrated Development Environment* (IDE). Android Studio dipilih karena kemampuannya yang lengkap untuk merancang, mengembangkan, dan menguji aplikasi Android dengan berbagai fitur pendukung.

Pengujian aplikasi dilakukan menggunakan *smartphone* dengan sistem operasi Android versi 14, RAM sebesar 8 GB, dan prosesor Snapdragon 685. *Smartphone* ini digunakan untuk memastikan aplikasi dapat berjalan dengan lancar dan mengintegrasikan fungsi deteksi *Malware* dengan algoritma yang telah dikembangkan. Perangkat ini menjadi alat utama dalam menguji fitur aplikasi dan validasi integritas model *Machine Learning* terhadap File yang diterima.