TUGAS AKHIR

METODE *CLEAN ARCHITECTURE* BERDASARKAN PRINSIP DESAIN *SOLID* UNTUK ANALISIS APLIKASI PENCATATAN ABSENSI DARING (PANDA)

Oleh:

Josua Manalu 2024240117

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER DAN REKAYASA
UNIVERSITAS MULTI DATA PALEMBANG
PALEMBANG

2024

Fakultas Ilmu Komputer dan Rekayasa
Universitas Multi Data Palembang

Program Studi Sistem Informasi
Tugas Akhir Sarjana Komputer
Semester Gasal Tahun 2023/2024

METODE *CLEAN ARCHITECTURE* BERDASARKAN PRINSIP DESAIN *SOLID* UNTUK ANALISIS APLIKASI PENCATATAN ABSENSI DARING (PANDA)

Josua Manalu 2024240117

Abstrak

Aplikasi PANDA merupakan aplikasi berbasis *android* dengan bahasa *Java* yang dimiliki oleh Universitas Multi Data Palembang untuk melakukan absensi dosen dan karyawan. Aplikasi PANDA menjadi objek analisis berdasarkan prinsip desain *SOLID* dan *Clean Architecture*. Tujuan dari penelitian ini adalah menganalisis sejauh mana aplikasi PANDA mematuhi kaidah *Clean Architecture* dan prinsip desain *SOLID* serta melakukan *refactoring* pada *source code* aplikasi PANDA berdasarkan kaidah dan prinsip desain tersebut. *Refactoring* dilakukan pada tujuh *class* yang masih tidak sesuai dengan prinsip *Single Responsibility* dengan memindahkan *function-function* yang memiliki tanggung jawab yang sama ke dalam suatu *class*. Prinsip *Interface Segregation* diimplementasikan dengan menghapus *function* implementasi yang tidak diperlukan. Empat *class* yang tidak mematuhi prinsip *Dependency Inversion* disesuaikan dengan membuat ketergantungan pada class abstraksi dan menggunakan injeksi dependensi dengan *Hilt*. Konsep *Clean Architecture* juga telah diterapkan dengan membuat *domain layer* yang berisi *entities* yang telah dimodifikasi agar tidak terpengaruh oleh *framework* luar dan *use case* yang berisi *business logic* aplikasi sehingga *presentation layer* tidak bergantung secara langsung pada *data layer* melainkan bergantung kepada *domain layer*. Penerapan *Clean Architecture* dan prinsip desain *SOLID* menghasilkan jumlah *source code* yang lebih sedikit dan rekomendasi kode dari penelitian ini diharapkan dapat memudahkan *developer* dalam pemeliharaan dan pengembangan aplikasi PANDA ke depan.

Kata kunci: Aplikasi PANDA, *Android*, Analisis, *Clean Architecture*, *Java*, *Refactoring*, *SOLID*, *Source code*


UNIVERSITAS
MDP

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Pengembangan perangkat lunak tidak hanya mengenai fungsionalitas semata, melainkan juga tentang perencanaan arsitektur yang kokoh. Arsitektur aplikasi memegang peranan penting untuk memastikan suatu aplikasi stabil, mudah untuk diuji serta dipelihara (*Developers Android*, 2023). Desain arsitektur yang baik menghasilkan sistem yang mudah untuk dirawat, fleksibel dan mudah diuji. Pemilihan arsitektur perangkat lunak menjadi krusial karena mempengaruhi kehandalan, fleksibilitas, dan kemudahan pemeliharaan aplikasi. Pendekatan arsitektur yang menjadi kajian adalah *Clean Architecture*. Penggunaan *Clean Architecture* bertujuan untuk mendukung siklus hidup sistem. Arsitektur yang baik akan memberikan kode yang mudah dimengerti oleh orang lain, mudah untuk dikembangkan, mudah untuk dipelihara dan mudah untuk di-*deploy* (Martin, 2017). Tujuan akhirnya adalah meminimalkan biaya selama siklus hidup sistem dan meningkatkan produktivitas *programmer*.

Clean Architecture adalah pendekatan desain arsitektur *software* yang menitikberatkan pada pemisahan tanggung jawab. *Clean Architecture* membagi *software* menjadi lapisan-lapisan, dengan setiap lapisan memiliki tanggung jawab yang berbeda (Martin, 2017). Menurut (Martin, 2017), *Clean Architecture* adalah

arsitektur yang memisahkan perhatian menjadi beberapa lapisan dengan mengikuti aturan "*Dependency Rule*", yang menyatakan bahwa ketergantungan setiap lapisan harus menuju ke dalam dan menciptakan karakteristik *software* yang independen terkait *framework*, *UI*, *database*, *business rules* serta memudahkan dalam pengujian. Arsitektur yang berkualitas dimulai dengan kode yang baik. Kode yang baik diperlukan sebagai fondasi kokoh untuk membangun arsitektur yang baik. Namun, kode yang buruk dapat mengakibatkan kekacauan pada arsitektur perangkat lunak. Prinsip desain *SOLID* hadir sebagai pedoman untuk membangun arsitektur yang handal.

Desain *SOLID* menjelaskan bagaimana mengorganisasikan *function* dan struktur data ke dalam suatu *class* dan interkoneksi antar *class* tersebut. Prinsip desain *SOLID* memastikan bahwa arsitektur yang dibangun memiliki fondasi yang kokoh. Desain *SOLID* memberikan pedoman dalam mengatur *class-class* dan hubungan antar *class-class* tersebut. Menurut (Martin, 2017), prinsip desain SOLID menyatakan tentang bagaimana mengatur struktur data dan *function* ke dalam suatu *class*, serta bagaimana *class* tersebut saling terhubung dengan karakteristik seperti mudah diubah, mudah dipahami, dan menjadi dasar komponen yang dapat digunakan di dalam *software*.

Kaidah *Clean Architecture* dan prinsip desain *SOLID* merupakan dua hal yang penting dalam pengembangan aplikasi. Keduanya berperan dalam menciptakan aplikasi yang mudah dipelihara, mudah dipahami dan mudah diuji. Aplikasi yang

akan dianalisis sesuai dengan kaidah *Clean Architecture* dan prinsip desain *SOLID* adalah aplikasi yang dimiliki oleh Universitas Multi Data Palembang.

Universitas Multi Data Palembang memiliki aplikasi yang bernama Aplikasi Pencatatan Absensi Daring (PANDA), yang berbasis *mobile* dan digunakan untuk melakukan pencatatan absensi dosen dan karyawan. Aplikasi ini dikembangkan untuk pengguna ponsel *Android* dengan menggunakan bahasa pemrograman *Java*. Alur penggunaan aplikasi dimulai dengan melakukan *login* menggunakan *username* dan *password* yang telah diberikan oleh tim Unit Pelaksana Teknis Sistem Informasi (UPT-SI). Selanjutnya, pengguna memilih opsi absensi datang sesuai dengan waktu kehadiran yang tersedia dan melakukan selfie wajah. Absensi harus dilakukan pada jam datang dan jam pulang, serta harus berada dalam radius lokasi Universitas Multi Data Palembang dan MDP IT Store.

Aplikasi PANDA dipilih menjadi objek analisis karena aplikasi ini masih aktif digunakan dan dipelihara di lingkungan Universitas Multi Data Palembang serta berdasarkan pengamatan awal terhadap *source code* aplikasi PANDA, ditemukan bahwa aplikasi belum menerapkan *Clean Architecture* sehingga belum ada pembagian *layer*, tidak terdapat abstraksi sehingga setiap *class* masih bergantung pada implementasi bukan abstraksi, dan tidak sesuai dengan prinsip desain *SOLID*. Hal ini menjadi alasan mengapa aplikasi PANDA dipilih menjadi objek analisis.

Berdasarkan penjelasan sebelumnya, akan dilakukan analisis terhadap aplikasi PANDA dengan menerapkan *Clean Architecture* serta prinsip desain *SOLID* untuk menghasilkan aplikasi yang mematuhi prinsip *Clean Architecture* dan desain

SOLID. Oleh karena itu, diusulkan judul penelitian **"Metode *Clean Architecture* berdasarkan Prinsip Desain *SOLID* untuk Analisis Aplikasi Pencatatan Absensi Daring (PANDA)"**.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, dirumuskan permasalahan yaitu bagaimana penerapan *Clean Architecture* dan prinsip desain *SOLID* pada aplikasi PANDA.

1.3 Ruang Lingkup

Dalam penelitian ini ditentukan ruang lingkup atau batasan masalah sebagai berikut.

1. Aplikasi Pencatatan Absensi Daring (PANDA) akan di-*refactor* menggunakan bahasa pemrograman *Java* dan *Android Studio* versi *Giraffe* sebagai *Integrated Development Environment* (*IDE*).
2. Aplikasi akan di-*refactor* sesuai dengan prinsip *Clean Architecture* dan desain *SOLID* berdasarkan buku *Clean Architecture: A Craftsman's Guide to Software Structure and Design* dari (Martin, 2017) dan *Clean Android Architecture* dari (Dumbravan, 2022).
3. Fitur yang di-*refactor* banyak dilakukan di *Presentation Layer* karena lapisan ini merupakan lapisan yang mengelola logika terkait tampilan antarmuka seperti *View* yang direpresentasikan sebagai *Activity*/*Fragment* (Dumbravan, 2022) dan

refactoring juga dilakukan pada *Data Layer* yang merupakan pemanggilan data serta *class HomeFragment* dan *HomeViewModel*. Fitur dari aplikasi PANDA yang akan di-*refactor* yaitu:

a. Fitur absensi yang meliputi *function saveImage*, *getLatLngFromJSONString*, *galleryAddPic*, *previewCapturedImage*, *rotateImage*, *drawFeatureFace*, *showFaceInfo*, *getOutputMediaFile*, *setPic*, dan *calculateInSampleSize* yang terdapat pada *class* AbsenActivity. Fitur absensi yang dipilih melibatkan manipulasi gambar, pengambilan lokasi, dan logika terkait wajah yang bertujuan untuk meningkatkan kejelasan dan pemeliharaan kode terkait manipulasi gambar dan pengelolaan lokasi, sehingga memudahkan pengembangan dan perubahan fitur ini di masa mendatang.

b. Fitur *login* yang meliputi *function loginUser*, *signInWithPhoneAuthCredential*, dan *showResultDialog* yang terdapat pada *class LoginActivity*. Fitur *login* yang dipilih melibatkan tanggung jawab autentikasi pengguna dengan tujuan untuk memisahkan logika dan memudahkan penggantian autentikasi di masa mendatang tanpa mengganggu tampilan antarmuka.

c. *Class ServiceRepository* yang meliputi *function getAbsenDataTerlambat*, *getAbsenToday* dan *getPilihanShift*. *Function* yang dipilih berkaitan dengan pemanggilan data untuk absensi yang bertujuan untuk memisahkan *function* sesuai tanggung jawabnya.

d. *Class MainActivity* yang meliputi *function locationEnabled*, *addPermission*, dan *getInboxCount*. *Function* yang dipilih melibatkan izin dan pengelolaan

lokasi dengan tujuan memudahkan perubahan terkait izin dan lokasi di masa mendatang tanpa mengganggu tampilan antarmuka.

e. *Class HomeFragment* yang meliputi *function updateRegId*, *onItemShiftClick*, dan *showPilihanShiftDialog*. *Function* yang dipilih terkait pilihan *shift* dan registrasi *ID* dengan tujuan memudahkan perubahan dan pengembangan yang melibatkan pilihan *shift* dan registrasi *ID* di masa mendatang tanpa mengganggu tampilan antarmuka.

f. *Class HomeViewModel* yang meliputi *function getAbsenToday* dan *getPilihanShift*. *Function* yang dipilih berkaitan proses absensi dengan tujuan memudahkan perubahan di masa mendatang.

g. *Class APIService* yang meliputi *function updateProfile*, *login*, *insertAbsenShift*, *updateRegId*, *getAbsenToday*, *getVersion*, *getAbsenByTgl*, *getPilihanShift*, dan *getAbsenTerlambat*. *Function* yang dipilih terkait absensi dan *login* dengan tujuan memisahkan *function* sesuai dengan tanggung jawabnya.

h. *Class Utilities* yang meliputi *function convertToDDMMMyyyy*, *hasAppPermission*, *getDateTimeWithModernFormat*, *convertToDDMMyyyy*, *convertToHHmmss*, *convertToDDMMyyyyHHmmss*, *convertToyyyyMMDD*, *getTanggalBulanTahun*, dan *distFrom*. *Function* yang dipilih berkaitan dengan konversi tanggal, dan pengelolaan izin dengan tujuan memisahkan *function* sesuai dengan tanggung jawabnya.

1.4 Tujuan dan Manfaat

Tujuan dari skripsi ini adalah sebagai berikut.

1. Menganalisis sejauh mana aplikasi PANDA mematuhi kaidah *Clean Architecture* dan prinsip desain *SOLID.*
2. Melakukan *refactoring* pada *source code* aplikasi PANDA berdasarkan *Clean Architecture* dengan prinsip desain *SOLID.*

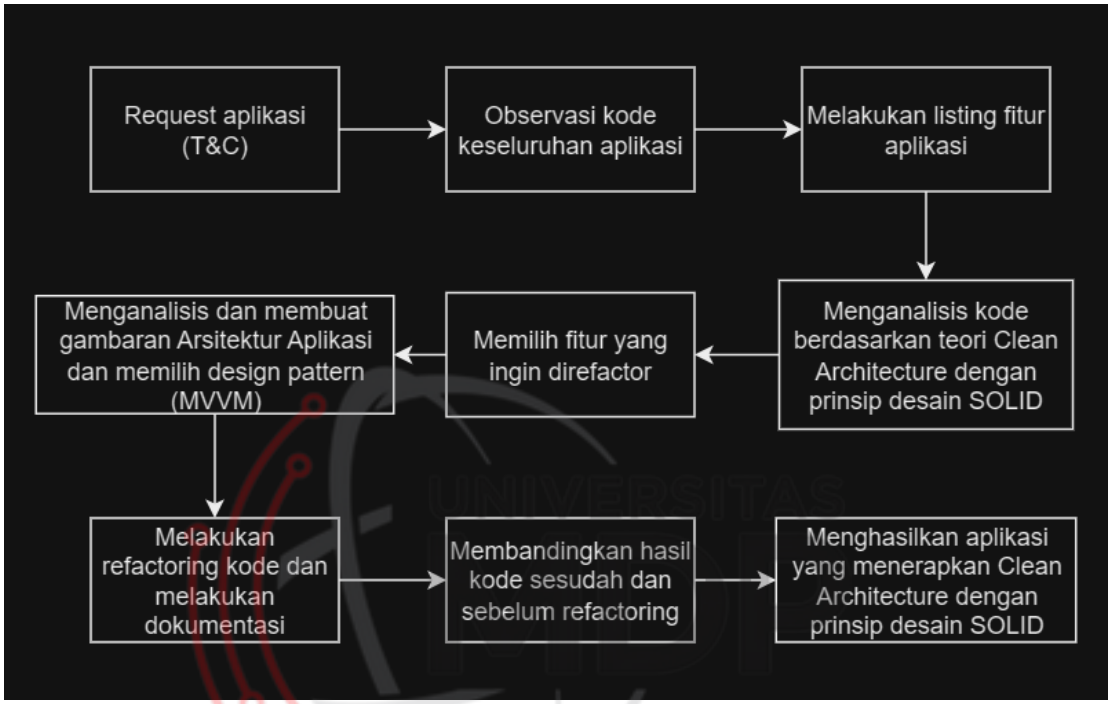
Manfaat dari skripsi ini adalah sebagai berikut.

1. Menghasilkan rekomendasi *source code* yang menerapkan kaidah *Clean Architecture* dengan prinsip desain *SOLID.*
2. Menghasilkan *prototype source code* aplikasi PANDA yang sudah menerapkan *Clean Architecture* dan prinsip desain *SOLID*.
3. Menghasilkan aplikasi yang sudah menerapkan kaidah *Clean Architecture* dan desain *SOLID* sehingga mudah untuk dipelihara oleh *developer* Universitas Multi Data Palembang.

1.5 Metodologi

Metodologi yang digunakan dalam penelitian ini terdiri dari serangkaian tahapan terstruktur yang dirancang khusus untuk memenuhi kebutuhan analisis aplikasi PANDA. Metodologi ini terdiri dari 9 tahapan yang dimulai dari meminta akses terhadap *source code* aplikasi hingga melakukan *refactoring* dan menghasilkan

aplikasi yang menerapkan *Clean Architecture* dengan prinsip desain *SOLID*. Berikut adalah penjelasan setiap tahapan metodologi yang digunakan dalam penelitian ini.



Gambar 1.1 Metodologi Penelitian

1. *Request* aplikasi (*Term* & *Conditions*)

Tahap ini bertujuan untuk meminta *source code* aplikasi dari pihak *stakeholder* untuk dilakukan analisis dengan syarat dan ketentuan yang harus disetujui dan dipatuhi untuk tidak menyebarluaskan atau membocorkan *source code* aplikasi ke pihak yang tidak bersangkutan.

2. Observasi kode keseluruhan aplikasi

Tahap ini bertujuan untuk mengamati keseluruhan kode aplikasi yang akan dianalisis untuk mengetahui bagaimana alur aplikasi berjalan dan memahami struktur dan komponen-komponen utama yang akan dianalisis lebih lanjut.

3. Melakukan *listing* fitur aplikasi

Tahap ini bertujuan untuk membuat daftar fitur-fitur yang ada dalam aplikasi untuk dilakukan analisis.

4. Menganalisis kode berdasarkan teori *Clean Architecture* dengan prinsip desain *SOLID*

Tahap ini berfokus pada proses analisis kode aplikasi untuk mengidentifikasi kode yang perlu diperbaiki sesuai dengan prinsip *Clean Architecture* dan desain *SOLID*.

5. Memilih fitur yang ingin di-*refactor*

Tahap ini bertujuan untuk menghasilkan daftar fitur yang akan di-*refactor* berdasarkan analisis yang dilakukan sebelumnya. Fitur yang akan di-*refactor* dipilih berdasarkan cakupan kode yang ada pada *class* serta kesalahan yang diperoleh setelah melakukan identifikasi kode di aplikasi PANDA. Fitur yang dipilih akan di-*refactor* sesuai dengan prinsip *Clean Architecture* dan desain *SOLID*.

6. Menganalisis dan membuat gambaran arsitektur aplikasi dan memilih *design pattern* (MVVM)

Tahap ini bertujuan untuk menghasilkan gambaran arsitektur aplikasi beserta memilih *design pattern* yang akan digunakan dalam proses *refactor* aplikasi yaitu *MVVM* (*Model, View, ViewModel*).

7. Melakukan *refactoring* kode dan melakukan dokumentasi

Tahap ini berfokus pada *refactoring* kode fitur aplikasi yang telah dipilih sebelumnya. Proses *refactoring* akan dilakukan dengan menerapkan prinsip *Clean Architecture* dan desain *SOLID*.

8. Membandingkan hasil kode sesudah dan sebelum *refactoring*

Tahap ini bertujuan untuk membandingkan kode sebelum dan sesudah *refactoring* yang telah menerapkan prinsip *Clean Architecture* dan desain *SOLID*. Hasil perbandingan akan menunjukkan perbedaan kode dan struktur *class* yang telah menerapkan dan belum menerapkan *Clean Architecture* dan prinsip desain *SOLID*.

9. Menghasilkan aplikasi yang menerapkan *Clean Architecture* dengan prinsip desain *SOLID*

Tahap ini merupakan tahapan akhir yang menghasilkan *source code* aplikasi yang telah mengimplementasikan *Clean Architecture* dan prinsip desain *SOLID*.

1.6 Sistematika Penulisan

Proposal tugas akhir ini memiliki tiga bab yang disusun secara sistematis. Berikut adalah sistematika penulisan dalam proposal tugas akhir.

BAB 1 PENDAHULUAN

Bab pertama yaitu pendahuluan membahas tentang latar belakang, permasalahan yang diangkat, tujuan dan manfaat, ruang lingkup, metodologi penelitian, dan sistematika penulisan.

BAB 2 LANDASAN TEORI

Bab kedua yaitu landasan teori membahas tinjauan pustaka yang digunakan dalam penelitian serta penelitian terdahulu sebagai referensi.

BAB 3 ANALISIS

Bab ketiga yaitu analisis membahas tentang analisis yang dilakukan pada penelitian.

BAB 4 PEMBAHASAN

Bab keempat yaitu pembahasan membahas tentang hasil dari *refactoring code* terhadap analisis yang telah dilakukan.

BAB 5 PENUTUP

Bab kelima yaitu penutup membahas tentang kesimpulan dari penelitian dan saran untuk pengembangan ke depan.


UNIVERSITAS
MDP

DAFTAR PUSTAKA

Aflah Taqiu Sondha, Umi Sa'adah, Fadilah Fahrul Hardiansyah, & Maulidan Bagus Afridian Rasyid. (2020). Framework dan Code Generator Pengembangan Aplikasi Android dengan Menerapkan Prinsip Clean Architecture. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, *9*(4), 327–335. https://doi.org/10.22146/jnteti.v9i4.572

Anhar, F. F., & Anggraeny, F. T. (2022). *Implementasi Clean Architecture MVVM Dan Repository Pattern Untuk Pengembangan Aplikasi Android Jual Beli Barang Bekas "SECONDHAND." XVII*(2), 19–24.

Badrudduja, M. H., & Putra, R. E. (2022). Penerapan Clean Architecture pada Aplikasi Pemesanan Makanan menggunakan Metode Slope One Algorithm. *Journal of Informatics and Computer Science (JINACS)*, *3*(04), 506–514. https://doi.org/10.26740/jinacs.v3n04.p506-514

Developers Android. (2023). https://developer.android.com/topic/architecture/

Dumbravan, A. (2022). *Clean Android Architecture*. Packt Publishing Ltd.

Fajri, A. R. (2022). Penerapan Design Pattern MVVM dan Clean Architecture pada Pengembangan Aplikasi Android (Studi Kasus: Aplikasi Agree). *Official Scientific Journals of Universitas Islam Indonesia*, 64.

Guardsquare. (2016). *Guard Square*. https://www.guardsquare.com/proguard

Makarenko, V., Olshevska, O., & Kornienko, Y. (2017). An Architectural Approach for Quality Improving of Android Applications Development Which Implemented To Communication Application for Mechatronics Robot Laboratory Onaft. *Automation of technological and business processes*, *9*(3), 8–13. https://doi.org/10.15673/atbp.v9i3.714

Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design* (1st ed.).

Muchlison, I. D., Kharisma, A. P., & Arwani, I. (2022). Pengembangan Aplikasi Perangkat Bergerak Sistem Informasi Event di bidang Teknologi Informasi berbasis Android. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *6*(1), 282–291.

Prawira, A. F., Putra, W. H. N., & Purnomo, W. (2022). Pengembangan Aplikasi E-Commerce Anggrek berbasis Android menggunakan Clean Architecture (Studi Kasus : PT . Java Indo Arjuna). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *6*(8), 3606–3612.

Rizkifar, M. A., Ilmu, F., Univesitas, T., Batu, T. B., Barat, J., Ilmu, F., Univesitas, T., Batu, T. B., Barat, J., Rosmiati, M., Ilmu, F., Univesitas, T., Batu, T. B., & Barat, J. (2021). *APLIKASI UNTUK PEMASARAN DAN PENJUALAN PRODUK DI BAKER'S CORNER BERBASIS ANDROID*. *7*(5), 1923–1931.

Rumapea, D. R., Tandrian, K., Kurniawan, R. A. T., Salsabila, M., Ngoh, D., Marwies, K., Aqid, B. M., & Azurat, A. (2023). Design and Development of EcoSense: Android-Based Incentivized Environmental Campaign App. *Jurnal Ilmu Komputer dan Informasi (Journal of Computer Science and Information)*, *16*(2), 89–101. https://doi.org/http://dx.doi.org/10.21609/jiki.v16i2.1144

Saifulloh, T., Kharisma, A. P., & Brata, D. W. (2023). *Pengembangan Aplikasi Perangkat Bergerak Pencarian Partner Lomba berbasis Android menggunakan Clean Architecture*. *7*(4), 1549–1559.

Subagio, A. W., & Muttaqin, F. (2022). Penerapan Clean Architecture pada Pengembangan Sistem Payment Point Online Bank. *Jurnal Teknologi Elektro dan Kejuruan*, *32*(2), 324–333.

Yoga, I. K., Putra, D., Kharisma, A. P., & Arwani, I. (2021). Pengembangan Aplikasi Berbasis Android Untuk Meningkatkan Kepatuhan Pasien Gagal Jantung Dalam Merawat Diri Di Rumah. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *5*(7), 2976–2985.