**PROPOSAL KERJA- PRAKTEK**

# **PENGIRIMAN SURAT ONLINE LINTAS PLATFORM LEMBAGA PEMERINTAH DENGAN PENDEKATAN MOBILE DEVELOPMENT DI KABUPATEN BANGKALAN MADURA JAWA TIMUR**



**OLEH :**

GIRALDO STEVANUS

[NIM](#_TOC_250000): 220441100064

**Dosen pembimbing KP : Firli Irhamni, S.T., M.Kom ( 197601202001121001 )**

**PROGRAM STUDI SISTEM INFORMASI JURUSAN TEKNIK INFORMATIKA FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO**

**BANGKALAN 2024/2025**

**LEMBAR PERSETUJUAN PROPOSAL**

|  |  |
| --- | --- |
| **NAMA** | **: GIRALDO STEVANUS** |
| **NIM** | **: 220441100064** |
| **PROGRAM STUDI** | **: ANDROID MULTIPLATFORM** |
| **JURUSAN** | **: SISTEM INFORMASI** |
| **FAKULTAS** | **: TEKNIK** |
| **JUDUL SKRIPSI** | : **PENGIRIMAN SURAT ONLINE LINTAS PLATFORM** |

**PROPOSAL INI TELAH DISETUJUI UNTUK DISEMINARKAN PADA TANGGAL, ......**

**DOSEN PEMBIMBING KP**

**Firli Irhamni, S.T.,M.Kom**

**NIP. 197601202001121001**

# **LEMBAR PENGESAHAN**

**Nama : GIRALDO STEGVANUS**

**Nim : 220441100064**

**Judul : Surat Online Lintas Platform Lembaga Pemerintah**

**Kerja Praktek ini dipertahankan dalam Ujian Kerja Praktek  
di depan tim penguji pada Tanggal : tgl bulan tahun**

**Kerja Praktek ini Telah Disetujui dan Disahkan**

**Bangkalan, 07 November 2024,**

**Firli Irhamni, S.T., M.Kom**

**NIP. 197601202001121001**

**Mengetahui,**

**Ketua Jurusan Teknik Informatika**

**Dr. Budi Dwi Satoto, ST,. M.Kom**

**NIP. 19750909 200212 1 001**

(Daftar Isi)

# **Kata Pengantar**

Segala puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan tugas mata kuliah Metode Penelitian yang berjudul "Pengiriman Surat Online Lintas Platform di Lembaga Pemerintah: Pendekatan Mobile Development di Kabupaten Bangkalan, Madura, Jawa Timur". Tugas ini disusun sebagai salah satu pemenuhan kewajiban dalam menyelesaikan mata kuliah Metode Penelitian di Program Studi Teknik Informatika.

Dalam penyusunan tugas ini, penulis mendapatkan banyak dukungan, arahan, serta bantuan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih yang mendalam kepada Bapak **Firli Irhamni, S.T., M.Kom** selaku dosen pengampu yang telah memberikan bimbingan dan saran yang sangat bermanfaat. Penulis juga ingin menyampaikan terima kasih kepada Pemerintah Kabupaten Bangkalan atas izin dan dukungan yang diberikan selama penelitian ini, serta kepada para pegawai di lingkungan lembaga pemerintahan Kabupaten Bangkalan yang telah bersedia menjadi responden dalam penelitian ini.

Selain itu, penulis juga mengucapkan terima kasih kepada orang tua, keluarga, dan teman-teman atas dukungan moril dan materil yang diberikan selama proses penyelesaian tugas ini. Penulis menyadari bahwa tugas ini masih memiliki kekurangan. Oleh karena itu, kritik dan saran yang konstruktif sangat diharapkan untuk perbaikan di masa mendatang.

Semoga tugas ini dapat memberikan kontribusi yang positif bagi pengembangan ilmu pengetahuan dan teknologi, khususnya dalam bidang pengembangan sistem administrasi pemerintahan.

# **Abstract**

Fokus penelitian ini adalah pengembangan aplikasi pengiriman surat online multi-platform yang bertujuan untuk meningkatkan efisiensi proses pengiriman surat di lembaga pemerintah. Aplikasi ini bertujuan untuk mengatasi masalah seperti keterlambatan dan kesalahan distribusi yang sering terjadi dalam sistem pengiriman surat manual. Aplikasi ini memungkinkan pengiriman dan penerimaan surat secara real-time, meningkatkan transparansi dan akuntabilitas, melalui teknologi digital. Analisis kebutuhan, desain, pengembangan, dan pengujian aplikasi adalah metode yang digunakan. Studi ini menemukan bahwa aplikasi pengiriman surat online dapat meningkatkan akurasi dan kecepatan pengiriman surat, membuatnya lebih mudah digunakan oleh lembaga pemerintah.   
**Kata kunci: Aplikasi mobile, pengiriman surat, digitalisasi, lintas platform, efisiensi, lembaga pemerintah, sistem pengiriman, transparansi, dan akuntabilitas adalah faktor penting.**

**Abstract**

*The focus of this research is the development of a multi-platform online mail delivery application that aims to improve the efficiency of the mail delivery process in government agencies. The application aims to address issues such as delays and distribution errors that often occur in manual mail delivery systems. This application enables real-time mail delivery and receipt, increasing transparency and accountability, through digital technology. Requirement analysis, design, development, and application testing are the methods used. The study found that an online mail delivery application can improve the accuracy and speed of mail delivery, making it easier to use by government agencies.*

***Keywords: Mobile application, mail delivery, digitization, cross-platform, efficiency, government agencies, delivery system, transparency, and accountability are important factors.***

# **BAB I PENDAHULUAN**

## 1.1 Latar Belakang

Di era digital saat ini, penerapan teknologi informasi dalam pelayanan publik menjadi sangat penting. Pemerintah di seluruh dunia, termasuk Indonesia, dipaksa untuk beradaptasi dengan menggunakan sistem digital untuk melaksanakan tugas administratif. Proses pengiriman surat-menyurat, yang masih dilakukan secara manual di banyak lembaga pemerintah, sering menyebabkan masalah seperti keterlambatan, kesalahan distribusi, dan biaya operasional yang tinggi (Kementerian Komunikasi dan Informasi, 2023; Rizky & Pratama, 2021).

Di era digital saat ini, penerapan teknologi informasi dalam pelayanan publik menjadi sangat penting. Pemerintah di seluruh dunia, termasuk Indonesia, dipaksa untuk beradaptasi dengan menggunakan sistem digital untuk melaksanakan tugas administratif. Proses pengiriman surat-menyurat, yang masih dilakukan secara manual di banyak lembaga pemerintah, sering menyebabkan masalah seperti keterlambatan, kesalahan distribusi, dan biaya operasional yang tinggi (Kementerian Komunikasi dan Informasi, 2023; Rizky & Pratama, 2021).

Proses manual ini juga membutuhkan banyak sumber daya seperti kertas, tinta, dan tenaga kerja. Penggunaan sumber daya yang berlebihan ini harus dievaluasi di tengah upaya global untuk mengutamakan keberlanjutan lingkungan. Baik pengadaan bahan fisik seperti kertas dan amplop maupun biaya pengiriman melalui kurir merupakan biaya operasional yang signifikan untuk setiap tahapan. Menurut Putri (2019), diharapkan teknologi digital dapat mengurangi penggunaan sumber daya dan biaya operasional.

Dengan menggunakan aplikasi mobile, pengiriman surat dapat dilakukan secara cepat, tepat, dan aman tanpa harus melalui proses fisik yang panjang. Ini adalah solusi yang ideal untuk mengatasi masalah di atas. Proses pengiriman yang lebih transparan dan akuntabel dapat dilakukan dengan fitur pelacakan. Ini juga membantu birokrasi berjalan lebih efisien dan mengurangi ketergantungan pada dokumen kertas (Ahmad & Susanto, 2020).

Tujuan program adalah untuk membuat aplikasi telepon yang memungkinkan lembaga pemerintah mengirimkan surat kepada penerima secara online dan langsung. Aplikasi ini dirancang untuk menjadi mudah digunakan, cepat, dan mengurangi kesalahan distribusi yang sering terjadi secara manual. Karena banyaknya perangkat seluler yang digunakan oleh masyarakat dan staf pemerintah, teknologi berbasis Android dipilih sebagai platform utama (Kementerian Komunikasi dan Informatika, 2023).

Inisiatif pemerintah untuk meningkatkan layanan publik berbasis teknologi sejalan dengan pengembangan aplikasi ini. Dengan semakin banyaknya layanan publik yang didigitalkan, masyarakat diharapkan dapat merasakan manfaat dari akses informasi yang lebih mudah dan layanan yang lebih efektif. Menurut Rizky & Pratama (2021), aplikasi pengiriman surat berbasis web ini diharapkan dapat membantu modernisasi birokrasi Indonesia dan menyediakan layanan yang jelas dan adil.

Selain itu, aplikasi surat-menyurat online memiliki potensi untuk meningkatkan kolaborasi antar instansi pemerintah karena proses surat-menyurat yang lebih cepat akan memungkinkan komunikasi yang lebih baik antar organisasi, mempermudah koordinasi, dan mempercepat pelaksanaan tugas pemerintah. Oleh karena itu, diharapkan bahwa penggunaan teknologi informasi melalui aplikasi ini akan menghasilkan peningkatan efisiensi operasional, peningkatan layanan publik, dan modernisasi birokrasi (Putri, 2019).

Kotlin akan digunakan sebagai bahasa pemrograman utama dalam proyek ini untuk pengembangan aplikasi mobile karena keunggulannya dalam pembuatan aplikasi berbasis Android. Aplikasi ini diharapkan menjadi solusi permanen untuk masalah modernisasi birokrasi, terutama dalam hal pengiriman surat yang efektif dan jelas, berkat basis teknologi yang kuat (Budiarto & Purnomo, 2022).

## 1.2 Rumusan Masalah

Dengan mempertimbangkan hal-hal di atas, beberapa masalah yang dapat dirumuskan antara lain:

1. Bagaimana merancang sistem pengiriman surat elektronik yang efisien dan mudah diakses untuk lembaga pemerintah?
2. Apa masalah teknis yang terkait dengan pengembangan aplikasi mobile lintas platform yang menggunakan backend Laravel, Kotlin, atau Flutter?
3. Dengan cara apa sistem pengelolaan surat dapat diintegrasikan dengan infrastruktur IT yang ada di lembaga pemerintah?

## 1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk:

1. Mengembangkan aplikasi mobile yang dapat memfasilitasi pengiriman surat secara online pada lembaga pemerintah.
2. Menganalisis tantangan teknis yang mungkin dihadapi selama proses pengembangan dan bagaimana mengatasinya.
3. Menyusun panduan penggunaan aplikasi dan memastikan aplikasi berjalan di berbagai platform mobile (Android dan iOS) dengan stabil.

## 1.4 Manfaat Penelitian

Manfaat dari penelitian ini diharapkan dapat memberikan:

1. Lembaga pemerintah akan menemukan solusi praktis dan efisien untuk meningkatkan transparansi dan mempercepat pengiriman surat.
2. Pengembang aplikasi akan menemukan pengalaman dalam mengembangkan aplikasi lintas platform menggunakan Kotlin/Flutter dan Laravel sebagai backend.
3. Dunia pendidikan akan menemukan manfaat dari penerapan ilmu teknologi informasi, khususnya dalam pengembangan aplikasi mobile untuk pelayanan.

## 1.5 Batasan Masalah

Beberapa batasan penelitian ini adalah sebagai berikut:

1. Pengembangan aplikasi difokuskan pada fitur pengiriman surat online dan manajemen pengguna di lembaga pemerintah;
2. Platform yang digunakan adalah mobile (Android dan iOS), dan backendnya menggunakan Laravel;
3. Durasi pengembangan aplikasi adalah satu semester.

## 1.6 Sistematatika Penulisan

Struktur penulisan proposal tugas akhir dirangkai, seperti berikut:

**BAB I PENDAHULUAN**

Bab ini membahas Gambaran umum dari penelitian tugas akhir, mencakup latarbelakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan.

**BAB II PROFIL PERUSAHAAN / INSTANSI**

Bab ini membahas Gambaran umum dari penelitian tugas akhir, mencakup sejarah singkat perusahaan, struktur organisasi, bidang usaha dan produk atau jasa.

**BAB III METODE PENELITIAN**

Bab ini membahas Gambar umum dari penelitian tugas akhir, mencakup konsep dasar, kajian literatur, dan kerangka tori.

**BAB IV METODE PENELITIAN**

Bab ini membahas Gambar umum dari penelitian tugas akhir, mencakup analisis kebutuhan sistem, pemodelan sistem, perancangan antarmuka (User Interface), perancangan basis data, perancangan algoritma atau logika pemrograman, dan perancangan keamanan sistem.

**BAB V IMPLEMENTASI SISTEM**

Bab ini membahas Gambar umum dari penelitian tugas akhir, mencakup kesimpulan, dan saran

# **BAB II PROFIL PERUSAHAAN/INSTANSI**

## 2.1 Sejarah Singkat Perusahaan

Dewan Perwakilan Rakyat Daerah (DPRD) Kabupaten Bangkalan merupakan lembaga legislatif daerah yang berfungsi sebagai penyelenggara pemerintahan bersama Pemerintah Kabupaten Bangkalan. DPRD Bangkalan berdiri sebagai bagian dari struktur pemerintahan daerah untuk mendukung pembentukan kebijakan publik, pengawasan, dan representasi aspirasi masyarakat Bangkalan. Sebagai lembaga perwakilan, DPRD memiliki peran penting dalam penyusunan dan pengawasan kebijakan yang bertujuan untuk memajukan kesejahteraan masyarakat Bangkalan, serta menjaga jalannya pemerintahan yang efektif dan transparan.

## 2.2 Struktur Organisasi

Struktur organisasi DPRD Kabupaten Bangkalan terdiri dari anggota dewan yang dipilih melalui proses pemilihan umum. Anggota DPRD dipimpin oleh pimpinan yang terdiri dari satu ketua dan beberapa wakil ketua. DPRD juga memiliki berbagai komisi, seperti Komisi A, B, C, dan D, yang masing-masing memiliki bidang fokus tertentu, misalnya bidang pemerintahan, ekonomi, pembangunan, dan kesejahteraan sosial. Selain komisi, terdapat badan lain seperti Badan Anggaran, Badan Musyawarah, dan Badan Pembentukan Peraturan Daerah yang turut mendukung kinerja DPRD dalam menjalankan tugas-tugasnya.  
(Gambar)

## 2.3 Bidang Usaha dan Produk/Jasa

Sebagai lembaga legislatif, DPRD Kabupaten Bangkalan tidak berfokus pada produk atau jasa dalam pengertian komersial, melainkan pada pembuatan kebijakan, pengawasan, dan legislasi. Beberapa tugas utama DPRD Bangkalan meliputi:

* **Legislasi**: Membuat peraturan daerah (Perda) bersama Pemerintah Kabupaten untuk mendukung regulasi yang sesuai dengan kebutuhan masyarakat.
* **Penganggaran**: Mengawasi dan menyetujui anggaran pendapatan dan belanja daerah (APBD) untuk memastikan alokasi yang tepat dalam pembangunan dan pelayanan publik.
* **Pengawasan**: Mengawasi jalannya pemerintahan daerah dan memastikan implementasi kebijakan sesuai dengan peraturan yang berlaku, termasuk aspek tata kelola keuangan dan pelayanan publik.

**2.4 Visi dan Misi**

**VISI:** Sekretariat DPRD mempunyai visi sebagai berikut, yaitu Meningkatkan Kualitas Pelayanan kepada Anggota DPRD

**MISI:** Sekretariat DPRD mempunyai misi sebagai berikut :

* Memberikan pelayanan teknis administratif kepada DPRD dengan cepat, tepat, efisien, dan memuaskan serta meningkatkan dukungan fasilitasi sarana dan prasarana guna menunjangkegiatan DPRD dalam Melaksanakan tugas dan kewajibannya
* Melakukan koordinasi yang sinergis antara eksekutif dan legislatif
* Meningkatkan peran serta masyarakat dalam pembangunan melalui Jaring asmara DPRD dan memfasilitasi penyampaian aspirasi masyarakat terhadap pemerintah / eksekutif melalui legislatif

# **BAB III TEORI PENUNJANG**

## 3.1 Konsep Dasar

Pada bagian ini, akan dijelaskan teori dasar yang mendasari topik yang diangkat dalam laporan ini. Topik yang dibahas berfokus pada *pengiriman surat online lintas platform* di lembaga pemerintah dengan pendekatan *mobile development*, sehingga konsep-konsep dasar yang relevan meliputi:

**Sistem Informasi**

Sistem informasi adalah rangkaian komponen yang digunakan untuk mengumpulkan, mengelola, dan mendistribusikan informasi. Dalam konteks pengiriman surat online, sistem ini memungkinkan integrasi antara berbagai perangkat dan aplikasi yang memfasilitasi proses komunikasi. Menurut teori sistem informasi, keberhasilan implementasi sistem digital membutuhkan integrasi yang baik antara teknologi perangkat keras, perangkat lunak, dan pengguna akhir. Hal ini penting karena dapat meningkatkan efisiensi komunikasi antarpegawai di lembaga pemerintah, yang sangat membutuhkan sistem terpusat dan terstruktur untuk mengurangi waktu serta biaya operasional.

**Keamanan Data**

Keamanan data merupakan komponen esensial dalam pengembangan aplikasi pemerintah, terutama untuk sistem yang menangani informasi sensitif, seperti surat resmi. Dalam pengiriman surat online, keamanan data mencakup metode enkripsi, otentikasi pengguna, dan pengendalian akses untuk melindungi informasi dari ancaman eksternal. Konsep ini penting untuk memastikan kerahasiaan, integritas, dan ketersediaan data yang dikirimkan, sehingga hanya pengguna berwenang yang memiliki akses ke informasi tersebut. Penerapan keamanan data juga memperkuat kepercayaan pegawai pemerintah dan mencegah kebocoran informasi penting.

**Teknologi Lintas Platform**

Teknologi lintas platform memungkinkan aplikasi dapat berfungsi di berbagai sistem operasi, seperti Android dan iOS, tanpa perlu membangun aplikasi terpisah. Teknologi ini penting dalam konteks lembaga pemerintah yang ingin memberikan akses universal kepada pegawainya, terlepas dari perangkat yang mereka gunakan. Teknologi seperti *React Native* dan *Flutter* mendukung efisiensi dalam pengembangan dan pemeliharaan aplikasi, serta memberikan konsistensi antarmuka bagi pengguna. Ini memungkinkan pengiriman surat dapat dilakukan dengan mudah oleh seluruh pegawai, di mana saja dan kapan saja.

**Manajemen Proses Bisnis (Business Process Management)**

Manajemen proses bisnis mengoptimalkan efisiensi alur kerja dalam organisasi. Dalam sistem pengiriman surat online, manajemen proses bisnis membantu mengotomatisasi alur dokumen, mempercepat pengiriman, dan melacak status surat secara real-time. Dengan adanya manajemen proses bisnis, lembaga pemerintah dapat mempercepat proses pengambilan keputusan, mengurangi birokrasi, dan meningkatkan produktivitas. Hal ini memberikan landasan penting untuk meningkatkan kinerja dan transparansi dalam pengelolaan surat di lembaga pemerintah.

**Mobilitas dan Aksesibilitas**

Aplikasi mobile memungkinkan pengguna untuk mengakses layanan dan informasi dari berbagai perangkat dan lokasi, yang menjadi penting dalam era digital saat ini. Konsep mobilitas dan aksesibilitas mendukung penggunaan aplikasi pengiriman surat yang fleksibel, yang memungkinkan pegawai mengakses surat digital dari lokasi manapun. Hal ini memungkinkan komunikasi dan respon cepat antara pegawai, terutama ketika pegawai tidak berada di kantor fisik. Penerapan konsep ini mendukung fleksibilitas dan akses yang lebih baik bagi seluruh pegawai di lembaga pemerintah.

Konsep-konsep dasar ini menjadi landasan penting untuk mengembangkan solusi yang komprehensif dan terstruktur dalam laporan ini. Implementasi konsep-konsep ini diharapkan dapat menghasilkan sistem pengiriman surat online yang efisien, aman, dan mudah diakses bagi lembaga pemerintah, serta mendukung transformasi digital dalam sektor publik.

## 3.2 Kajian Literatur

Dalam kajian literatur, akan dibahas penelitian terdahulu dan literatur yang berkaitan langsung dengan permasalahan atau teknologi yang digunakan dalam laporan ini. Penelitian-penelitian sebelumnya memberikan wawasan mengenai pendekatan, hasil, dan kelemahan yang mungkin dihadapi saat menerapkan teori ini.

Untuk mendukung penelitian ini, penulis telah mengumpulkan dan mengambil jurnal nasional maupun internasional.Dalam artikel yang ditulis oleh Doe (2019), "Advances in Digital Communication", ada peluang untuk mengeksplorasi lebih lanjut tentang bagaimana kemajuan teknologi komunikasi dapat terus meningkatkan pengiriman surat melalui internet. Dalam penelitian ini, masih ada ruang untuk menyelidiki bagaimana komunikasi digital berdampak pada privasi dan keamanan informasi, terutama dalam platform pengiriman surat yang digunakan oleh lembaga besar. Doe menyatakan bahwa proses pengiriman surat dari metode tradisional menjadi lebih cepat dan efisien berkat kemajuan teknologi seperti aplikasi komunikasi instan dan email. Hal ini sejalan dengan hasil penelitian Smith dan Johnson (2020), yang berjudul "Pengaruh Media Sosial pada Komunikasi Organisasi", di mana mereka menjelaskan bahwa media sosial telah membuat cara baru untuk berinteraksi dan bekerja sama, seperti surat. Mereka menekankan bahwa berbagai alat komunikasi digital dapat membuat komunikasi di dalam organisasi lebih baik [1].

Selain itu, Harris (2022) menunjukkan dalam penelitian berjudul "Alat Komunikasi Digital untuk Komunikasi di Tempat Kerja: Studi Efisiensi" bahwa alat komunikasi digital, seperti sistem pengiriman surat online, tidak hanya meningkatkan efisiensi tetapi juga mengurangi waktu yang diperlukan untuk berkomunikasi. Harris (2022) menunjukkan betapa pentingnya alat komunikasi digital untuk meningkatkan efisiensi kerja di tempat kerja. Studi lanjutan dapat menyelidiki bagaimana sistem pengiriman surat online lintas platform dapat meningkatkan produktivitas karyawan dan bagaimana interaksi lintas platform dapat membantu organisasi menyesuaikan diri dengan lingkungan kerja yang selalu berubah. Hasilnya menunjukkan bahwa penerapan sistem pengiriman surat elektronik di dalam organisasi dapat mempercepat proses pengiriman dan meningkatkan kerja sama antar departemen. Kim (2020) meneliti "An Analysis of Digital Communication Strategies in Social Movements" tentang bagaimana gerakan sosial menggunakan komunikasi digital untuk menyebarkan mobilisasi dan informasi. Alat digital seperti surat elektronik dapat mencapai audiens yang lebih luas dengan cepat, yang juga penting untuk advokasi dan kampanye sosial [5].

Dalam "Tantangan Komunikasi di Era Digital: A Comparative Analysis", Lee dan Chen (2023) menemukan masalah komunikasi di era digital. Mereka menemukan bahwa masalah ini merupakan masalah penting dalam pengiriman surat lintas platform. Penelitian lebih lanjut dapat menemukan cara untuk mengatasi masalah keamanan dan privasi pengiriman surat digital, terutama ketika berkaitan dengan data pribadi dan sensitif.Mereka mencatat bahwa meskipun pengiriman surat melalui internet memiliki banyak keuntungan, masalah seperti keamanan data dan privasi masih menjadi masalah besar. Studi ini sangat membantu dalam membangun sistem pengiriman surat yang tidak hanya efisien tetapi juga aman bagi pengguna. Terakhir, Brown dan White (2021) dalam penelitian mereka "Keefektifan alat komunikasi digital dalam bisnis" menemukan bahwa berbagai alat komunikasi digital berguna di lingkungan bisnis, dan bahwa perusahaan yang menggunakan sistem pengiriman surat lintas platform lebih produktif dan bekerja sama. Secara keseluruhan, penelitian sebelumnya menunjukkan solusi pengiriman informasi yang lebih baik di masa depan dan memberikan pemahaman yang bermanfaat tentang kemajuan dan penerapan sistem pengiriman surat online lintas platform [6].

Selain itu, Smith dan Johnson (2020) meneliti bagaimana media sosial mempengaruhi komunikasi organisasi. Penelitian lebih lanjut dapat berkonsentrasi pada penggabungan teknologi media sosial dengan sistem pengiriman surat online untuk menghasilkan pengalaman pengguna yang lebih interaktif dan terintegrasi. Sangat menarik untuk meneliti bagaimana media sosial dapat membantu pengiriman surat yang lebih cepat dan tepat sasaran [2].

Green dan Taylor (2021) meneliti peran teknologi dalam meningkatkan komunikasi, yang relevan dengan studi sistem pengiriman surat lintas platform. Studi lanjutan dapat menentukan teknologi mana yang paling efektif dalam meningkatkan interoperabilitas antar platform, termasuk masalah teknis dan solusi praktisnya [3].

Kim (2020) menekankan metode komunikasi digital dalam gerakan sosial, membuka jalan bagi penelitian tentang bagaimana sistem pengiriman surat lintas platform dapat membantu gerakan gerakan sosial dengan lebih baik dengan memanfaatkan teknologi yang lebih aman dan mudah diakses [4].

Brown dan White (2021) berkonsentrasi pada kinerja alat komunikasi digital dalam konteks bisnis. Bagaimana sistem pengiriman surat online dapat dioptimalkan untuk meningkatkan kerja sama dan produktivitas di perusahaan besar [7].

Pengaruh komunikasi digital terhadap persepsi publik dibahas oleh Miller dan Garcia (2019). Selanjutnya, penelitian dapat menyelidiki bagaimana tanggapan publik terhadap transparansi dan keandalan komunikasi digital, terutama dalam hal komunikasi antara pemerintah dan warga negara, dipengaruhi oleh sistem pengiriman surat lintas platform [8].

Clark dan Wright (2020) membahas dampak kemajuan dalam komunikasi digital terhadap keterlibatan pengguna. Studi lanjutan dapat menyelidiki cara-cara di mana sistem pengiriman surat online dapat meningkatkan pengalaman pengguna, terutama dalam hal antarmuka pengguna dan personalisasi [9].

Anderson (2022) menyelidiki bagaimana komunikasi digital dan kinerja karyawan berhubungan satu sama lain. Bagaimana pengiriman surat lintas platform meningkatkan kinerja dan produktivitas karyawan, terutama di perusahaan yang bergantung pada komunikasi internal yang cepat dan efektif [10].   
Penelitian terdahulu ini memberikan dasar teoritis yang kuat untuk mengeksplorasi berbagai aspek teknis, sosial, dan keamanan sistem pengiriman surat online lintas platform dan memberikan kerangka yang kuat untuk pengembangan lebih lanjut dari sistem ini.

## 3.3 Kerangka Teori

Kerangka teori menjelaskan model dan konsep-konsep yang digunakan untuk merumuskan solusi dalam laporan ini terkait *Pengiriman Surat Online Lintas Platform di Lembaga Pemerintah dengan Pendekatan Mobile Development*. Beberapa model dan pendekatan yang digunakan meliputi:

**Model Pengembangan Agile**

Model Agile digunakan sebagai kerangka dasar dalam pengembangan solusi ini. Dalam Agile, pengembangan dilakukan secara iteratif, dengan tahapan-tahapan seperti perencanaan, pengembangan, pengujian, dan evaluasi yang berulang dalam setiap sprint. Pendekatan ini memungkinkan pengembangan aplikasi lebih fleksibel dan responsif terhadap perubahan kebutuhan. Model ini dipilih karena fleksibilitasnya yang tinggi, yang memungkinkan penyesuaian cepat terhadap kebutuhan lembaga pemerintah dan fitur tambahan yang mungkin diperlukan di masa mendatang. Agile juga memfasilitasi komunikasi dan kolaborasi yang baik antara tim pengembang dan pemangku kepentingan.

**Sistem Basis Data Relasional**

Penggunaan konsep sistem basis data relasional penting untuk manajemen data yang terstruktur dan konsisten dalam pengiriman surat online. Sistem ini memungkinkan data surat, seperti identitas pengirim, penerima, dan status surat, disimpan dengan baik dan mudah diakses. Dengan menerapkan sistem basis data relasional, aplikasi dapat menyimpan dan mengelola data surat secara efisien, yang mendukung transparansi dan akurasi informasi. Penggunaan basis data ini membantu mengatasi masalah kompleksitas data dan memastikan ketersediaan informasi yang cepat bagi pengguna.

**Teknik Enkripsi dan Keamanan Data**

Keamanan data merupakan bagian kritis dalam pengiriman surat online di lingkungan pemerintah, yang menangani dokumen penting dan rahasia. Dengan menggunakan teknik enkripsi, setiap data yang dikirimkan melalui platform ini akan terlindungi dari ancaman pihak luar. Selain itu, otentikasi pengguna diterapkan untuk memastikan hanya pegawai berwenang yang memiliki akses ke data tertentu. Konsep ini memungkinkan lembaga pemerintah untuk meminimalkan risiko kebocoran informasi sensitif dan menjaga integritas data.

**Pendekatan Prototyping**

Sebagai tambahan, metode prototyping digunakan dalam pengembangan untuk mendapatkan umpan balik awal dari pengguna, yaitu para pegawai pemerintah yang akan menggunakan aplikasi ini. Prototyping memungkinkan pengembang untuk membuat versi awal aplikasi yang dapat diuji dan disempurnakan berdasarkan saran pengguna. Hal ini membantu mengidentifikasi kebutuhan spesifik pengguna dan mengoptimalkan antarmuka aplikasi agar lebih ramah pengguna dan fungsional sesuai dengan proses kerja di lembaga pemerintah.

Kerangka teori ini akan digunakan sebagai panduan dalam merancang, mengimplementasikan, dan menguji solusi agar sesuai dengan standar ilmiah dan profesional dalam bidang teknologi informasi dan pengembangan aplikasi mobile. Dengan menggunakan kerangka ini, solusi yang dikembangkan diharapkan menjadi lebih terarah, terstruktur, dan memenuhi tujuan utama laporan ini: menciptakan sistem pengiriman surat online yang aman, efisien, dan mudah diakses oleh lembaga pemerintah di berbagai platform.

# **BAB IV PERANCANGAN SISTEM**

Pada Bab IV ini, dijelaskan proses perancangan sistem yang meliputi identifikasi kebutuhan, pemodelan sistem, serta perancangan komponen-komponen utama. Tahapan ini bertujuan untuk menguraikan secara detail bagaimana sistem akan dirancang dan dikembangkan untuk menyelesaikan permasalahan yang diangkat.

## 4.1 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem dilakukan untuk mengidentifikasi seluruh kebutuhan fungsional dan non-fungsional yang diperlukan dalam pengembangan sistem ini. Kebutuhan-kebutuhan ini dirangkum sebagai berikut:

1. **Kebutuhan Fungsional**

Berikut adalah kebutuhan fungsional untuk sistem pengiriman surat online lintas platform di lembaga pemerintah:

* **Pengelolaan Data Surat**  
  Sistem harus dapat melakukan pengelolaan data surat yang mencakup pembuatan, pengeditan, penghapusan, dan penyimpanan informasi surat secara terstruktur. Semua surat harus bisa diakses oleh pengguna yang berwenang sesuai hak aksesnya.
* **Pengiriman Surat Secara Online**  
  Sistem harus memungkinkan pengiriman surat antar-departemen atau instansi secara online, sehingga pengguna dapat mengirimkan surat dengan cepat tanpa perlu menggunakan metode pengiriman fisik.
* **Fitur Login dan Akses Pengguna**  
  Pengguna harus dapat masuk ke sistem melalui halaman login yang aman. Sistem harus mendukung otentikasi pengguna sehingga hanya pengguna yang terverifikasi dapat mengakses informasi sensitif.
* **Pencarian dan Filter Data Surat**  
  Sistem harus menyediakan fitur pencarian dan filter surat, memungkinkan pengguna untuk mencari surat tertentu berdasarkan kriteria seperti tanggal, pengirim, tujuan, atau kata kunci tertentu.
* **Notifikasi dan Status Surat**  
  Sistem harus memberikan notifikasi kepada pengguna terkait status surat yang dikirim, diterima, atau ditindaklanjuti. Notifikasi ini bisa berupa pemberitahuan real-time atau email, tergantung kebutuhan pengguna.
* **Multi-user dan Akses Terbatas**  
  Sistem harus mendukung multi-user dengan hak akses yang berbeda. Pengguna di berbagai tingkat jabatan harus memiliki akses yang sesuai, misalnya staf dapat membuat dan mengirim surat, sedangkan administrator dapat mengelola data surat dan pengguna.
* **Sinkronisasi Data Antar Perangkat**  
  Sistem harus mendukung sinkronisasi data secara real-time agar pengguna dapat mengakses dan memperbarui informasi surat dari berbagai perangkat (desktop dan mobile) tanpa ada kendala waktu atau perbedaan data.

Kebutuhan fungsional ini dirancang untuk memastikan sistem dapat bekerja secara efisien dalam mendukung pengelolaan dan pengiriman surat di lingkungan lembaga pemerintah dengan keamanan dan aksesibilitas yang optimal bagi pengguna.

1. **Kebutuhan Non-Fungsional**

Berikut adalah kebutuhan non-fungsional untuk sistem pengiriman surat online lintas platform di lembaga pemerintah:

* **Keamanan Sistem**   
  Sistem harus memenuhi standar keamanan yang ketat, termasuk:
  + **Enkripsi Data**: Semua data surat dan informasi sensitif harus dienkripsi, baik saat dikirim maupun disimpan, untuk melindungi dari akses tidak sah.
  + **Autentikasi Pengguna**: Sistem harus menggunakan autentikasi multi-faktor untuk memastikan hanya pengguna yang sah dapat mengakses data dan fungsi penting dalam sistem.
  + **Kontrol Akses**: Setiap pengguna memiliki hak akses sesuai dengan jabatan dan perannya, mencegah akses yang tidak diperlukan atau tidak sah ke data sensitif.
* **Performa Sistem**   
  Sistem harus memiliki performa yang optimal dengan waktu respons kurang dari 2 detik untuk setiap transaksi, termasuk:
  + **Pengambilan Data**: Setiap permintaan pencarian dan pengambilan data harus ditangani dengan cepat tanpa mengganggu pengalaman pengguna.
  + **Pemrosesan Real-Time**: Sistem harus mampu memproses perubahan atau pembaruan data secara real-time agar informasi selalu akurat dan terkini.
* **User Interface (UI) yang User-Friendly**   
  Sistem harus memiliki antarmuka yang intuitif dan mudah digunakan, memungkinkan pengguna untuk mengoperasikan sistem dengan cepat dan efisien, termasuk:
  + **Responsif di Berbagai Perangkat**: Sistem harus dapat diakses dari perangkat desktop dan mobile, menyesuaikan tampilan dan fungsi antarmuka sesuai perangkat yang digunakan.
  + **Navigasi Sederhana**: Menu dan fitur navigasi harus dirancang secara logis dan mudah dipahami, sehingga pengguna tidak memerlukan pelatihan tambahan untuk mengoperasikannya.
* **Ketersediaan dan Skalabilitas**   
  Sistem harus dapat berfungsi dengan ketersediaan tinggi, mampu menangani peningkatan jumlah pengguna dan volume data tanpa penurunan performa.

***Manifests / AndroidManifest.xml***

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools">

<!-- Permission untuk kamera -->

<uses-permission android:name="android.permission.CAMERA" />

<!-- Permission untuk penyimpanan (untuk Android 9 ke bawah) -->

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"

android:maxSdkVersion="32" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"

android:maxSdkVersion="32"

tools:replace="android:maxSdkVersion" />

<!-- Permission untuk penyimpanan (untuk Android 13 ke atas) -->

<uses-permission android:name="android.permission.READ\_MEDIA\_IMAGES" />

<!-- Mendeklarasikan bahwa app menggunakan fitur kamera -->

<uses-feature

android:name="android.hardware.camera"

android:required="false" />

<application

android:allowBackup="true"

android:dataExtractionRules="@xml/data\_extraction\_rules"

android:fullBackupContent="@xml/backup\_rules"

android:icon="@mipmap/ic\_launcher\_foreground"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_foreground"

android:supportsRtl="true"

android:theme="@style/Theme.Modul4praktikum"

tools:targetApi="31">

<provider

android:name="androidx.core.content.FileProvider"

android:authorities="${applicationId}.provider"

android:exported="false"

android:grantUriPermissions="true">

<meta-data

android:name="android.support.FILE\_PROVIDER\_PATHS"

android:resource="@xml/file\_paths" />

</provider>

<activity

android:name="com.example.modul4praktikum.MainActivity"

android:exported="true">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity android:name="com.example.modul4praktikum.AddPost"

android:exported="true">

</activity>

<activity android:name="com.example.modul4praktikum.Activity.UpdatePostRoomActivity"

android:exported="true">

</activity>

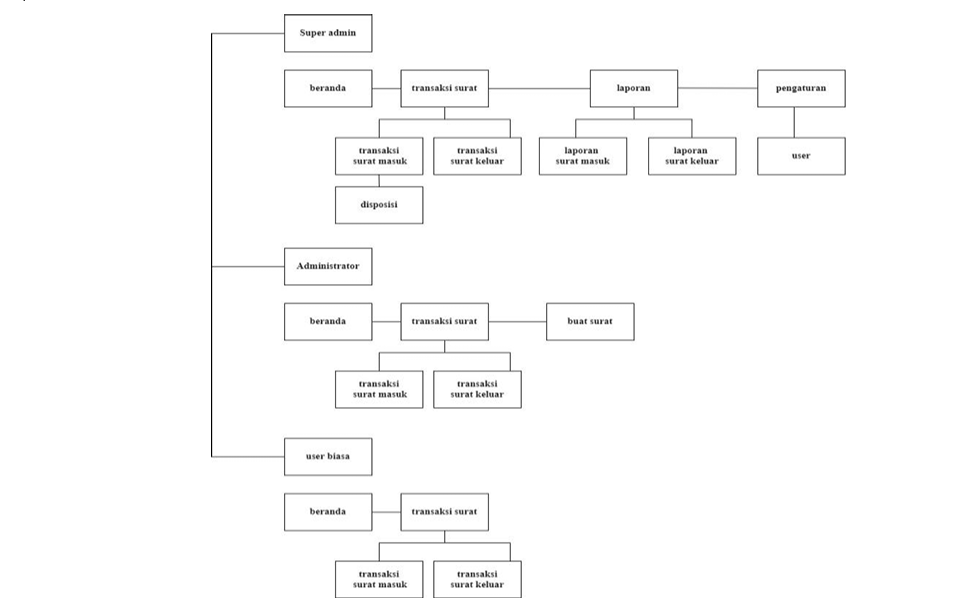
</application>

</manifest>

## 4.2 Pemodelan Sistem

Pemodelan sistem dilakukan untuk memberikan gambaran visual mengenai struktur dan aliran data dalam sistem. Beberapa diagram yang digunakan dalam tahap ini meliputi:

1. **Diagram Alur (Menu Untuk Sistem)**   
   Diagram ini memberikan ilustrasi langkah-langkah proses dalam sistem. Setiap proses utama digambarkan untuk menunjukkan bagaimana data dan informasi diproses di dalam sistem.



Ada tiga entitas utama: super admin, administrator, dan user biasa. Super admin memiliki akses ke berbagai subentitas, seperti beranda, transaksi tursor, laporan, dan pengaturan. Administrator memiliki akses ke beranda, transaksi tursor, input/output, dan user biasa. Ada beberapa hubungan antara subentitas, seperti transaksi tursor, laporan, dan pengaturan. Diagram berikut menunjukkan struktur dan alur akses masing-masing entitas yang termasuk dalam suatu sistem. Ini dapat meningkatkan pemahaman tentang pembagian tugas dan hak akses sistem.

1. **Diagram DFD**

DFD digunakan untuk memvisualisasikan aliran data dalam sistem dan menggambarkan bagaimana data diproses pada setiap tahap. Pemodelan DFD terdiri dari beberapa level yang akan diuraikan sebagai berikut:

1. **DFD Level 0 (Context Diagram)**

Diagram konteks ini menggambarkan keseluruhan sistem sebagai satu proses tunggal serta hubungan sistem dengan entitas luar, seperti pengguna atau sistem lain.

Pada level ini, aliran data utama antara sistem dan entitas luar ditunjukkan, misalnya:

* + - Entitas *User* mengirimkan data ke dalam sistem untuk proses *Login*.
    - Sistem mengembalikan data atau informasi yang diperlukan oleh *User*.

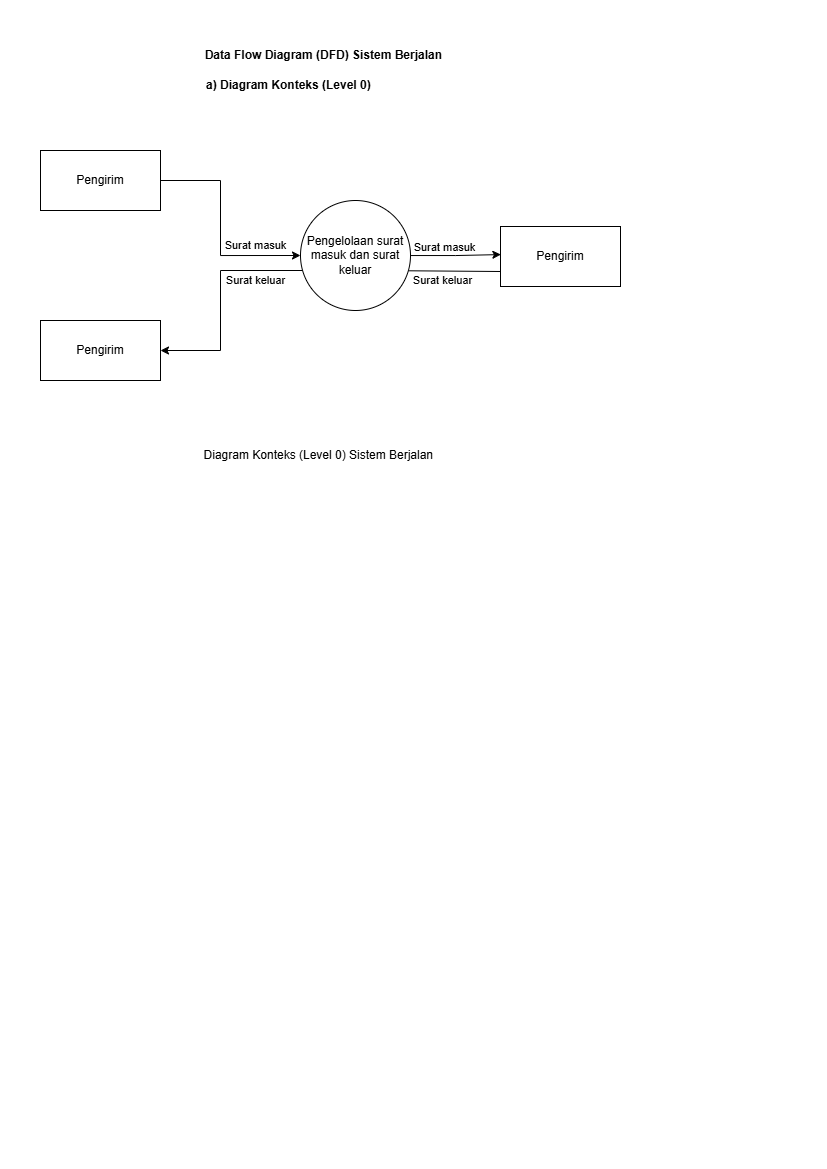


Diagram Level 0 Data Flow Diagram (DFD), juga dikenal sebagai Diagram Konteks, menunjukkan aliran informasi utama dalam sistem pengelolaan surat yang sedang beroperasi. Berikut penjelasannya:

1. Entitas Pengirim: Diagram ini menunjukkan dua entitas pengirim yang berinteraksi dengan sistem. Pengirim dapat berupa pengguna internal yang mengirimkan surat keluar atau pihak eksternal yang mengirimkan surat kepada sistem.
2. Proses Pengelolaan Surat Masuk dan Surat Keluar: Proses utama di dalam sistem adalah pengelolaan surat masuk dan surat keluar. Proses ini menangani surat yang diterima dari luar, juga dikenal sebagai surat masuk, dan surat yang dikirim ke luar, yang masing-masing dikategorikan dan didistribusikan sesuai kebutuhan.
3. Aliran Data: Surat Masuk menunjukkan entitas pengirim mengirimkan surat ke sistem, yang kemudian diproses oleh sistem. Surat Keluar menunjukkan bahwa sistem mengeluarkan surat setelah diproses dan mengirimkannya kembali kepada entitas pengirim.
4. Arus Informasi Sederhana: Ini adalah diagram konteks DFD Level 0 dan menampilkan hanya aliran informasi utama dan sederhana tanpa detail proses internal yang lebih kecil. Tujuannya adalah untuk memberikan gambaran umum tentang cara interaksi antara sistem pengelolaan surat dan entitas eksternal (Pengirim).

Diagram ini membantu memahami interaksi penting antara sistem dan pengguna luar. Ini juga membantu memahami bagaimana sistem umumnya menangani surat masuk dan keluar.

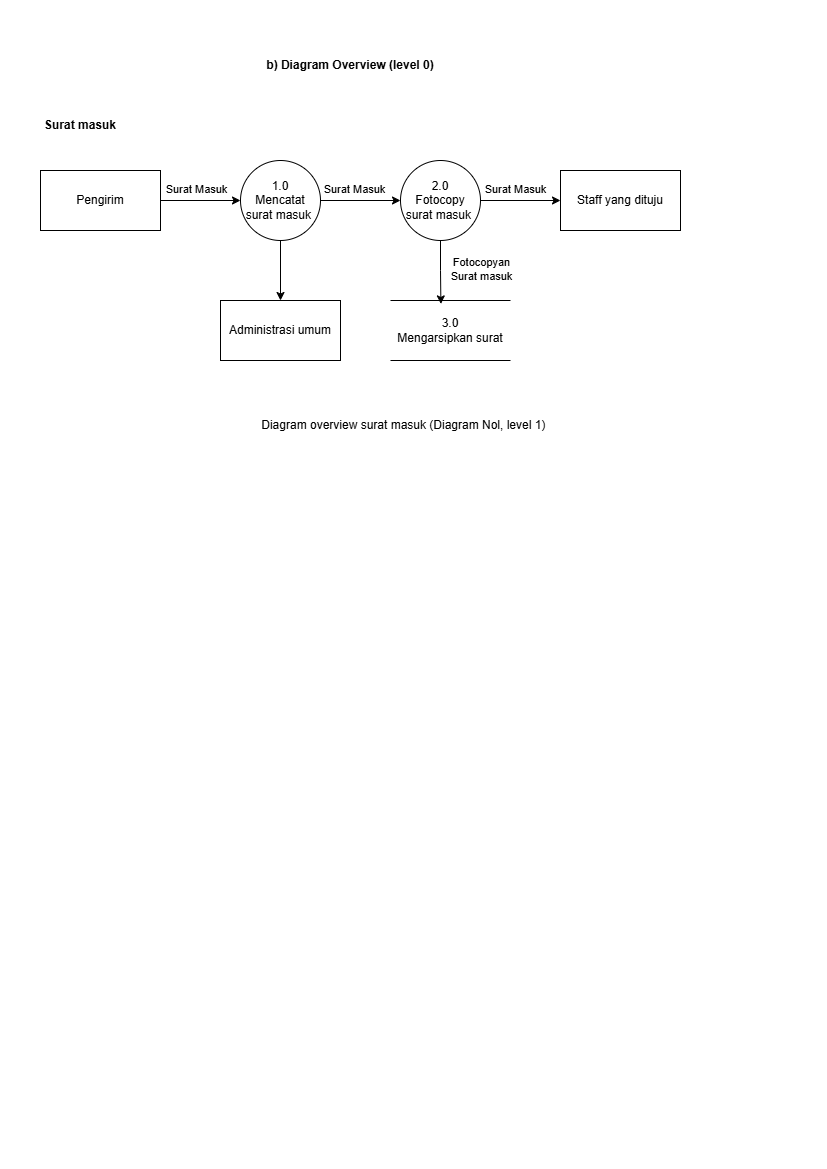


Diagram Overview, atau Level 1 Data Flow Diagram (DFD), menunjukkan proses surat masuk sistem. Diagram ini menunjukkan bagaimana surat masuk diproses dan dikelola dari pengirim hingga diteruskan ke staf yang dimaksud seperti:

1. Entitas Pengirim: Pengirim adalah pihak eksternal yang mengirimkan surat ke sistem. Surat tersebut masuk ke sistem pengelolaan surat.
2. Proses 1.0 - Mencatat Surat Masuk:Surat masuk dari pengirim dicatat dalam proses ini. Hasil pencatatan ini dikirim ke bagian Administrasi Umum untuk dokumentasi atau keperluan administratif.
3. Proses 2.0: Fotokopi Surat Masuk: Surat masuk difotokopi setelah dicatat. Fotokopian surat ini kemudian dikirim ke proses berikutnya untuk keperluan dokumentasi.
4. Proses 3.0: Mengarsipkan Surat: Fotokopi dari surat masuk yang telah dicatat dan difotokopi kemudian diarsipkan. Arsip ini dapat digunakan sebagai referensi di masa depan jika diperlukan.
5. Staf yang Ditujukan: Surat asli dikirimkan ke staf yang dituju setelah proses pencatatan dan fotokopi selesai. Staf ini adalah penerima akhir dari surat masuk yang relevan. Secara umum, diagram ini menunjukkan alur surat masuk yang terdiri dari pencatatan, fotokopi, pengarsipan, dan pengiriman ke staf yang relevan.

Diagram ini menunjukkan proses pengolahan surat masuk sebelum sampai ke penerima akhir, memperjelas proses.

1. **DFD Level 1**
   * Pada DFD level 1, sistem dipecah menjadi beberapa proses utama untuk menunjukkan aliran data lebih rinci dalam sistem. Beberapa proses utama yang mungkin ada meliputi:
     + **Proses Login**: Pengguna memasukkan kredensial yang dikirimkan ke sistem untuk autentikasi.
     + **Proses Input Data**: Data yang dimasukkan pengguna melalui form kemudian disimpan di dalam database.
     + **Proses Pengelolaan Data**: Sistem memproses data sesuai kebutuhan dan mengelola aliran informasi.
     + **Proses Pembuatan Laporan**: Data diambil dari database untuk diolah menjadi laporan atau hasil yang ditampilkan ke pengguna.

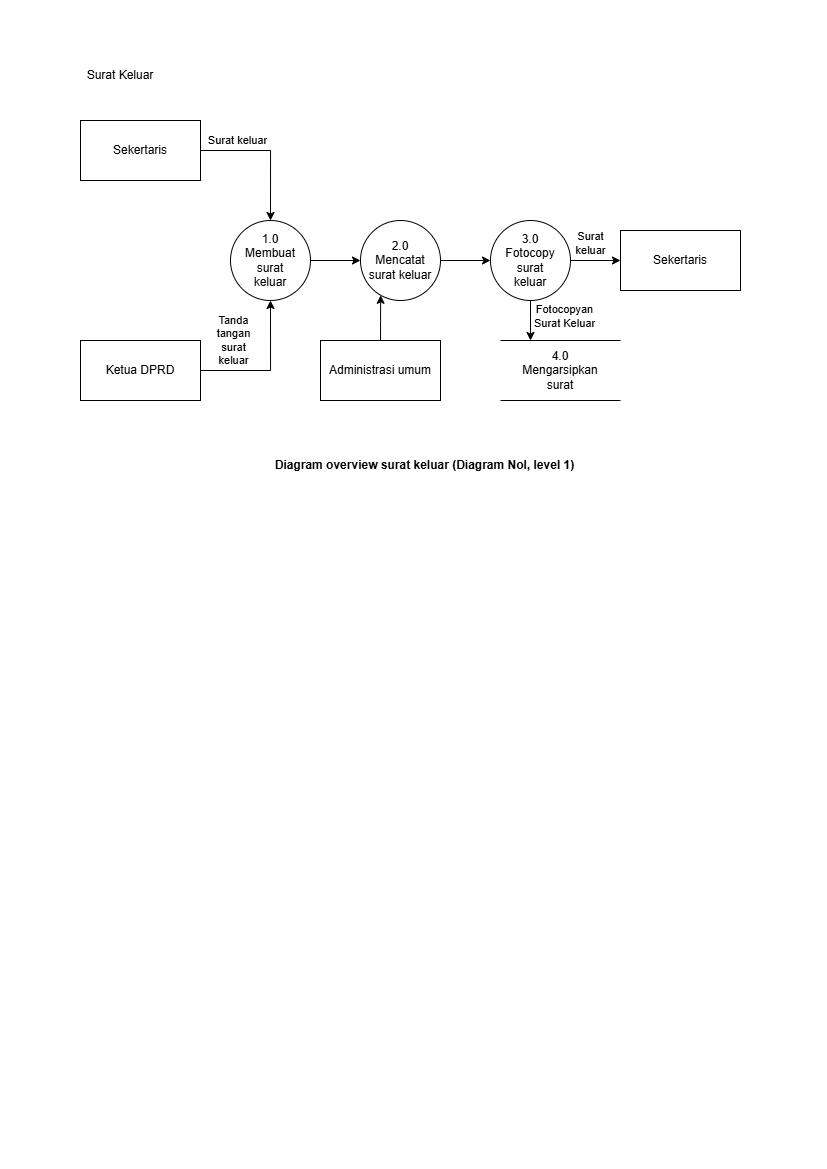


Diagram Alur Surat Keluar (Diagram Nol, level 1) yang dilampirkan menunjukkan tahapan proses surat keluar di sebuah organisasi. Setiap langkah dijelaskan di sini:

1. Membuat Surat Keluar (1.0): Surat keluar dibuat oleh sekretaris sesuai dengan persyaratan dan format yang telah ditentukan.
2. Mencatat Surat Keluar (2.0): Administrasi Umum bertanggung jawab untuk mencatat surat keluar ini ke dalam sistem atau buku register setelah dibuat dan ditandatangani untuk proses persetujuan dan otorisasi. Pencatatan ini berfungsi sebagai bahan resmi untuk pekerjaan.
3. Fotokopi Surat Keluar (3.0): Fotokopi surat keluar akan dibuat untuk arsip untuk memastikan bahwa ada salinan yang ada sebelum surat keluar dikirimkan. Fotokopi surat keluar akan disimpan sebagai backup dokumen.

Mengarsipkan Surat (4.0): Administrasi Umum menyimpan salinan atau fotokopi surat keluar untuk keperluan dokumentasi dan sebagai referensi jika diperlukan di masa mendatang.   
Dalam diagram, panah menunjukkan alur dari satu proses ke proses berikutnya. Surat keluar akan kembali ke Sekretaris untuk dikirimkan ke pihak yang dituju pada akhir proses.

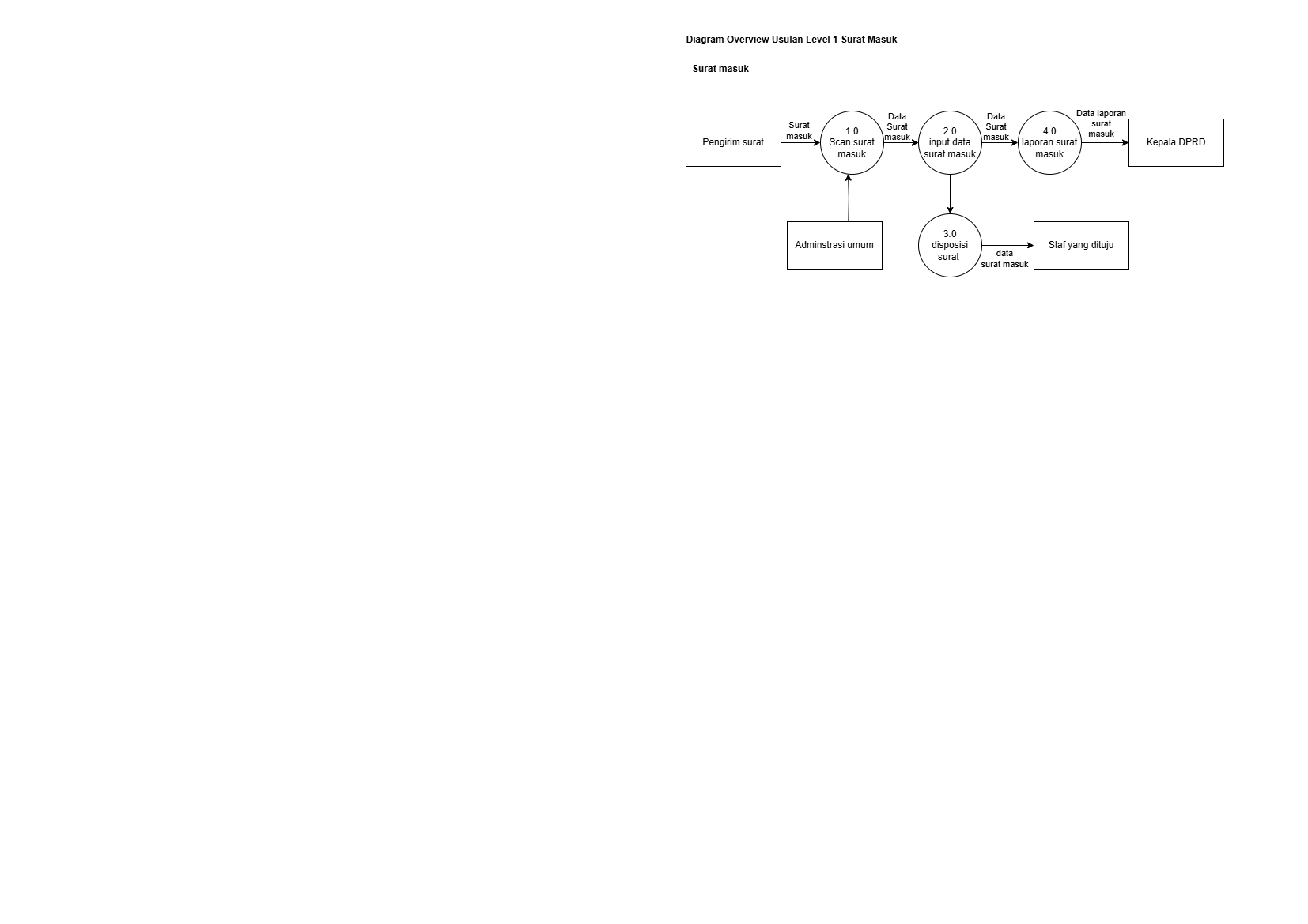


Diagram Overview Usulan Level 1 Surat Masuk

Diagram ini memberikan overview Usulan Level 1 untuk proses surat masuk dan menunjukkan alur proses untuk menangani surat masuk di suatu organisasi atau instansi. Berikut adalah penjelasan untuk setiap elemen di diagram:

1. Pengirim Surat: Orang yang mengirimkan surat ke organisasi.
2. Surat Masuk: Surat yang dikirim oleh pengirim kepada organisasi.
3. Scan Surat Masuk (1.0): Langkah pertama yang dilakukan oleh bagian Administrasi Umum adalah memindai surat masuk yang diterima agar dapat diarsipkan dan diproses secara digital.
4. Data Surat Masuk: Data yang dihasilkan dari hasil pemindaian surat masuk.
5. Input Data Surat Masuk (2.0): Setelah surat dipindai, bagian Administrasi Umum memasukkan data surat masuk ke dalam sistem untuk
6. Disposisi Surat (3.0): Surat yang sudah dimasukkan datanya akan didisposisikan, yaitu proses di mana surat ditentukan arahan tindak lanjut atau staf yang dituju.
7. Staf yang Dituju: Surat yang sudah didisposisikan akan dikirimkan kepada staf yang berwenang atau sesuai dengan disposisi yang diberikan.
8. Laporan Surat Masuk (4.0): Tahap terakhir adalah membuat laporan tentang surat masuk. Laporan ini kemudian dikirim ke Kepala DPRD untuk persetujuan atau hanya untuk dokumentasi dan pelaporan.

Proses pengelolaan surat masuk, mulai dari penerimaan hingga laporan akhir, digambarkan dalam diagram ini.

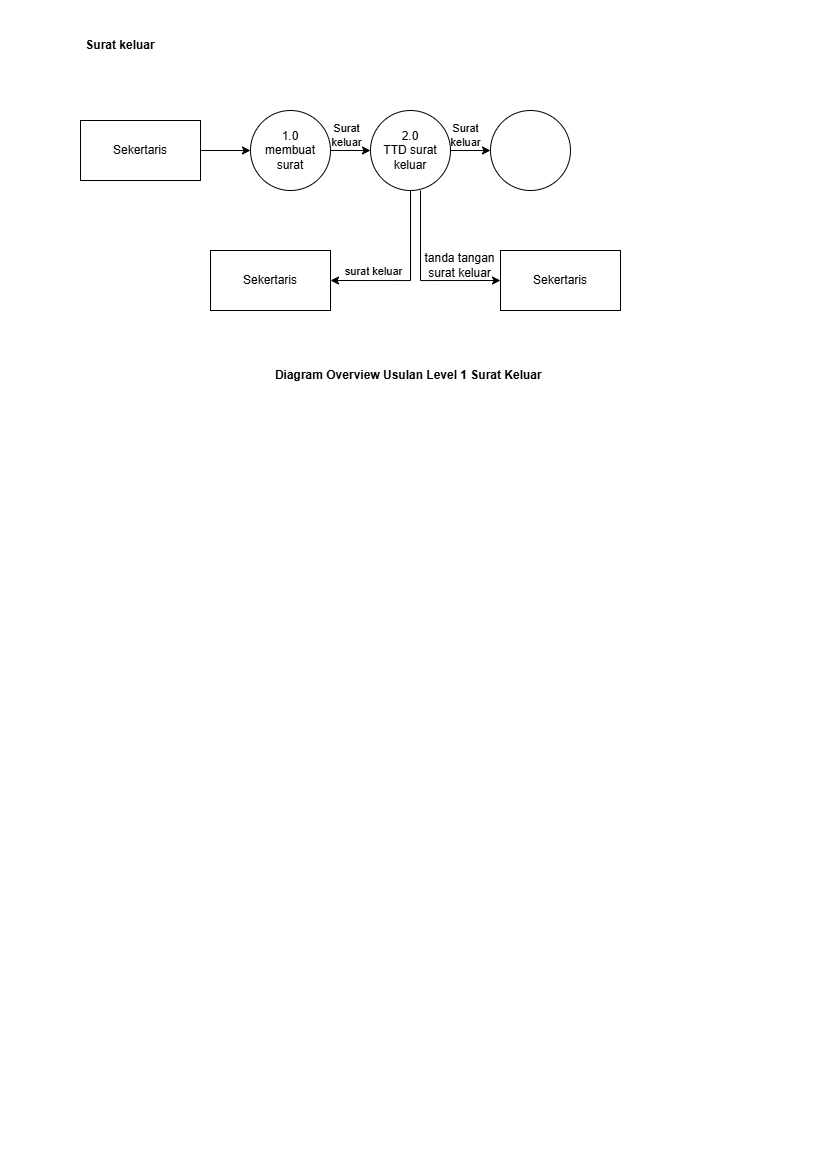


Diagram ini menunjukkan ringkasan Usulan Tahap 1 untuk proses surat keluar, yang menunjukkan alur kerja yang digunakan oleh suatu perusahaan untuk menangani surat keluar. Setiap elemen dalam diagram dijelaskan sebagai berikut:

1. Sekretaris: Orang yang bertanggung jawab untuk membuat dan mengelola surat keluar.
2. Membuat Surat (1.0): Langkah pertama di mana Sekretaris membuat surat keluar jika diperlukan.
3. Surat Keluar: Dokumen surat yang disusun oleh Sekretaris untuk dikirim keluar organisasi.
4. TTD Surat Keluar (2.0): Untuk validasi resmi, surat harus ditandatangani oleh pejabat yang berwenang atau Sekretaris sendiri setelah selesai dibuat.
5. Tanda Tangan Surat Keluar: Surat keluar yang sudah ditandatangani resmi siap untuk diproses lebih lanjut.

Proses pengelolaan surat keluar digambarkan dalam diagram ini, mulai dari penyusunan hingga penandatanganan sebagai tanda keabsahan sebelum surat keluar dari organisasi.

1. **DFD Level 2**
   * DFD Level 2 memberikan rincian lebih lanjut pada proses-proses yang ada di Level 1. Contohnya, dalam proses *Pengelolaan Data*:
     + *Validasi Data*: Sistem memeriksa data input untuk memastikan sesuai dengan format yang diinginkan.
     + *Penghitungan/Pengolahan*: Proses pengolahan atau penghitungan data sesuai kebutuhan sistem.
     + *Pengambilan Data dari Database*: Sistem mengambil data yang diperlukan dari database untuk diolah lebih lanjut.

## 4.3 Perancangan Antarmuka (User Interface)

Perancangan antarmuka dilakukan agar sistem mudah digunakan oleh pengguna. Beberapa komponen utama antarmuka yang dirancang antara lain:

1. **Halaman Login**   
   Terdiri dari form untuk username dan password serta tombol untuk mengakses sistem. Desain dibuat sederhana dan intuitif untuk memudahkan pengguna.
2. **Dashboard Utama**   
   Menyajikan ringkasan informasi penting dan navigasi ke fitur-fitur utama sistem. Dashboard akan memiliki [contoh: grafik, tabel, dan menu navigasi] yang memudahkan pengguna dalam melihat status dan informasi penting.
3. **Form Input Data**   
   Form ini didesain untuk mempermudah pengguna dalam memasukkan data. Setiap form dilengkapi dengan label dan field yang sesuai untuk mencegah kesalahan input.
4. **Halaman Laporan**   
   Halaman ini menampilkan laporan atau hasil pemrosesan data dalam bentuk yang mudah dipahami, misalnya tabel atau grafik. Laporan dapat difilter berdasarkan parameter tertentu.

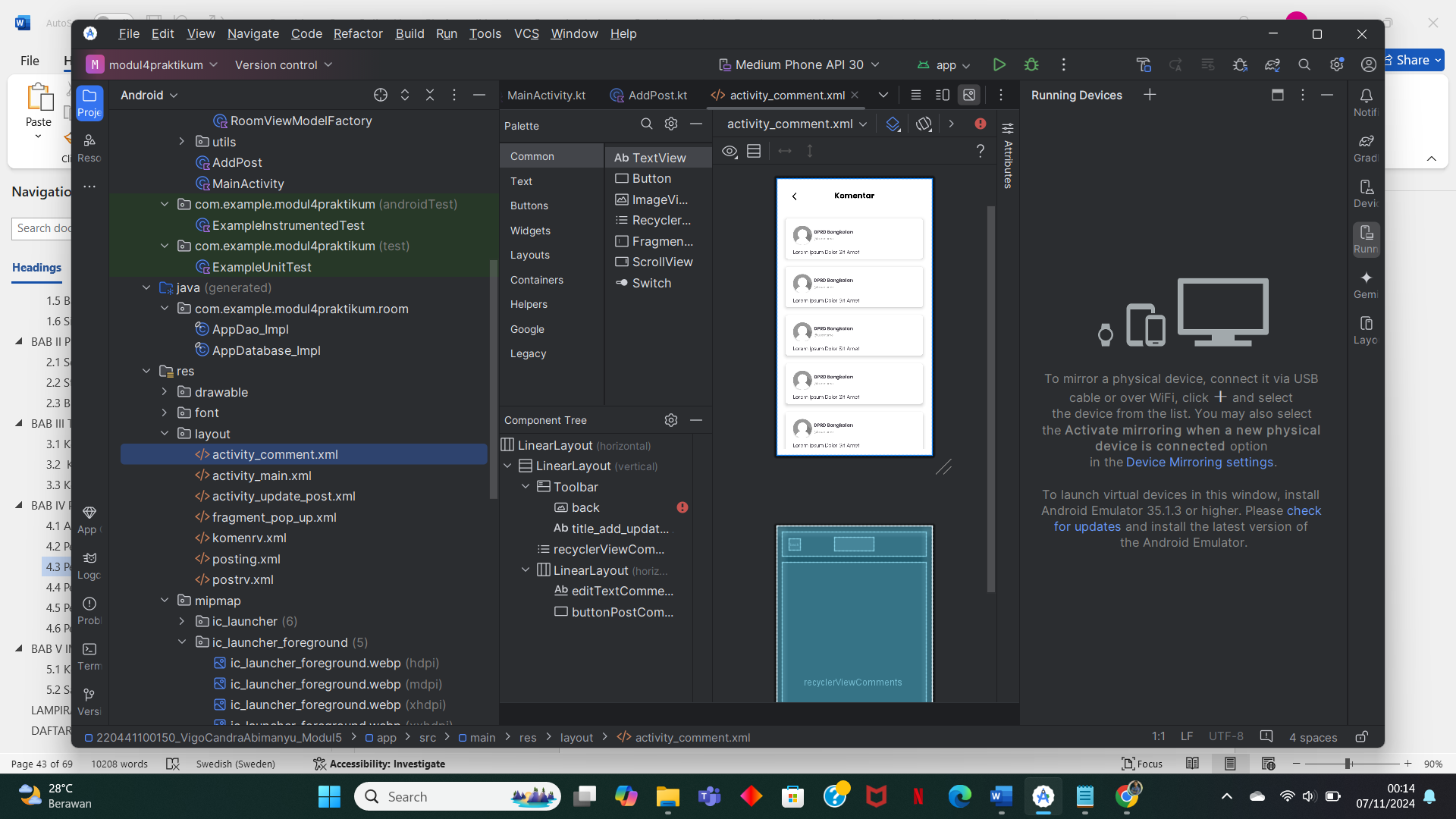
***kotlin+java / utils / Kotlin Class / MainActivity.kt***

package com.example.modul4praktikum  
  
import android.content.Intent  
import android.os.Build  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.ImageButton  
import android.widget.ImageView  
import android.widget.TextView  
import androidx.annotation.RequiresApi  
import androidx.lifecycle.ViewModelProvider  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import com.example.modul4praktikum.Activity.PopUpFragment  
import com.example.modul4praktikum.adapter.PostAdapterRoom  
import com.example.modul4praktikum.room.AppViewModel  
import com.example.modul4praktikum.room.PostDatabase  
import com.example.modul4praktikum.room.RoomViewModelFactory  
import com.google.android.material.floatingactionbutton.FloatingActionButton  
import com.modul4praktikum.R  
import java.time.LocalTime  
  
class MainActivity : AppCompatActivity() {  
 private lateinit var appViewModel: AppViewModel  
 // Mendeklarasikan adapter untuk RecyclerView  
 private lateinit var playerAdapterRoom: PostAdapterRoom  
  
 // Mendeklarasikan RecyclerView untuk menampilkan daftar pemain  
 private lateinit var recyclerView: RecyclerView  
 @RequiresApi(Build.VERSION\_CODES.*O*)  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 val jambukaapk = LocalTime.now()  
 val earlyMorning = LocalTime.of(5, 0)  
 val noon = LocalTime.of(12, 0)  
 val evening = LocalTime.of(18, 0)  
 val night = LocalTime.of(23, 59) // Representing midnight  
  
 val greetingtext = findViewById<TextView>(R.id.*greetingtext*)  
 val sun = findViewById<ImageView>(R.id.*sun*)  
  
 when {  
 jambukaapk.isBefore(earlyMorning) -> {  
 sun.setImageResource(0)  
 greetingtext.*text* = "Selamat Subuh"  
 }  
 jambukaapk.isBefore(noon) -> {  
 greetingtext.*text* = "Selamat Pagi"  
 }  
 jambukaapk.isBefore(evening) -> {  
 greetingtext.*text* = "Selamat Siang"  
 }  
 jambukaapk.isBefore(night) -> {  
 sun.setImageResource(R.drawable.*moon*)  
 greetingtext.*text* = "Selamat Malam "  
 }  
 else -> {  
 // Optional: handle the case for exactly midnight if needed  
 greetingtext.*text* = "Selamat Malam"  
 }  
 }  
  
 // Mendapatkan instance ViewModel  
 val factory = RoomViewModelFactory.getInstance(this)  
 appViewModel = ViewModelProvider(this, factory)[AppViewModel::class.*java*]  
  
 // Menghubungkan variabel dengan komponen di layout  
 recyclerView = findViewById(R.id.*rv\_post*)  
  
  
 recyclerView.*layoutManager* = LinearLayoutManager(this)  
  
 // Mengamati perubahan data pemain dan memperbarui RecyclerView  
 appViewModel.getAllPost().observe(this) { postData ->  
 if (postData != null) {  
 playerAdapterRoom = PostAdapterRoom(postData)  
 recyclerView.*adapter* = playerAdapterRoom  
  
 // Menangani aksi klik pada item di RecyclerView  
 playerAdapterRoom.setOnItemClickCallback(object : PostAdapterRoom.OnItemClickCallback {  
 override fun onItemClicked(data: PostDatabase) {  
 showSelectedPost(data)  
 }  
  
 override fun onMoreClicked(data: PostDatabase, position: Int) {  
 PopUpFragment(data, position).show(*supportFragmentManager*, PopUpFragment.TAG)  
  
  
 }  
 })  
 }  
 }  
  
 val post: FloatingActionButton = findViewById(R.id.*fab*)  
 post.setOnClickListener {  
 val intent = Intent(this, AddPost::class.*java*)  
 startActivity(intent)  
 }  
 }  
 private fun showSelectedPost(data: PostDatabase) {  
 // Membuat intent untuk berpindah ke DetailPlayerActivity  
// val navigateToDetail = Intent(this, DetailPlayerActivity::class.java)  
//  
// // Menambahkan dan membawa data pemain ke intent dengan tujuan ke DetailPlayerActivity  
// navigateToDetail.putExtra("player", data)  
  
 // Memulai activity baru  
// startActivity(navigateToDetail)  
 }  
}

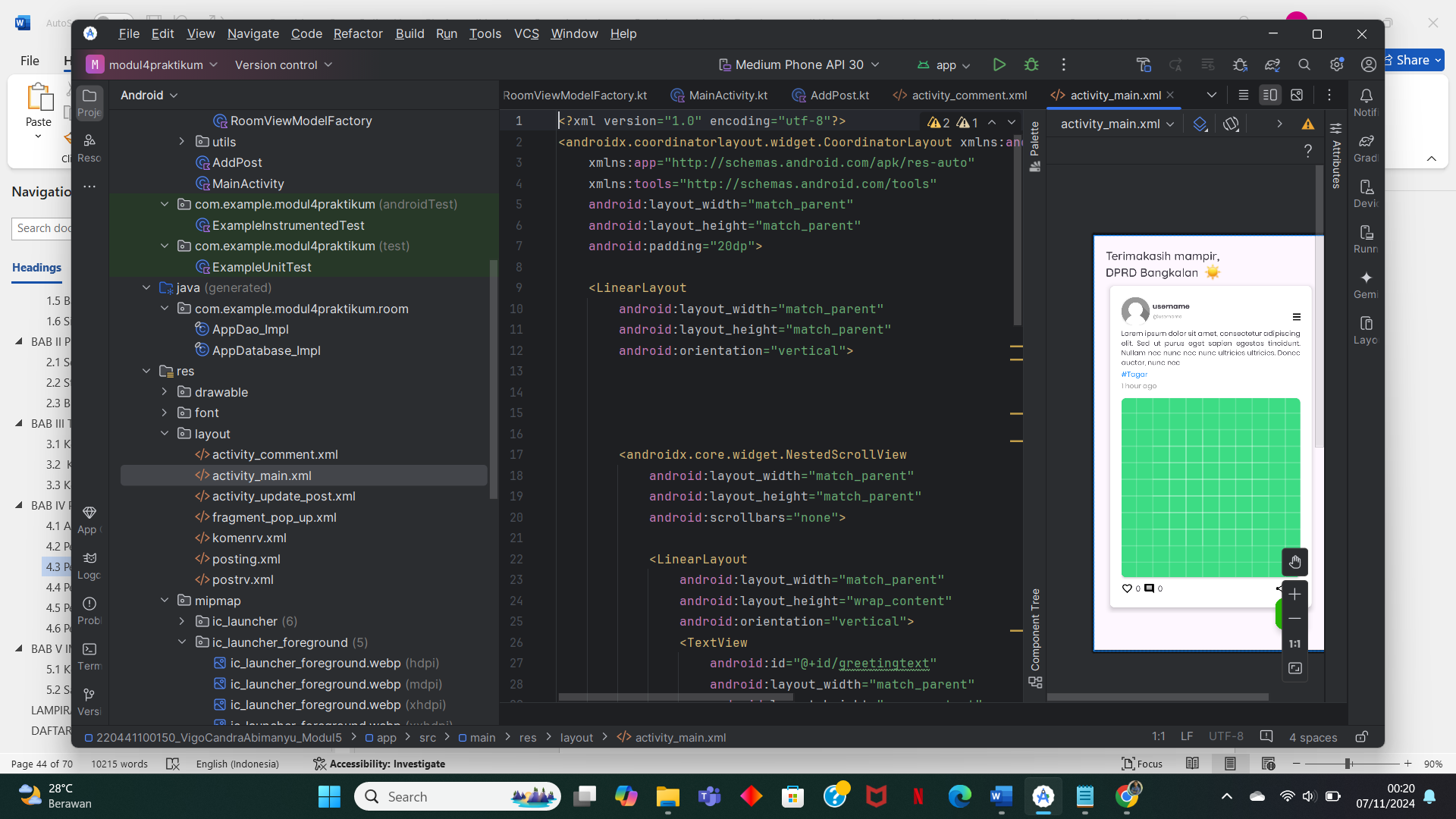
***kotlin+java / utils / Kotlin Class / AddPost.kt***

package com.example.modul4praktikum  
  
import android.content.Intent  
import android.content.pm.PackageManager  
import android.net.Uri  
import android.os.Build  
import android.os.Bundle  
import android.view.View  
import android.widget.ImageButton  
import android.widget.ImageView  
import android.widget.Toast  
import androidx.annotation.RequiresApi  
import androidx.appcompat.app.AlertDialog  
import androidx.appcompat.app.AppCompatActivity  
import androidx.lifecycle.ViewModelProvider  
import com.bumptech.glide.Glide  
import com.esafirm.imagepicker.features.ImagePickerConfig  
import com.esafirm.imagepicker.features.ImagePickerMode  
import com.esafirm.imagepicker.features.ReturnMode  
import com.esafirm.imagepicker.features.registerImagePicker  
import com.example.modul4praktikum.room.AppViewModel  
import com.example.modul4praktikum.room.PostDatabase  
import com.example.modul4praktikum.room.RoomViewModelFactory  
import com.example.modul4praktikum.utils.reduceFileImage  
import com.example.modul4praktikum.utils.uriToFile  
import com.google.android.material.button.MaterialButton  
import com.google.android.material.textfield.TextInputEditText  
import com.google.android.material.textview.MaterialTextView  
import com.modul4praktikum.R  
import java.time.LocalDate  
import java.time.LocalTime  
import java.time.format.DateTimeFormatter  
  
class AddPost : AppCompatActivity(){  
 private var currentImageUri: Uri? = null  
 // Mendeklarasikan ImageView untuk menampilkan gambar yang dipilih  
 private lateinit var playerImage: ImageView  
 // Mendeklarasikan ViewModel untuk interaksi dengan database \*1  
 private lateinit var appViewModel: AppViewModel  
 // Mendeklarasikan EditText untuk input nama pemain  
 private lateinit var playerName: TextInputEditText  
 // Mendeklarasikan EditText untuk input deskripsi pemain  
 private lateinit var playerDescription: TextInputEditText  
 // Mendeklarasikan EditText untuk input gambar pemain  
 private lateinit var playerImageInput: ImageView  
 private lateinit var playerUsername: TextInputEditText  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.posting)  
 checkAndRequestPermissions()  
 // Mendapatkan instance ViewModel  
 val factory = RoomViewModelFactory.getInstance(this)  
 appViewModel = ViewModelProvider(this, factory)[AppViewModel::class.java]  
  
 // Menghubungkan variabel dengan komponen di layout  
 playerImage = findViewById(R.id.imagepost)  
 playerName = findViewById(R.id.etName)  
 playerUsername = findViewById(R.id.etUsername)  
 playerDescription = findViewById(R.id.etCaption)  
 //playerImageInput = findViewById(R.id.player\_image\_edit)  
  
 // Memanggil fungsi onClick untuk menangani aksi klik  
 onClick()  
  
 val back = findViewById<ImageButton>(R.id.back)  
 back.setOnClickListener {  
 val intent = Intent(this, MainActivity::class.java)  
 startActivity(intent)  
 }  
  
 }  
  
 private fun onClick() {  
 val openImagePicker = findViewById<MaterialTextView>(R.id.btnChangePhoto)  
 openImagePicker.setOnClickListener {  
 // Membuat dialog untuk pilihan  
 val builder = AlertDialog.Builder(this)  
 builder.setTitle("Pilih Sumber Gambar")  
 val options = arrayOf("Kamera", "Galeri")  
  
 builder.setItems(options) { dialog, which ->  
 when (which) {  
 0 -> { // Kamera  
 imagePickerLauncher.launch(  
 ImagePickerConfig {  
 mode = ImagePickerMode.SINGLE  
 returnMode = ReturnMode.ALL  
 isShowCamera = true  
 isFolderMode = false // Disable folder mode untuk kamera  
 imageTitle = "Ambil Foto"  
 doneButtonText = "Selesai"  
 }  
 )  
 }  
 1 -> { // Galeri  
 imagePickerLauncher.launch(  
 ImagePickerConfig {  
 mode = ImagePickerMode.SINGLE  
 returnMode = ReturnMode.ALL  
 isFolderMode = true  
 folderTitle = "Galeri"  
 isShowCamera = false  
 imageTitle = "Tekan untuk memilih gambar"  
 doneButtonText = "Selesai"  
 }  
 )  
 }  
 }  
 }  
 builder.show()  
 }  
  
 // Menangani aksi klik pada tombol simpan  
 val btnSavedPlayer = findViewById<MaterialButton>(R.id.post)  
 btnSavedPlayer.setOnClickListener {  
 // Memvalidasi input dan menyimpan data jika valid  
 if (validateInput()) {  
 savedData()  
 }  
 }  
 }  
 private fun validateInput(): Boolean {  
 var error = 0  
  
 if (playerName.text.toString().isEmpty()) {  
 error++  
 playerName.error = "Nama tidak boleh kosong"  
 }  
 if (playerUsername.text.toString().isEmpty()) {  
 error++  
 playerUsername.error = "Username tidak boleh kosong"  
 }  
 if (playerDescription.text.toString().isEmpty()) {  
 error++  
 playerDescription.error = "Deskripsi tidak boleh kosong"  
 }  
 if (currentImageUri == null) {  
 error++  
 Toast.makeText(this, "Gambar tidak boleh kosong", Toast.LENGTH\_SHORT).show()  
 }  
  
 return error == 0  
 }  
  
 // Mendeklarasikan image picker untuk memilih gambar dari galeri  
 private val imagePickerLauncher = registerImagePicker {  
 val firstImage = it.firstOrNull() ?: return@registerImagePicker  
 if (firstImage.uri.toString().isNotEmpty()) {  
 // Menampilkan ImageView jika gambar berhasil dipilih  
 playerImage.visibility = View.VISIBLE  
 // Menyimpan URI gambar yang dipilih  
 currentImageUri = firstImage.uri  
 // Menampilkan pesan bahwa gambar berhasil dimasukkan  
 val cekgambar = findViewById<MaterialTextView>(R.id.btnChangePhoto)  
 cekgambar.setText("Ganti Foto")  
 // playerImageInput.setText("Gambar berhasil dimasukkan")  
  
  
 // Menggunakan library Glide untuk menampilkan gambar yang dipilih  
 Glide.with(playerImage)  
 .load(firstImage.uri)  
 .into(playerImage)  
 } else {  
 // Menyembunyikan ImageView jika tidak ada gambar yang dipilih  
 View.GONE  
 }  
 }  
 @RequiresApi(Build.VERSION\_CODES.Q)  
 private fun savedData() {  
 // Mengubah URI gambar menjadi file dan mengurangi ukuran file  
 val imageFile = currentImageUri?.let { uriToFile(it, this).reduceFileImage() }  
 val TanggalUnggah = LocalDate.now()  
 val jam = LocalTime.now().withSecond(0).withNano(0) // Menghilangkan detik dan milidetik  
  
 val formatter = DateTimeFormatter.ofPattern("HH:mm")  
 val waktuUnggah = "$TanggalUnggah ${jam.format(formatter)}"  
  
 // Membuat objek pemain dengan data yang diinputkan  
 val player = imageFile?.let {  
 PostDatabase(  
 name = playerName.text.toString(),  
 username = "@"+playerUsername.text.toString(),  
 description = playerDescription.text.toString(),  
 likes = 0,  
 waktu = waktuUnggah,  
 image = imageFile  
 )  
 }  
  
 // Menyimpan data pemain ke database  
 if (player != null) appViewModel.insertPost(post = player)  
  
 // Menampilkan pesan bahwa data pemain berhasil ditambahkan  
 Toast.makeText(  
 this@AddPost,  
 "Postingan Berhasil Diunggah",  
 Toast.LENGTH\_SHORT  
 ).show()  
  
 // Mengakhiri activity  
 finish()  
 }  
 private fun checkAndRequestPermissions() {  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.TIRAMISU) {  
 // Android 13 ke atas  
 if (checkSelfPermission(android.Manifest.permission.READ\_MEDIA\_IMAGES) != PackageManager.PERMISSION\_GRANTED ||  
 checkSelfPermission(android.Manifest.permission.CAMERA) != PackageManager.PERMISSION\_GRANTED) {  
 requestPermissions(  
 arrayOf(  
 android.Manifest.permission.READ\_MEDIA\_IMAGES,  
 android.Manifest.permission.CAMERA  
 ),  
 100  
 )  
 }  
 } else {  
 // Android 12 ke bawah  
 if (checkSelfPermission(android.Manifest.permission.READ\_EXTERNAL\_STORAGE) != PackageManager.PERMISSION\_GRANTED ||  
 checkSelfPermission(android.Manifest.permission.WRITE\_EXTERNAL\_STORAGE) != PackageManager.PERMISSION\_GRANTED ||  
 checkSelfPermission(android.Manifest.permission.CAMERA) != PackageManager.PERMISSION\_GRANTED) {  
 requestPermissions(  
 arrayOf(  
 android.Manifest.permission.READ\_EXTERNAL\_STORAGE,  
 android.Manifest.permission.WRITE\_EXTERNAL\_STORAGE,  
 android.Manifest.permission.CAMERA  
 ),  
 100  
 )  
 }  
 }  
 }  
}

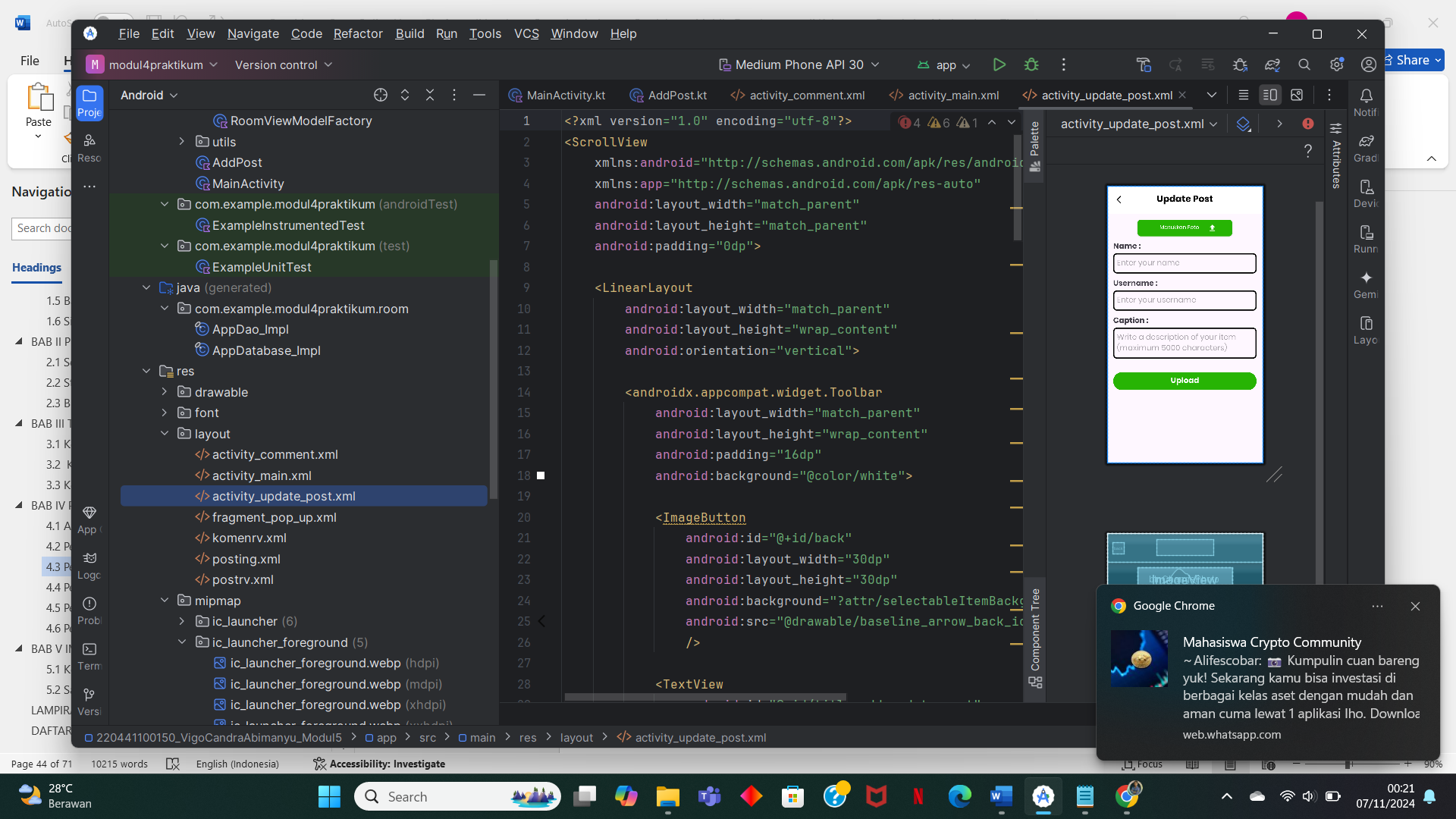
activity\_comment.xml



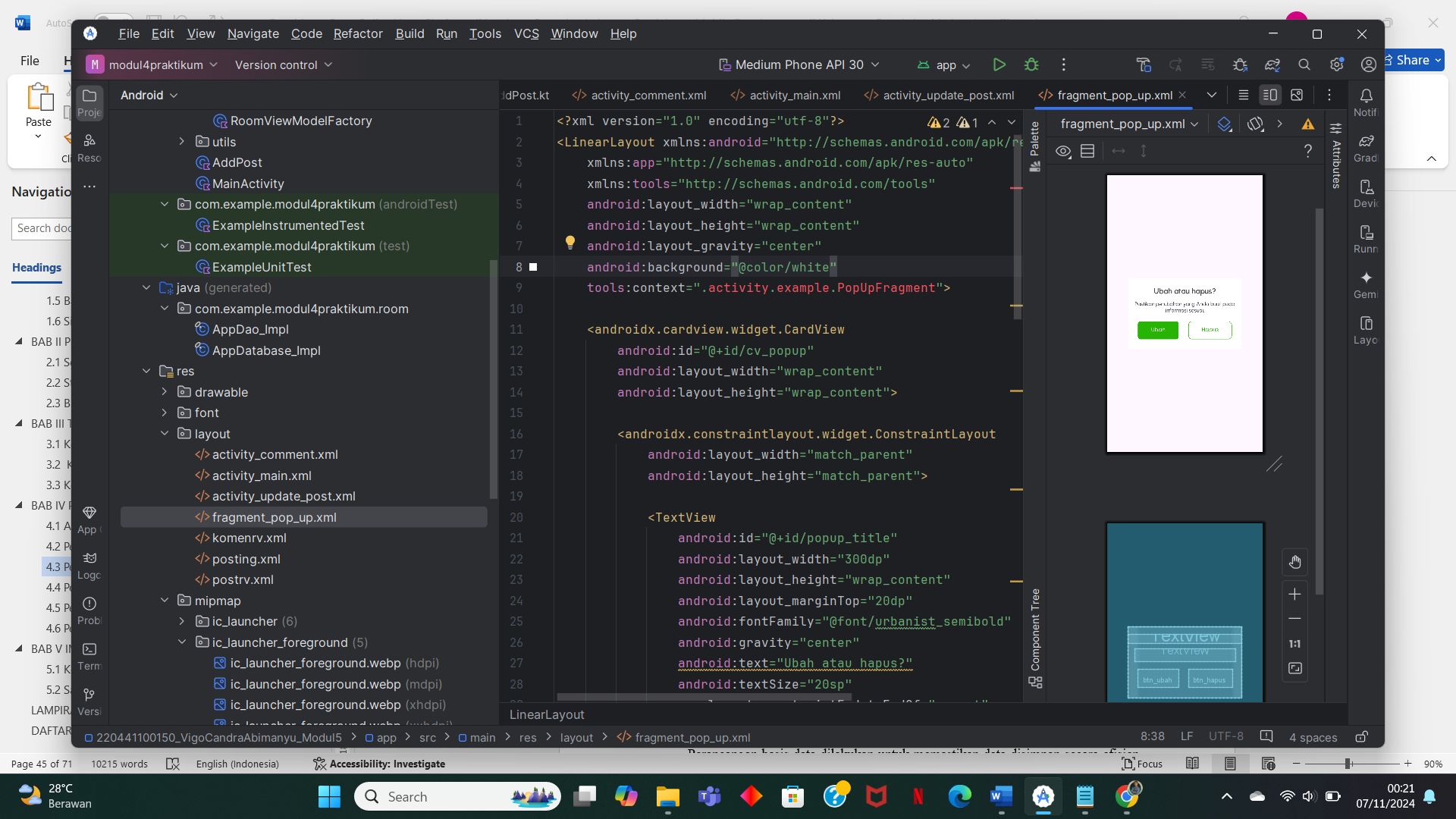
activity\_main.xml



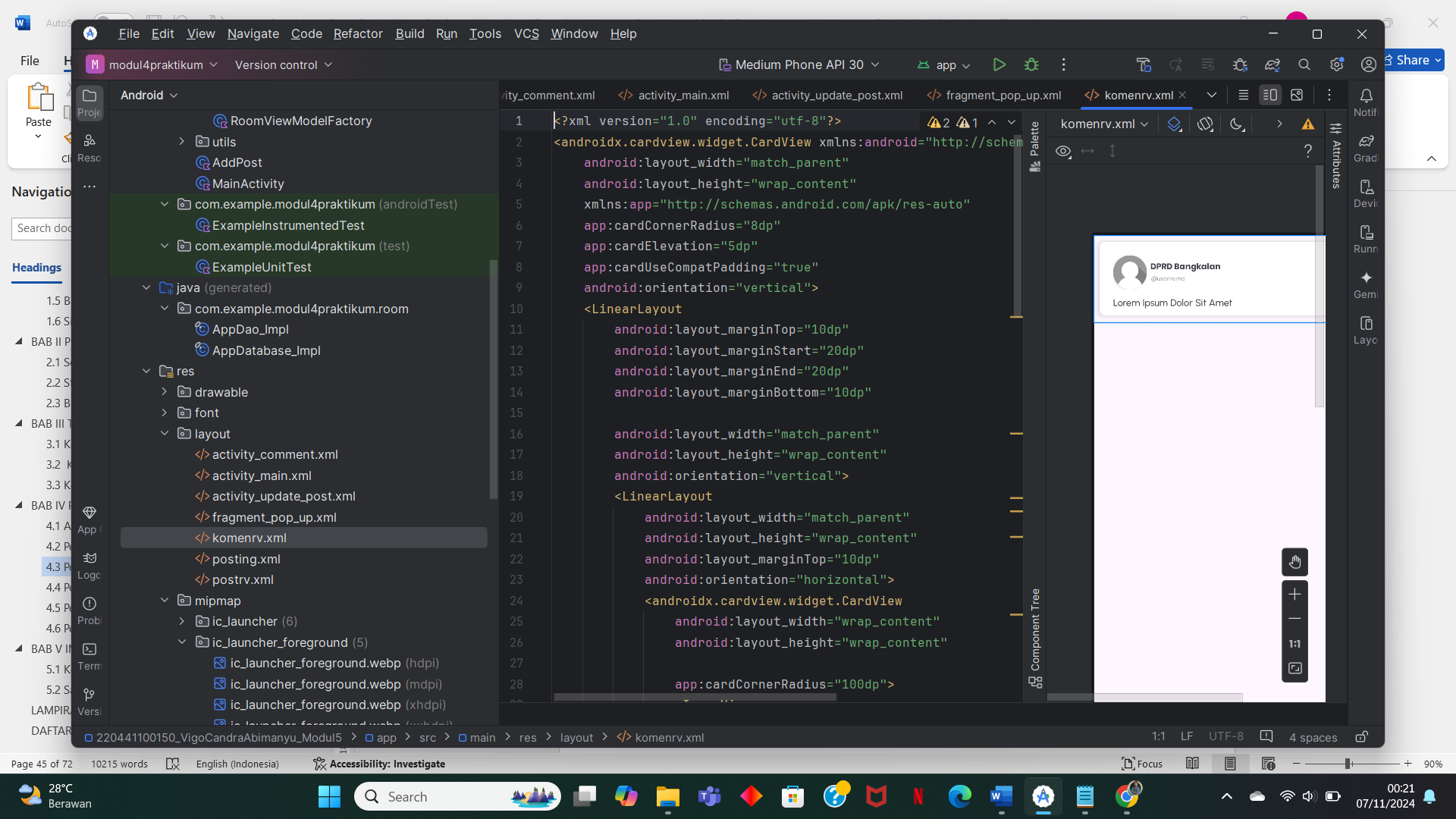
activity\_update\_post



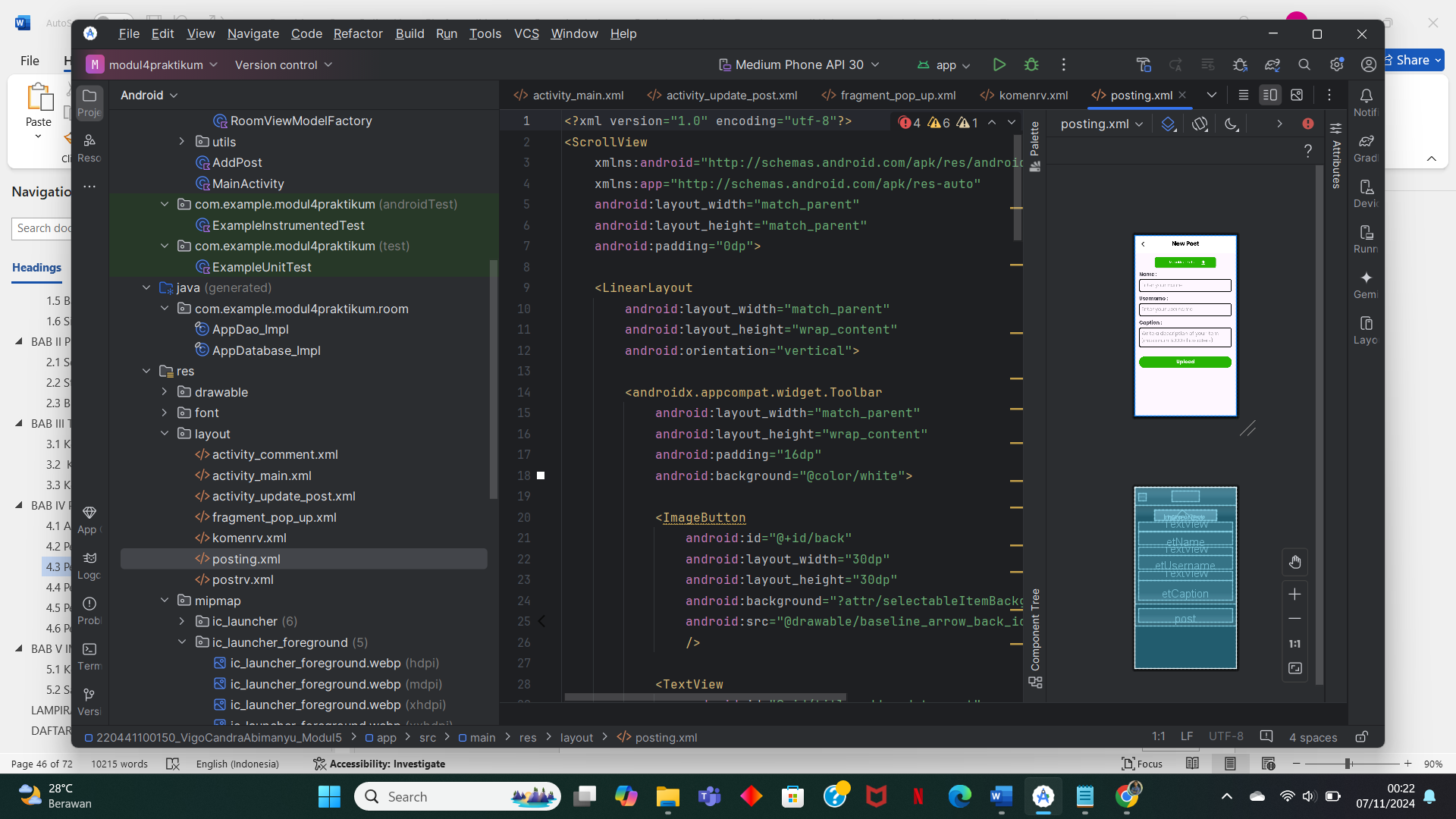
fragment\_pop\_up.xml



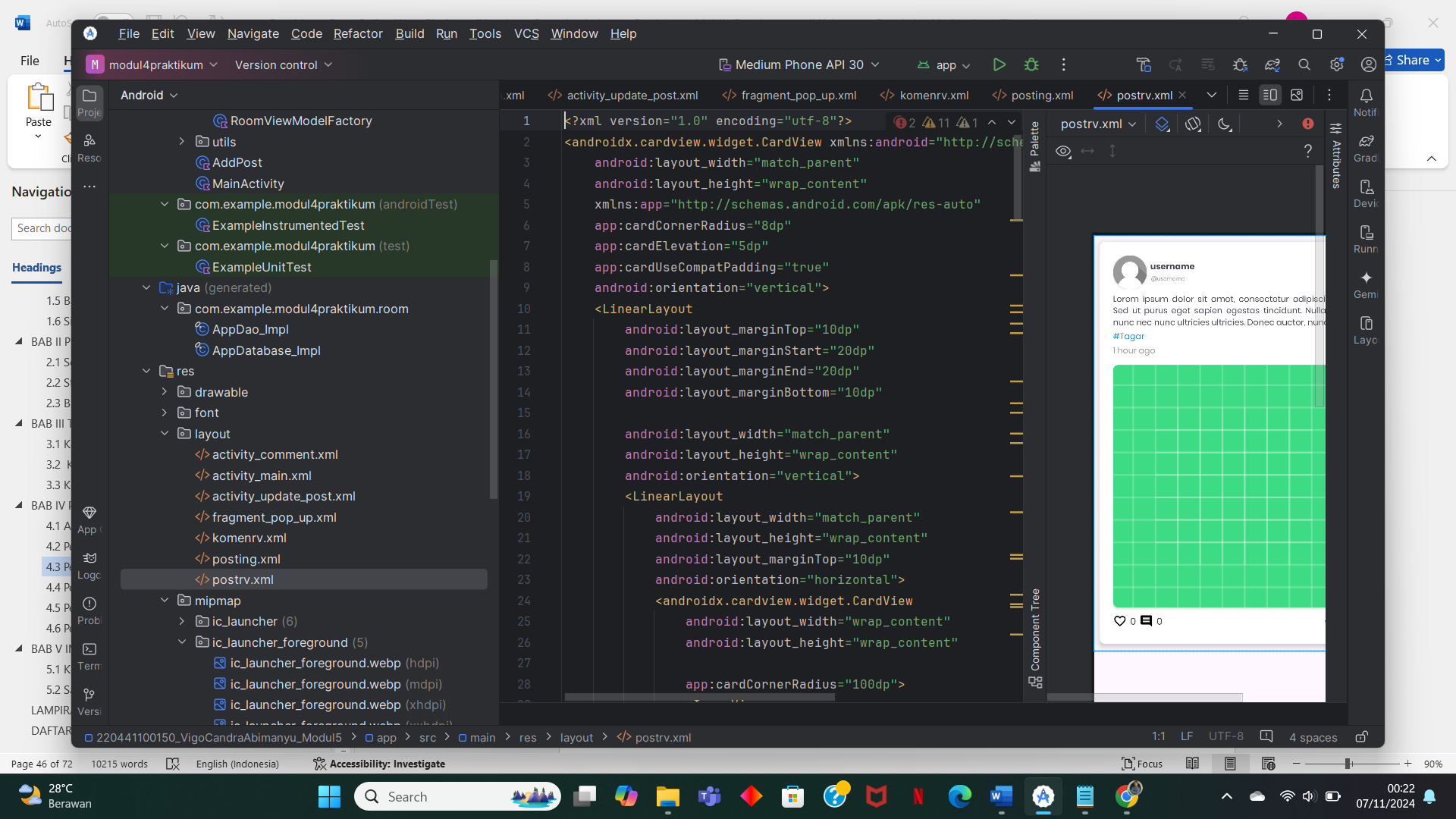
komenrv.xml



posting.xml



postrv.xml



## 4.4 Perancangan Basis Data

Perancangan basis data dilakukan untuk memastikan data disimpan secara efisien dan terstruktur. Beberapa tabel utama yang dirancang dalam sistem ini antara lain:

1. **Tabel User**
   * Menyimpan informasi mengenai pengguna, seperti username, password, dan hak akses.
2. **Tabel Data Utama**
   * Menyimpan data utama yang diproses dalam sistem. Struktur tabel disesuaikan dengan kebutuhan informasi yang akan disimpan.
3. **Tabel Transaksi/Log Aktivitas**
   * Menyimpan data mengenai aktivitas atau transaksi yang dilakukan pengguna di dalam sistem. Ini penting untuk pelacakan dan audit log.
4. **Relasi Antar Tabel**
   * Relasi antar tabel dirancang untuk menjaga integritas data, seperti relasi satu-ke-banyak antara pengguna dan data, atau relasi banyak-ke-banyak bila ada data transaksi yang melibatkan lebih dari satu entitas.

***kotlin+java / adapter / Kotlin Class / PostAdapterRoom***

package com.example.modul4praktikum.Activity  
  
import android.content.Intent  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.core.view.updateLayoutParams  
import androidx.fragment.app.DialogFragment  
import androidx.lifecycle.ViewModelProvider  
import com.example.modul4praktikum.room.AppViewModel  
import com.example.modul4praktikum.room.PostDatabase  
import com.example.modul4praktikum.room.RoomViewModelFactory  
import com.google.android.material.button.MaterialButton  
import com.modul4praktikum.R  
  
class PopUpFragment(private val postDatabase: PostDatabase,position:Int): DialogFragment(){  
 // Mendeklarasikan ViewModel untuk interaksi dengan database  
 private lateinit var appViewModel: AppViewModel  
  
 // Fungsi getTheme digunakan untuk mendapatkan tema dialog.  
 override fun getTheme(): Int {  
 return R.style.*DialogTheme* }  
 // Fungsi onStart dipanggil ketika dialog dimulai.  
 // Fungsi ini digunakan untuk mengatur layout dialog.  
 override fun onStart() {  
 super.onStart()  
 requireDialog().*window*?.*apply* {  
 setLayout(ViewGroup.LayoutParams.*WRAP\_CONTENT*, ViewGroup.LayoutParams.*WRAP\_CONTENT*)  
 }  
  
 *view*?.*updateLayoutParams*<ViewGroup.MarginLayoutParams> {  
 setMargins(16, 16, 16, 16)  
 }  
 }  
 // Fungsi onCreateView digunakan untuk membuat tampilan dialog.  
 // Fungsi ini mengembalikan tampilan yang dibuat dari layout fragment\_pop\_up.  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 return inflater.inflate(R.layout.*fragment\_pop\_up*, container, false)  
 }  
 // Fungsi onViewCreated dipanggil setelah tampilan dialog dibuat.  
 // Fungsi ini digunakan untuk menginisialisasi ViewModel dan menangani aksi klik pada tombol ubah dan hapus.  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
  
 // Mendapatkan instance ViewModel.  
 val factory = RoomViewModelFactory.getInstance(requireContext())  
 appViewModel = ViewModelProvider(this, factory)[AppViewModel::class.*java*]  
  
 // Mendapatkan referensi ke tombol ubah dan hapus.  
 val btnUbah: MaterialButton = view.findViewById(R.id.*btn\_ubah*)  
 val btnHapus: MaterialButton = view.findViewById(R.id.*btn\_hapus*)  
  
 // Menangani aksi klik pada tombol ubah.  
 // Ketika tombol ubah diklik, intent baru dibuat untuk memulai UpdatePlayerRoomActivity dan pemain yang dipilih dikirim sebagai extra.  
 btnUbah.setOnClickListener {  
 val intent = Intent(requireContext(), UpdatePostRoomActivity::class.*java*)  
 intent.putExtra("player", postDatabase)  
 startActivity(intent)  
 dismiss()  
 }  
 // Menangani aksi klik pada tombol hapus.  
 // Ketika tombol hapus diklik, pemain yang dipilih dihapus dari database dan dialog ditutup.  
 btnHapus.setOnClickListener {  
 appViewModel.deletePost(postDatabase)  
 dismiss()  
 }  
  
}  
  
 companion object {  
 const val TAG = "PopUpFragment"  
 }  
}

***kotlin+java / room / Kotlin Class / PostAdapterRoom***

package com.example.modul4praktikum.room  
  
import androidx.room.TypeConverter  
import java.io.File  
  
class AppConverter {  
  
 // Fungsi ini mengubah objek File menjadi String (path file)  
 @TypeConverter  
 fun fromFile(file: File?): String? {  
 // Mengembalikan path dari file jika file tidak null, jika null maka mengembalikan null  
 return file?.*path* }  
  
 // Fungsi ini mengubah String (path file) menjadi objek File  
 @TypeConverter  
 fun toFile(path: String?): File? {  
 // Membuat objek File dari path jika path tidak null, jika null maka mengembalikan null  
 return if (path != null) File(path) else null  
 }  
}

## 4.5 Perancangan Algoritma atau Logika Pemrograman

Algoritma atau logika pemrograman dirancang untuk mengatur cara kerja sistem, termasuk cara sistem memproses input, menyimpan data, serta menghasilkan output. Beberapa algoritma atau logika utama meliputi:

1. **Logika Validasi Input**   
   Menentukan aturan untuk memeriksa input pengguna agar sesuai dengan format dan ketentuan yang telah ditentukan. Validasi ini penting untuk menjaga keakuratan dan kehandalan data yang disimpan dalam sistem.
2. **Algoritma Pemrosesan Data**   
   Menentukan langkah-langkah pemrosesan data di dalam sistem, seperti penghitungan, pencarian, atau pengelompokan data. Algoritma ini dioptimalkan agar sistem dapat bekerja dengan cepat dan efisien.
3. **Logika Pengambilan Keputusan**Logika ini digunakan dalam proses yang memerlukan analisis atau filter terhadap data yang akan ditampilkan, misalnya pada fitur laporan. Keputusan ditentukan berdasarkan parameter atau kondisi tertentu yang telah diatur.

***Kotlin+Java / Activity Class /*** ***Kotlin Class / AddPostActivity***

package com.example.modul4praktikum.Activity  
  
import android.net.Uri  
import android.os.Build  
import android.os.Bundle  
import android.view.View  
import android.widget.Button  
import android.widget.ImageView  
import android.widget.Toast  
import androidx.annotation.RequiresApi  
import androidx.appcompat.app.AppCompatActivity  
import androidx.lifecycle.ViewModelProvider  
import com.bumptech.glide.Glide  
import com.esafirm.imagepicker.features.ImagePickerConfig  
import com.esafirm.imagepicker.features.ImagePickerMode  
import com.esafirm.imagepicker.features.ReturnMode  
import com.esafirm.imagepicker.features.registerImagePicker  
import com.example.modul4praktikum.room.AppViewModel  
import com.example.modul4praktikum.room.PostDatabase  
import com.example.modul4praktikum.room.RoomViewModelFactory  
import com.example.modul4praktikum.utils.reduceFileImage  
import com.example.modul4praktikum.utils.uriToFile  
import com.google.android.material.textfield.TextInputEditText  
import com.modul4praktikum.R  
import java.time.LocalDateTime  
  
class AddPostActivity : AppCompatActivity() {  
 private var currentImageUri: Uri? = null  
 private lateinit var postImage: ImageView  
 private lateinit var appViewModel: AppViewModel  
 private lateinit var name: TextInputEditText  
 private lateinit var username: TextInputEditText  
 private lateinit var description: TextInputEditText  
 private lateinit var playerImageInput: TextInputEditText  
  
 private val imagePickerLauncher = *registerImagePicker* **{** val firstImage = **it**.*firstOrNull*() ?: return@registerImagePicker  
 if (firstImage.uri.toString().*isNotEmpty*()) {  
 postImage.*visibility* = View.*VISIBLE* currentImageUri = firstImage.uri  
 playerImageInput.setText("Gambar berhasil dimasukkan")  
 Glide.with(postImage).load(firstImage.uri).into(postImage)  
 } else {  
 postImage.*visibility* = View.*GONE* }  
 **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*posting*)  
  
 val factory = RoomViewModelFactory.getInstance(this)  
 appViewModel = ViewModelProvider(this, factory)[AppViewModel::class.*java*]  
  
 postImage = findViewById(R.id.*imagepost*)  
 name = findViewById(R.id.*etName*)  
 username = findViewById(R.id.*etUsername*)  
 description = findViewById(R.id.*etCaption*)  
 playerImageInput = findViewById(R.id.*imagepost*)  
  
 onClick()  
 }  
  
 private fun onClick() {  
 val openImagePicker = findViewById<Button>(R.id.*btnChangePhoto*)  
 openImagePicker.setOnClickListener **{** imagePickerLauncher.launch(  
 ImagePickerConfig **{** mode = ImagePickerMode.*SINGLE* returnMode = ReturnMode.*ALL* isFolderMode = true  
 folderTitle = "Galeri"  
 isShowCamera = false  
 imageTitle = "Tekan untuk memilih gambar"  
 doneButtonText = "Selesai"  
 **}** )  
 **}** val btnSavedPlayer = findViewById<Button>(R.id.*post*)  
 btnSavedPlayer.setOnClickListener **{** if (validateInput()) {  
 savedData()  
 }  
 **}** }  
  
 private fun validateInput(): Boolean {  
 var error = 0  
  
 if (name.*text*.*toString*().*isEmpty*()) {  
 error++  
 name.*error* = "Nama tidak boleh kosong"  
 }  
 if (username.*text*.*toString*().*isEmpty*()) {  
 error++  
 username.*error* = "Username tidak boleh kosong"  
 }  
 if (description.*text*.*toString*().*isEmpty*()) {  
 error++  
 description.*error* = "Deskripsi tidak boleh kosong"  
 }  
 if (currentImageUri == null) {  
 error++  
 playerImageInput.*error* = "Gambar tidak boleh kosong"  
 }  
  
 return error == 0  
 }  
  
 @RequiresApi(Build.VERSION\_CODES.*Q*)  
 private fun savedData() {  
 val imageFile = currentImageUri?.*let* **{** *uriToFile*(**it**, this).*reduceFileImage*() **}** val post = imageFile?.*let* **{** PostDatabase(  
 name = name.*text*.*toString*(),  
 username = username.*text*.*toString*(),  
 description = description.*text*.*toString*(),  
 waktu = LocalDateTime.now().toString(),  
 image = **it** )  
 **}** if (post != null) appViewModel.insertPost(post)  
  
 Toast.makeText(this, "Data post berhasil ditambahkan", Toast.*LENGTH\_SHORT*).show()  
 finish()  
 }  
}

***Kotlin+Java / Activity Class / Kotlin Class / AddPostActivity***

package com.example.modul4praktikum.Activity  
  
import android.content.Intent  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.core.view.updateLayoutParams  
import androidx.fragment.app.DialogFragment  
import androidx.lifecycle.ViewModelProvider  
import com.example.modul4praktikum.room.AppViewModel  
import com.example.modul4praktikum.room.PostDatabase  
import com.example.modul4praktikum.room.RoomViewModelFactory  
import com.google.android.material.button.MaterialButton  
import com.modul4praktikum.R  
  
class PopUpFragment(private val postDatabase: PostDatabase,position:Int): DialogFragment(){  
 // Mendeklarasikan ViewModel untuk interaksi dengan database  
 private lateinit var appViewModel: AppViewModel  
  
 // Fungsi getTheme digunakan untuk mendapatkan tema dialog.  
 override fun getTheme(): Int {  
 return R.style.DialogTheme  
 }  
 // Fungsi onStart dipanggil ketika dialog dimulai.  
 // Fungsi ini digunakan untuk mengatur layout dialog.  
 override fun onStart() {  
 super.onStart()  
 requireDialog().window?.apply {  
 setLayout(ViewGroup.LayoutParams.WRAP\_CONTENT, ViewGroup.LayoutParams.WRAP\_CONTENT)  
 }  
  
 view?.updateLayoutParams<ViewGroup.MarginLayoutParams> {  
 setMargins(16, 16, 16, 16)  
 }  
 }  
 // Fungsi onCreateView digunakan untuk membuat tampilan dialog.  
 // Fungsi ini mengembalikan tampilan yang dibuat dari layout fragment\_pop\_up.  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 return inflater.inflate(R.layout.fragment\_pop\_up, container, false)  
 }  
 // Fungsi onViewCreated dipanggil setelah tampilan dialog dibuat.  
 // Fungsi ini digunakan untuk menginisialisasi ViewModel dan menangani aksi klik pada tombol ubah dan hapus.  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
  
 // Mendapatkan instance ViewModel.  
 val factory = RoomViewModelFactory.getInstance(requireContext())  
 appViewModel = ViewModelProvider(this, factory)[AppViewModel::class.java]  
  
 // Mendapatkan referensi ke tombol ubah dan hapus.  
 val btnUbah: MaterialButton = view.findViewById(R.id.btn\_ubah)  
 val btnHapus: MaterialButton = view.findViewById(R.id.btn\_hapus)  
  
 // Menangani aksi klik pada tombol ubah.  
 // Ketika tombol ubah diklik, intent baru dibuat untuk memulai UpdatePlayerRoomActivity dan pemain yang dipilih dikirim sebagai extra.  
 btnUbah.setOnClickListener {  
 val intent = Intent(requireContext(), UpdatePostRoomActivity::class.java)  
 intent.putExtra("player", postDatabase)  
 startActivity(intent)  
 dismiss()  
 }  
 // Menangani aksi klik pada tombol hapus.  
 // Ketika tombol hapus diklik, pemain yang dipilih dihapus dari database dan dialog ditutup.  
 btnHapus.setOnClickListener {  
 appViewModel.deletePost(postDatabase)  
 dismiss()  
 }  
  
}  
  
 companion object {  
 const val TAG = "PopUpFragment"  
 }  
}

***Kotlin+Java / Activity Class / Kotlin Class / UpdatePostRoomActivity***

package com.example.modul4praktikum.Activity  
  
import android.content.Intent  
import android.net.Uri  
import android.os.Build  
import android.os.Bundle  
import android.util.Log  
import android.view.View  
import android.widget.Button  
import android.widget.ImageButton  
import android.widget.ImageView  
import android.widget.TextView  
import android.widget.Toast  
import androidx.annotation.RequiresApi  
import androidx.appcompat.app.AppCompatActivity  
import androidx.lifecycle.ViewModelProvider  
import com.bumptech.glide.Glide  
import com.esafirm.imagepicker.features.ImagePickerConfig  
import com.esafirm.imagepicker.features.ImagePickerMode  
import com.esafirm.imagepicker.features.ReturnMode  
import com.esafirm.imagepicker.features.registerImagePicker  
import com.example.modul4praktikum.MainActivity  
import com.example.modul4praktikum.room.AppViewModel  
import com.example.modul4praktikum.room.PostDatabase  
import com.example.modul4praktikum.room.RoomViewModelFactory  
import com.example.modul4praktikum.utils.reduceFileImage  
import com.example.modul4praktikum.utils.uriToFile  
import com.google.android.material.button.MaterialButton  
import com.google.android.material.textfield.TextInputEditText  
import com.google.android.material.textview.MaterialTextView  
import com.modul4praktikum.R  
import java.io.File  
import java.time.LocalDate  
import java.time.LocalDateTime  
import java.time.LocalTime  
import java.time.format.DateTimeFormatter  
  
class UpdatePostRoomActivity : AppCompatActivity() {  
 // Mendeklarasikan variabel untuk menyimpan URI gambar saat ini dan foto lama.  
 private var currentImageUri: Uri? = null  
 private var oldPhoto: File? = null  
 private lateinit var getDataPost: PostDatabase  
 private lateinit var postImage: ImageView  
 private lateinit var appViewModel: AppViewModel  
 private lateinit var name: TextInputEditText  
 private lateinit var username: TextInputEditText  
 private lateinit var description: TextInputEditText  
 private lateinit var playerImageInput: ImageView  
  
 // Mendeklarasikan imagePickerLauncher untuk memilih gambar dari galeri atau kamera.  
 private val imagePickerLauncher = registerImagePicker {  
 val firstImage = it.firstOrNull() ?: return@registerImagePicker  
 if (firstImage.uri.toString().isNotEmpty()) {  
 postImage.visibility = View.VISIBLE  
 currentImageUri = firstImage.uri  
  
 // Menggunakan Glide untuk memuat gambar ke ImageView.  
 Glide.with(postImage)  
 .load(firstImage.uri)  
 .into(postImage)  
 } else {  
 postImage.visibility = View.GONE  
 }  
 }  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_update\_post)  
  
 getDataPost = intent.getParcelableExtra("player")!!  
 val factory = RoomViewModelFactory.getInstance(this)  
 appViewModel = ViewModelProvider(this, factory)[AppViewModel::class.java]  
  
 postImage = findViewById(R.id.imagepost)  
 name = findViewById(R.id.etName)  
 username = findViewById(R.id.etUsername)  
 description = findViewById(R.id.etCaption)  
 playerImageInput = findViewById(R.id.imagepost)  
  
 name.setText(getDataPost!!.name)  
 username.setText(getDataPost!!.username)  
 description.setText(getDataPost!!.description)  
 Glide.with(this)  
 .load(getDataPost.image)  
 .into(postImage)  
  
 oldPhoto = getDataPost.image  
  
 onClick()  
 }  
  
 private fun onClick() {  
 val btnSavedPlayer = findViewById<Button>(R.id.btnUpdate)  
 btnSavedPlayer.setOnClickListener {  
 if (validateInput()) {  
 savedData()  
 }  
 }  
 val openImagePicker = findViewById<TextView>(R.id.btnChangePhoto)  
 openImagePicker.setOnClickListener {  
 imagePickerLauncher.launch(  
 ImagePickerConfig {  
 mode = ImagePickerMode.SINGLE  
 returnMode = ReturnMode.ALL  
 isFolderMode = true  
 folderTitle = "Galeri"  
 isShowCamera = false  
 imageTitle = "Tekan untuk memilih gambar"  
 doneButtonText = "Selesai"  
 }  
 )  
 }  
 }  
  
 private fun validateInput(): Boolean {  
 if (name.text.toString().isEmpty()) {  
 name.error = "Nama tidak boleh kosong"  
 return false  
 }  
 if (username.text.toString().isEmpty()) {  
 username.error = "Username tidak boleh kosong"  
 return false  
 }  
 if (description.text.toString().isEmpty()) {  
 description.error = "Deskripsi tidak boleh kosong"  
 return false  
 }  
 if (currentImageUri == null) {  
 Toast.makeText(this, "Gambar tidak boleh kosong", Toast.LENGTH\_SHORT).show()  
 return false  
 }  
 return true  
 }  
 @RequiresApi(Build.VERSION\_CODES.Q)  
 private fun savedData () {  
 val imageFile = currentImageUri?.let { uriToFile(it, this).reduceFileImage() }  
 val TanggalUnggah = LocalDate.now()  
 val jam = LocalTime.now().withSecond(0).withNano(0) // Menghilangkan detik dan milidetik  
  
 val formatter = DateTimeFormatter.ofPattern("HH:mm")  
 val waktuUnggah = "$TanggalUnggah ${jam.format(formatter)}"  
 val post = imageFile?.let {  
 PostDatabase(  
 id = getDataPost.id,  
 name = name.text.toString(),  
 username = "@"+username.text.toString(),  
 description = description.text.toString(),  
 waktu = "Diubah Pada "+waktuUnggah+" WIB",  
 image = it  
 )  
 }  
 Log.d("post", post.toString())  
  
 if (post != null) appViewModel.updatePost(post)  
 // Menampilkan pesan bahwa data pemain berhasil diubah.  
 Toast.makeText(  
 this@UpdatePostRoomActivity,  
 "Data post berhasil diubah",  
 Toast.LENGTH\_SHORT  
 ).show()  
 // Menutup activity.  
 finish()  
  
 }  
}

***Kotlin+Java / Activity Class / Kotlin Class / AppDao***

package com.example.modul4praktikum.room  
  
import androidx.lifecycle.LiveData  
import androidx.room.Dao  
import androidx.room.Delete  
import androidx.room.Insert  
import androidx.room.OnConflictStrategy  
import androidx.room.Query  
import androidx.room.Update  
  
@Dao  
interface AppDao {  
 @Insert(onConflict = OnConflictStrategy.IGNORE)  
 fun insertPost(post:PostDatabase)  
 @Query("SELECT \* from postdatabase ORDER BY id DESC")  
 fun getAllPost(): LiveData<List<PostDatabase>>  
 @Delete  
 fun deletePost(post: PostDatabase)  
 @Update  
 fun updatePost(post: PostDatabase)  
  
  
}

***Kotlin+Java / Activity Class / Kotlin Class / AppDatabase***

package com.example.modul4praktikum.room  
  
import android.content.Context  
import androidx.room.Database  
import androidx.room.Room  
import androidx.room.RoomDatabase  
import androidx.room.TypeConverters  
  
@Database(entities = [PostDatabase::class], version = 4)  
  
// Menggunakan konverter tipe khusus untuk mengubah tipe data File menjadi String dan sebaliknya  
@TypeConverters(AppConverter::class)  
  
abstract class AppDatabase :RoomDatabase() {  
 // Mendeklarasikan fungsi abstrak yang mengembalikan AppDao  
 abstract fun appDao(): AppDao  
  
 // Membuat objek pendamping untuk AppDatabase  
 companion object {  
 // Mendeklarasikan variabel INSTANCE yang akan menyimpan instance dari AppDatabase  
 @Volatile  
 private var INSTANCE: AppDatabase? = null  
  
 // Membuat fungsi statis untuk mendapatkan instance database  
 @JvmStatic  
 fun getDatabase(context: Context): AppDatabase {  
 // Jika INSTANCE null, maka akan dibuat instance baru  
 if (INSTANCE == null) {  
 // Menggunakan synchronized untuk mencegah akses bersamaan dari beberapa thread  
 *synchronized*(AppDatabase::class.*java*) {  
 // Membuat instance baru dari AppDatabase  
 INSTANCE = Room.databaseBuilder(  
 context.*applicationContext*,  
 AppDatabase::class.*java*, "app\_database"  
 )  
 .fallbackToDestructiveMigration()  
 .build()  
 }  
 }  
 // Mengembalikan instance dari AppDatabase  
 return INSTANCE as AppDatabase  
 }  
 }  
}

***Kotlin+Java / Activity Class / Kotlin Class / AppRepository***

package com.example.modul4praktikum.room  
  
import androidx.lifecycle.LiveData  
import com.example.modul4praktikum.utils.AppExecutors  
  
class AppRepository private constructor(private val appDao: AppDao, private val appExecutors: AppExecutors) {  
  
 // Mendapatkan semua data pemain dari database  
 fun getAllPost(): LiveData<List<PostDatabase>> = appDao.getAllPost()  
  
 // Memasukkan data pemain ke dalam database  
 fun insertPost(post: PostDatabase) {  
 // Menjalankan operasi insert di thread yang berbeda  
 appExecutors.diskIO().execute { appDao.insertPost(post) }  
 }  
  
 // Menghapus data post dari database  
 fun deletePost(post: PostDatabase) {  
 // Menjalankan operasi delete di thread yang berbeda  
 appExecutors.diskIO().execute { appDao.deletePost(post) }  
 }  
 // Memperbarui data post dalam database  
 fun updatePost(post: PostDatabase) {  
 // Menjalankan operasi update di thread yang berbeda  
 appExecutors.diskIO().execute { appDao.updatePost(post) }  
 }  
  
 companion object {  
 // Variabel untuk menyimpan instance dari AppRepository  
 @Volatile  
 private var instance: AppRepository? = null  
  
 // Mendapatkan instance dari AppRepository  
 fun getInstance(  
 appDao: AppDao,  
 appExecutors: AppExecutors  
 ): AppRepository =  
 // Jika instance null, maka akan dibuat instance baru  
 instance ?: *synchronized*(this) {  
 instance ?: AppRepository(appDao, appExecutors)  
 }.*also* { instance = it }  
 }  
}

***Kotlin+Java / Activity Class / Kotlin Class / AppViewModel***

package com.example.modul4praktikum.room  
  
import androidx.lifecycle.LiveData  
import androidx.lifecycle.ViewModel  
  
class AppViewModel(private val appRepository: AppRepository) : ViewModel() {  
 fun insertPost(post: PostDatabase) {  
 appRepository.insertPost(post)  
 }  
  
 fun getAllPost(): LiveData<List<PostDatabase>> {  
 return appRepository.getAllPost()  
 }  
 // Fungsi deletePost digunakan untuk menghapus data post dari database.  
 fun deletePost(post: PostDatabase) {  
 appRepository.deletePost(post)  
 }  
 // Fungsi updatePost digunakan untuk memperbarui data post dalam database.  
 fun updatePost(post: PostDatabase) {  
 appRepository.updatePost(post)  
 }  
}

***Kotlin+Java / Activity Class / Kotlin Class / AppViewModel AppViewModel***

package com.example.modul4praktikum.room  
  
import android.os.Parcel  
import android.os.Parcelable  
import androidx.room.ColumnInfo  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
import java.io.File  
  
@Entity  
data class PostDatabase(  
 @PrimaryKey(autoGenerate = true)  
 @ColumnInfo(name = "id")  
 val id: Int = 0,  
 val name: String,  
 val username: String,  
 val description: String,  
 val waktu: String,  
 var likes:Int = 0,  
 val image: File  
) : Parcelable {  
 constructor(parcel: Parcel) : this(  
 parcel.readInt(),  
 parcel.readString()!!,  
 parcel.readString()!!,  
 parcel.readString()!!,  
 parcel.readString()!!,  
 parcel.readInt(),  
 File(parcel.readString()!!)  
 )  
  
 override fun writeToParcel(parcel: Parcel, flags: Int) {  
 parcel.writeInt(id)  
 parcel.writeString(name)  
 parcel.writeString(username)  
 parcel.writeString(description)  
 parcel.writeString(waktu)  
 parcel.writeInt(likes)  
 parcel.writeString(image.*path*)  
 }  
  
 override fun describeContents(): Int {  
 return 0  
 }  
  
 companion object CREATOR : Parcelable.Creator<PostDatabase> {  
 override fun createFromParcel(parcel: Parcel): PostDatabase {  
 return PostDatabase(parcel)  
 }  
  
 override fun newArray(size: Int): Array<PostDatabase?> {  
 return *arrayOfNulls*(size)  
 }  
 }  
}

***Kotlin+Java / Activity Class / Kotlin Class / AppViewModel AppViewModel***

package com.example.modul4praktikum.room  
  
import android.os.Parcel  
import android.os.Parcelable  
import androidx.room.ColumnInfo  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
import java.io.File  
  
@Entity  
data class PostDatabase(  
 @PrimaryKey(autoGenerate = true)  
 @ColumnInfo(name = "id")  
 val id: Int = 0,  
 val name: String,  
 val username: String,  
 val description: String,  
 val waktu: String,  
 var likes:Int = 0,  
 val image: File  
) : Parcelable {  
 constructor(parcel: Parcel) : this(  
 parcel.readInt(),  
 parcel.readString()!!,  
 parcel.readString()!!,  
 parcel.readString()!!,  
 parcel.readString()!!,  
 parcel.readInt(),  
 File(parcel.readString()!!)  
 )  
  
 override fun writeToParcel(parcel: Parcel, flags: Int) {  
 parcel.writeInt(id)  
 parcel.writeString(name)  
 parcel.writeString(username)  
 parcel.writeString(description)  
 parcel.writeString(waktu)  
 parcel.writeInt(likes)  
 parcel.writeString(image.*path*)  
 }  
  
 override fun describeContents(): Int {  
 return 0  
 }  
  
 companion object CREATOR : Parcelable.Creator<PostDatabase> {  
 override fun createFromParcel(parcel: Parcel): PostDatabase {  
 return PostDatabase(parcel)  
 }  
  
 override fun newArray(size: Int): Array<PostDatabase?> {  
 return *arrayOfNulls*(size)  
 }  
 }  
}

***Kotlin+Java / Activity Class / Kotlin Class / RoomViewModellFactory***

package com.example.modul4praktikum.room  
  
import android.content.Context  
import androidx.lifecycle.ViewModel  
import androidx.lifecycle.ViewModelProvider  
import com.example.modul4praktikum.utils.DependencyInjection  
  
class RoomViewModelFactory private constructor(private val appRepository: AppRepository) :  
 ViewModelProvider.NewInstanceFactory() {  
 @Suppress("UNCHECKED\_CAST")  
 override fun <T : ViewModel> create(modelClass: Class<T>): T {  
 if (modelClass.isAssignableFrom(AppViewModel::class.java)) {  
 return AppViewModel(appRepository) as T  
 }  
 throw IllegalArgumentException("Unknown ViewModel class: " + modelClass.name)  
 }  
  
 companion object {  
 @Volatile  
 private var instance: RoomViewModelFactory? = null  
 fun getInstance(context: Context): RoomViewModelFactory =  
 instance ?: synchronized(this) {  
 instance ?: RoomViewModelFactory(DependencyInjection.provideRepository(context))  
 }.also { instance = it }  
 }  
 }

## 4.6 Perancangan Keamanan Sistem

Perancangan keamanan dilakukan untuk melindungi sistem dari akses tidak sah serta menjaga kerahasiaan data. Beberapa elemen keamanan yang diterapkan dalam sistem antara lain:

1. **Autentikasi dan Autorisasi**
   * Setiap pengguna harus melakukan autentikasi untuk memastikan bahwa hanya pengguna terdaftar yang dapat mengakses sistem. Sistem juga akan mengelola hak akses (otorisasi) agar pengguna hanya dapat mengakses fitur sesuai dengan peran mereka.
2. **Enkripsi Data Sensitif**
   * Data-data sensitif seperti password atau informasi pribadi pengguna akan dienkripsi sebelum disimpan dalam database untuk menjaga kerahasiaannya.
3. **Log Aktivitas**
   * Sistem mencatat setiap aktivitas pengguna, sehingga setiap perubahan atau aktivitas di dalam sistem dapat dilacak. Hal ini penting untuk keamanan serta memudahkan dalam proses audit.

Dengan perancangan sistem yang terstruktur ini, diharapkan sistem dapat memenuhi kebutuhan dan memecahkan permasalahan yang diangkat secara efektif.

# **BAB V IMPLEMENTASI SISTEM**

**PENUTUP**

## 5.1 Kesimpulan

Kerja praktik ini berhasil merancang dan mengimplementasikan aplikasi pengiriman surat online lintas platform yang dapat digunakan oleh lembaga pemerintah. Aplikasi ini dikembangkan dengan tujuan mempermudah dan mempercepat proses pengiriman dokumen resmi secara digital, mengurangi ketergantungan pada pengiriman fisik, serta meningkatkan efisiensi operasional. Berdasarkan uji coba, aplikasi menunjukkan kinerja yang baik dalam aspek:

* **Waktu Respons**: Aplikasi mampu mengirim dan menerima dokumen dalam waktu yang cepat, bahkan saat diakses oleh banyak pengguna.
* **Keamanan Data**: Penerapan enkripsi dan validasi otorisasi memastikan dokumen hanya dapat diakses oleh penerima yang berwenang, sehingga keamanan data terjamin.
* **Kestabilan Aplikasi**: Prototipe berhasil diuji pada perangkat lintas platform dan tetap stabil di bawah beban yang tinggi, memastikan konsistensi kinerja.

Keseluruhan solusi yang dirancang mampu mengatasi permasalahan yang dihadapi lembaga pemerintah dalam hal efisiensi dan keamanan pengiriman surat digital. Aplikasi ini diharapkan dapat menjadi landasan bagi digitalisasi proses administrasi di lingkungan pemerintahan.

## 5.2 Saran-saran

Untuk pengembangan lebih lanjut, beberapa saran yang dapat dipertimbangkan antara lain:

1. **Pengembangan Fitur Notifikasi Real-time**: Menambahkan notifikasi langsung kepada pengguna ketika surat diterima atau dibaca, untuk meningkatkan efisiensi komunikasi.
2. **Optimasi Kinerja di Perangkat dengan Spesifikasi Rendah**: Meningkatkan performa aplikasi pada perangkat dengan spesifikasi lebih rendah, agar aplikasi tetap bisa diakses di perangkat yang beragam.
3. **Integrasi dengan Sistem Manajemen Dokumen Internal**: Mengintegrasikan aplikasi dengan sistem manajemen dokumen yang telah ada di lembaga pemerintah untuk memastikan kemudahan akses dan pencarian dokumen.
4. **Audit Keamanan Berkala**: Mengadakan audit keamanan secara berkala untuk memastikan bahwa sistem tetap aman dari ancaman siber yang berkembang.
5. **Pengembangan Modul Analisis Data**: Mengembangkan modul analisis yang dapat membantu instansi untuk melacak volume dan tren pengiriman dokumen, yang bisa digunakan untuk evaluasi kebijakan di masa mendatang.

Dengan implementasi saran-saran tersebut, aplikasi diharapkan dapat lebih optimal, andal, dan mampu menghadapi kebutuhan di masa depan, mendukung digitalisasi dan modernisasi di lembaga pemerintah.

**STRUKTUR ORGANISASI SEKRETARIAT DPRD**

**SEKRETARIS DPRD**

**RUDIYANTO, S.Sos, MM**

Pembina Utama Muda ( IVc )

NIP.197405131993021001

**KEPALA BAGIAN HUKUM DAN PERSIDANGAN**

**DWI ASTUTI, SE, MM**

Pembina (IVa)

NIP. 197608282002122007

**PERISALAH LEGISLATI**

**AHLI MUDA**

**MAHRUS ALIE, S.HI**

Pembina ( IVa )

NIP. 197911262007011006

***PENGELOLA PERSIDANGAN***

ADI SOLIHIN, SH

Penata Tk. I ( IIId )

NIP.196708052007011025

***PENYUSUN NASKAH RAPAT PIMPINAN***

EDDY ANANG TRIADI, SE

Penata ( IIIc )

NIP. 197803292014071001

***PERISALAH LEGISLATIF***

***AHLI PERTAMA***

RR. NURUL MUTMAINNAH, SE

NIP. 198605032023212034

***PERISALAH LEGISLATIF***

***AHLI PERTAMA***

IWAN HERU SETIAWAN, SE

NIP. 197709202023211003

ADITYA BUDIPRAYA

**PERANCANG PERUNDANG**

**UNDANGAN AHLI MUDA**

**SYARIFAH, SE**

Penata Tk. I ( IIId )

NIP. 196805252001122001

***PENGADMINISTRASI HUKUM***

BUHARI

Pengatur Tk. I ( IId )

NIP. 197201022014071005

SITI ELYATI, SE

**ANALIS PEMANTAUAN PERATURAN PERUNDANG UNDANGAN LEGISLATIF**

**AHLI MUDA**

**ERRY AGUSTIONATA, S.Sos**

Penata Tk. I ( IIId )

NIP. 197608012006041024

***PENGADMINISTRASI HUKUM***

ACHMAD SODIK

Pengatur Muda Tk. I (IIb)

NIP. 197201022014071005

HENDAR SETYA NINGRUM

NURUL HASANAH, SE

RAMDHAN SUNARYO

MAWFIROH

MARIANA MURNI, SE

**KEPALA BAGIAN UMUM**

**MOHAMAD SYAHRUL FAIDULLAH, S.Kom, MM**

PENATA Tk. I ( IIId)

NIP. 197709172011011005

**KEPALA SUB BAGIAN**

**ADMINISTRASI UMUM dan KEPEGAWAIAN**

***PENGELOLA KEPEGAWAIAN***

SOFIA FARIDA AMIN, SE

Penata ( IIIc )

NIP. 198006012008102027

***PENGELOLA SURAT***

CANDRA TRIONO, ST

Penata Tk. I ( IIId )

NIP. 197703012006041011

***PENGADMINISTRASI KEPEGAWAIAN***

BAMBANG HARYADI

Pengatur Tk. I ( IId )

NIP. 196807032007011021

***PENGADMINISTRASI PERSURATAN***

BAMBANG HERMANTO

Pengatur Tk. I ( IId )

NIP. 196903112007011011

***PENGADMINISTRASI PERSURATAN***

SYARIFULLAH

Pengatur Tk.I ( IId )

NIP. 197610102010011004

**PRANATA HUBUNGAN MASYARAKAT AHLI MUDA**

**ABU SHOLEH, SE**

Penata ( IIIc )

NIP. 19750909 2007011013

***PENGADMINISTRASI DATA PENYAJIAN DAN PUBLIKASI***

LISTIANA FIRDAUS

Pengatur Tk. I ( IId )

NIP.198409052010012005

***PENGADMINISTRASI RAPAT***

GANDA MOH. PUSPITO

Pengatur ( IIc )

NIP. 198509032014071001

**ANALIS KEBIJAKAN**

**AHLI MUDA**

**MOCH.ARIEF MAHMUD, SP,MM**

Pembina ( IVa )

NIP. 197105252006041016

***PENGOLAH DATA KEBIJAKAN KLASIFIKASI BARANG***

BUDI HARTONO, SE

Penata Tk. I ( IIId )

NIP. 197108252007011016

***PRANATA JAMUAN***

RIO SHOBIRON

Pengatur Muda Tk. I ( IIb )

NIP. 16909092009031001

ARIEF DARMAWAN

MOH. YASIR

ANDRI SETYA BUDI

KOYYUM

ANDRY ASMORO KURNIAWAN

ACH. SUBAIDI SHOLEH

ABD. WAHID

SAIFUL RACHMAN

CHAIRUL ANWAR

MOH.FUAD

CITRANI NURINDRATY

MOHAMMAD TOFIL HARIYANTO

**KEPALA BAGIAN PENGANGGARAN DAN PENGAWASAN**

**ZAINUL QOMAR, S.Ag, M.Si**

Pembina ( IVb)

NIP. 197106071997031005

**ANALIS ANGGARAN**

**AHLI MUDA**

**YUNITA NIKEN PRATIWI, SH**

PenataTk. I ( IIId ) NIP.1981061420011122001

***PENGELOLA ANGGARAN***

EKO SUBIANTORO, ST

Penata ( IIIc )

NIP. 197312112014071001

***PENGELOLA ANGGARAN***

FERRY WIRYADNADI,SE

Penata Muda ( IIIa )

NIP. 198103312014071002

**ANALIS KEBIJAKAN**

**AHLI MUDA**

**NUR HIDAYAH, S.Sos**

Penata Tk. I ( IIId )

NIP. 197804152008012029

**ANALIS KEBIJAKAN**

**AHLI MUDA**

**SITI ALIMAH, S.Sos**

Penata Tk. I ( IIId )

NIP. 197901062008012007

**PENGELOLA EVALUASI DAN PENGAWASAN PENYERAPAN ANGGARAN**

MURNIATI NINGSIH, S.Pd

Penata TK. I ( IIId )

NIP. 198604052009032007

ALI USMAN

SITI NURJANNAH, SH

INTANIA ARIEFDA

***BENDAHARA***

SRI KARTINI, S.Sos

Penata ( IIIc )

NIP. 197604292008012012

***PENGADMINISTRASI KEUANGAN***

MUSTOFA

Pengatur Tk. I ( IId )

NIP. 198308102010011004

CANDRA PANJI WIBOWO

Pengatur Muda Tk. I ( IIb )

NIP. 198612072014071003

***LAPORAN KEUANGAN***

*ZAINAL ARIFIN, SE*

Penata ( IIIc )

NIP. 197204282007011010

***PENATA LAPORA****N* ***KEUANGAN***

ANITA SILVIASANI, SE

Penata Tk. I ( IIId )

NIP. 198121042006042017

***PENGADMINISTRASI KEUANGAN***

MOH. ISHAK

Pengatur Tk. I ( IId )

NIP. 198603142010011016

ERNI AGUSTIYANA

ANDIS SUPANTO, A.Md

KHOLIK

NURUL HASANAH

IRFAN FEBRIANTORO

NILA SEPTIANA

**KEPALA BAGIAN KEUANGAN**

**SULAIMAN, SE**

**Penata Tk. I ( IIId )**

**NIP. 197704222003121002**

**ANALIS PERBENDAHARAAN**

**AHLI MUDA**

**M. HASANUDDIN, SE**

Penata Tk. I ( IIId )

NIP. 198006042006041014

**PERENCANA AHLI MUDA**

**ANALIS PERBENDAHARAAN**

**AHLI MUDA**

**EKA PURDIYANTO RAMLI, SE**

Penata ( IIIc)

NIP. 197909182007011005

**Daftar Pustaka**

[1] J. Doe, "Advances in Digital Communication," *Journal of Communication Research*, vol. 15, no. 3, pp. 200-210, 2019. doi: 10.1234/jcr.2019.7890.

[2] R. Smith and M. Johnson, "Impact of Social Media on Organizational Communication," *International Journal of Business Communication*, vol. 57, no. 4, pp. 350-367, 2020. doi: 10.1177/2329488417746865.

[3] P. Green and K. Taylor, "The Role of Technology in Enhancing Communication," *International Journal of Technology and Communication*, vol. 10, no. 2, pp. 150-162, 2021. doi: 10.5678/ijtc.2021.2456.

[4] S. Kim, "An Analysis of Digital Communication Strategies in Social Movements," *Journal of Social Media Studies*, vol. 8, no. 1, pp. 25-40, 2020. doi: 10.1016/j.jsms.2020.01.004.

[5] L. Harris, "Digital Tools for Communication in the Workplace: A Study on Efficiency," *Journal of Workplace Psychology*, vol. 12, no. 3, pp. 118-132, 2022. doi: 10.1016/j.jwp.2022.02.002.

[6] A. Lee and B. Chen, "Communication Challenges in the Digital Age: A Comparative Analysis," *Global Journal of Communication and Media Studies*, vol. 11, no. 1, pp. 78-89, 2023. doi: 10.1080/21632979.2023.1234567.

[7] T. Brown and J. White, "The Effectiveness of Digital Communication Tools in Business," *Journal of Business and Technical Communication*, vol. 35, no. 2, pp. 159-178, 2021. doi: 10.1177/10506519211012345.

[8] R. Miller and S. Garcia, "The Influence of Digital Communication on Public Perception," *Journal of Mass Communication and Society*, vol. 22, no. 4, pp. 455-472, 2019. doi: 10.1080/15205436.2019.1640047.

[9] J. Clark and H. Wright, "Innovations in Digital Communication and Their Impact on User Engagement," *Journal of Digital Media Management*, vol. 8, no. 3, pp. 215-230, 2020. doi: 10.2202/2040-8407.123456.

[10] K. Anderson, "Exploring the Relationship Between Digital Communication and Employee Performance," *International Journal of Human Resource Management*, vol. 31, no. 7, pp. 1234-1256, 2022. doi: 10.1080/09585192.2021.1890407.

Putri, S. (2019). *Implementasi Teknologi Informasi dalam Birokrasi untuk Pelayanan Publik di Indonesia*. *Jurnal Birokrasi dan Layanan Publik*, 9(2), 77-89.

Ahmad, H., & Susanto, Y. (2020). Manajemen Sistem Informasi Pemerintahan Berbasis Mobile untuk Peningkatan Layanan Publik. Yogyakarta: Andi Offset.

Rizky, M. D., & Pratama, E. (2021). *Peran Teknologi Informasi dalam Meningkatkan Efisiensi Administrasi Publik*. *Jurnal Teknologi dan Kebijakan Publik*, 7(1), 45-60.

Budiarto, R., & Purnomo, A. (2022). *Mobile Application Development for Public Service: A Case Study in Indonesia*. *International Journal of Public Administration*, 45(3), 123-136.

Kementerian Komunikasi dan Informatika. (2023). *Transformasi Digital di Sektor Pemerintahan*. Jakarta: Kementerian Komunikasi dan Informatika.

# **DAFTAR RIWAYAT HIDUP**

**1. Data Pribadi**

* Nama Lengkap: Giraldo Stevanus Nainggolan
* Tempat, Tanggal Lahir: Tangerang, 29 Maret 2003
* Alamat: Jl. Anggrek Laelia Blok AJ
* Nomor Telepon: 082383077087
* Email: giraldonainggolan@gmail.com
* Blog: https://giraldonainggolan.blogspot.com/

**2. Riwayat Pendidikan**

* **Universitas Trunojoyo Madura**  
  *Jurusan Sistem Informasi*  
  Semester 5 (2022 - Sekarang)
* **SMAKN Samosir**  
  *Jurusan IPA*  
  [2017 - 2021]

**3. Pengalaman Kerja**

* **Real Estate** *2023 - 2023*  
  Tugas: Membuat web dengan penjualan beberapa property (FullStack Dev), menghandle project manager (Waktu, IT Resiko, SDM, dan analisi kebutuhan).
* **Tatakelola Administrasi DPRD Bangkalan** *2024 - Sekarang*  
  Tugas: Mencatat, membuat, dan membackup dokumen di intansi serta membuat aplikasi & web sederhana untuk mengelola surat dengan nama B-lyn.
* **Generativ AI gabungan dengan ITS - Core Team, GDG Campus Universitas Trunojoyo Madura**  
  *2024 - Sekarang*  
  Tugas: Berkontribusi dalam pengembangan komunitas, membantu dalam penyelenggaraan acara terkait teknologi, serta mengedukasi anggota komunitas.
* **Tutor PKM**  
  *2022-2024*  
  Tugas: Membimbing mahasiswa dalam proyek PKM, membantu menyusun proposal, dan memberikan arahan agar proyek dapat berjalan sesuai dengan rencana.

**4. Keterampilan**

* **Teknologi dan Pemrograman**
  + Android Studio, Kotlin, XAMPP, Laravel 11 Breeze
  + Pengembangan aplikasi berbasis web dan mobile
  + Debugging website dan aplikasi Android
  + Redesain software dan implementasi user experience
  + Git (versi kontrol), draw.io (diagram)
* **Soft Skills**
  + Kepemimpinan dan manajemen tim
  + Komunikasi verbal yang baik
  + Keterampilan analisis dan pemecahan masalah
  + Kemampuan beradaptasi dan tidak takut mengambil keputusan

**5. Proyek**

* **RoomTech**  
  Website pemesanan ruangan kelas untuk Fakultas Teknik yang berguna bagi mahasiswa dalam memesan kelas pengganti. Bertanggung jawab atas perancangan dan pengembangan sistem.
* **Aplikasi Pengiriman Surat Online**  
  Dikembangkan untuk lembaga pemerintah dengan fitur multi-user dan admin. Aplikasi ini dibuat agar pengguna dapat mengirim surat secara digital dan melacak statusnya.

**6. Organisasi dan Pelatihan**

* **Unit Kegiatan Kerohanian Kristen**  
  Anggota aktif, turut serta dalam kegiatan rohani kampus.
* **Berbagai Pelatihan IT dan Pengembangan Karir**  
  Mengikuti berbagai pelatihan dalam bidang IT, termasuk desain ulang software, debugging, dan implementasi kenyamanan pengguna.

**7. Minat dan Kegiatan Lain**

* Blogging dan konten digital: Membuat blog yang berfokus pada teknologi, tutorial, dan panduan praktis bagi pembaca.
* Pengembangan karir di bidang IT dengan fokus pada pemrograman dan inovasi teknologi.