Penerapan Prinsip *Clean Architecture* pada Aplikasi Android Kesehatan Mental Menggunakan Kotlin

Kadek Dhiva Tiradika[1], Ni Luh Ratniasih[2], Ni Made Dwijayani[3]
[1,2,3]Sistem Komputer, Fakultas Informatika dan Komputer
[1,2,3]Institut Teknologi dan Bisnis STIKOM Bali
Denpasar, Indonesia
e-mail: [1]dhivatiradika@gmail.com, [2]ratni@stikom-bali.ac.id, [3]nimade_dwijayani@stikom-bali.ac.id

Diajukan: 6 Oktober 2023; Direvisi: 3 April 2024; Diterima: 5 April 2024

Abstrak

Kesehatan mental merupakan suatu aspek yang vital dari kehidupan seorang manusia. Di Indonesia lebih dari 19 juta orang berusia di atas 15 tahun mengalami gangguan mental dan emosional, dan lebih dari 12 juta orang berusia di atas 15 tahun mengalami depresi. Menurut Mental Health America, salah satu cara untuk mengatasi gangguan mental adalah dengan membuat kelompok pendukung atau support group. Menemukan support group yang tepat sesuai masalah yang dialami penderita gangguan jiwa kerap memakan waktu dan cenderung sulit. Platform digital menjadi solusi dari masalah tersebut, karena penderita gangguan mental dan caregiver dapat mengakses support group dari mana saja dan kapan saja. Berdasarkan masalah tersebut dibutuhkan aplikasi Android kesehatan mental yang bertujuan untuk memfasilitasi penyelenggaraan support group. Aplikasi android kesehatan mental dirancang menggunakan Unified Modeling Language (UML) dan dikembangkan menggunakan bahasa pemograman Kotlin dengan menerapkan prinsip Clean Architecture. Hasil penelitian berupa sebuah aplikasi android support group untuk penderita gangguan mental dan caregiver dimana pengujian aplikasi menggunakan metode blackbox testing menghasilkan hasil yang sudah sesuai.

Kata kunci: *Kesehatan Mental, Support Group, Android, Kotlin, Clean Architecture.*

Abstract

Mental health is a vital aspect of a human's life. In Indonesia, more than 19 million people over the age of 15 had mental and emotional disorders, and more than 12 million people over the age of 15 had depression. According to Mental Health America, one of several way to deal with mental disorders is to create support groups. Finding the right support group that suit to the problems that people with mental disorders had often takes time and tends to be difficult. Digital platforms are the solution of this problem, because people with mental disorders and caregivers can access support groups from anywhere and at any time. Based on these problems, Android application of mental health is needed which aims to facilitate the implementation of support groups. The mental health Android application was designed using the Unified Modeling Language (UML) and developed using the Kotlin programming language by applying Clean Architecture principles. The results of the research are an Android support group application for people with mental disorders and caregivers where testing the application using the blackbox testing method produces appropriate results.

Keywords*: Mental health, Support Group, Android, Kotlin, Clean Architecture.*

1. Pendahuluan

Kesehatan mental merupakan suatu aspek yang vital dari kehidupan seorang manusia. Menurut *WHO*, kesehatan mental merupakan kondisi dari kesejahteraan yang disadari individu, yang di dalamnya terdapat kemampuan-kemampuan untuk mengelola stres kehidupan yang wajar, untuk bekerja secara produktif dan menghasilkan, serta berperan serta di komunitasnya[1], [2] . Di Indonesia lebih dari 19 juta orang berusia di atas 15 tahun mengalami gangguan mental dan emosional, dan lebih dari 12 juta orang berusia di atas 15 tahun mengalami depresi. Pemerintah Indonesia telah mengatasi hal tersebut dengan menyediakan puskesmas dan rumah sakit jiwa di setiap provinsi. Namun 31,2% puskesmas dan 6 provinsi di Indonesia belum memiliki rumah sakit jiwa dan perawatan kesehatan jiwa. Menurut *Mental Health*

America, salah satu cara untuk mengatasi gangguan mental adalah dengan membuat kelompok pendukung[3].

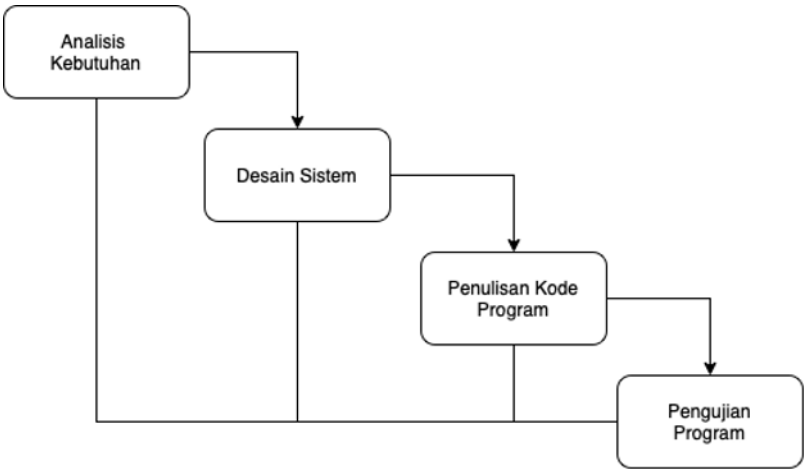
Support group atau kelompok pendukung merupakan bagian dari terapi kelompok. Terapi kelompok terdiri dari lima hingga sepuluh individu yang bertemu untuk menyelesaikan masalah tertentu. Anggota kelompok didorong untuk memberikan umpan balik terhadap anggota kelompok lainnya. Umpan balik dapat berbentuk ekspresi perasaan ataupun respons perilaku terhadap anggota kelompok lain. Interaksi antar anggota kelompok terjadi dalam bentuk saling memberikan dorongan dan kesempatan kepada masing-masing anggota untuk mencoba cara baru dalam berinteraksi dengan orang lain.

Support group terbuka untuk siapa saja tanpa terkecuali, idealnya *support group* dibentuk berdasarkan dengan inti masalah yang sama dengan dipimpin seseorang dalam memediasi diskusi dalam kelompok tersebut atau biasa disebut *caregiver*. *Caregiver* merupakan orang yang bertugas memimpin dan memediasi sebuah *support group*, *caregiver* juga membantu memberikan saran kepada anggota *support group* lainnya sesuai dengan kapasitasnya. Menjadi *caregiver* memerlukan pemahan terkait topik yang dibahas pada *support group*, oleh sebab itu *caregiver* seringkali adalah seorang penyintas atau orang yang telah berhasil menangani masalah yang dibahas pada sebuah *support group*.

Masalah yang muncul dari *support group* yang diadakan secara luring serta berdasarkan komunitas dan wilayah adalah terbatasnya anggota yang dapat bergabung ke dalam *support group* tersebut karena perbedaan wilayah, selain itu informasi mengenai *support group* juga terbatas pada komunitas dan sulit dijangkau orang-orang di luar komunitas tersebut[4]. Untuk menyediakan *support group* secara virtual yang tidak terbatas pada tempat dan waktu, maka dikembangkan aplikasi Android kesehatan mental dengan menerapkan prinsip *Clean Architecture*[5].

2. Metode Penelitian

Dalam pengembangan aplikasi Android kesehatan mental menggunakan Kotlin dengan menerapkan prinsip *clean architecture* penulis menggunakan metode yang bersifat linear dari satu tahap ke tahap yang lainnya yang dimulai dari tahap perencanaan sampai dengan tahap pengujian[6]. Tahapan dilakukan secara runut, tahapan berikutnya tidak akan dilaksanakan sebelum tahapan sebelumnya selesai dilaksanakan. Tahapan pengembangan sistem ini dapat dilihat pada Gambar 1.



Gambar 1. Tahapan Pengembangan Sistem.

2.1 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk mengetahui permasalahan yang ada sesuai dengan data dan informasi yang didapat saat pengumpulan data serta menjadikan data dan informasi tersebut dasar dalam pembangunan aplikasi Android kesehatan mental. Hasil analisis kebutuhan akan menjadi acuan dalam pengembangan fitur aplikasi sesuai dengan kebutuhan pengguna aplikasi.

2.2 Desain Sistem

Setelah dilakukannya analisis kebutuhan dan menghasilkan rumusan masalah serta kebutuhan pengguna, maka hasil tersebut diimplementasikan ke dalam desain sistem. Desain sistem mencakup konfigurasi dari komponen perangkat lunak dan perangkat keras dari suatu sistem, desain sistem juga mengambarkan bagaimana suatu sistem dibentuk. Proses perancangan sistem pada aplikasi Android

kesehatan mental ini terdiri dari perancangan *Unified Modelling Language (UML)* dan perancangan antaramuka. Rancangan ini akan menjadi dasar serta acuan dalam pembuatan aplikasi Android kesehatan mental[7].

2.3 Penulisan Kode Program

Penulisan kode program adalah tahapan yang dilakuakan setelah desain sistem selesai. Pada tahap ini aplikasi Android kesehatan mental mulai dibuat dengan melakukan penulisan kode program yang sesuai dengan desain sistem yang telah dibuat sebelumnya. Dalam pembuatan aplikasi Android kesehatan mental ini, penulis menggunakan bahasa pemograman Kotlin dengan IDE Android Studio. Untuk meningkatkan *maintainability* dari aplikasi Android kesehatan mental ini, penulis menerapkan prinsip *clean architecture* dalam penulisan kode program.

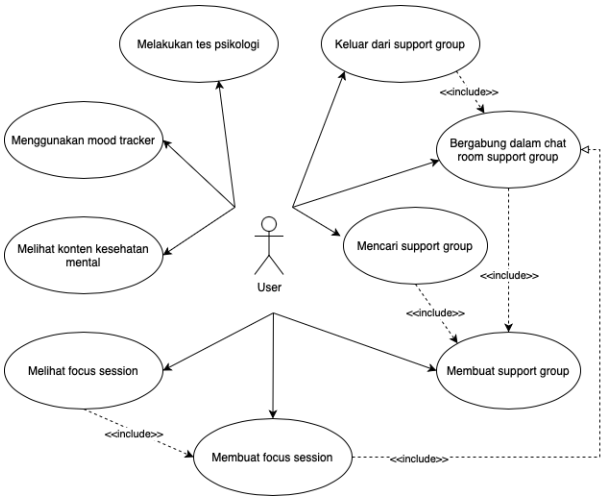
2.4 Pengujian Program

Pengujian program dilakukan setelah tahapan penulisan kode program selesai dilaksanakan dan program berhasil dijalankan. Pengujian program dilakukan untuk memastikan bahwa program sudah berjalan sesuai kebutuhan serta sesuai dengan rancangan yang telah dibuat. Pengujian yang dilakukan pada aplikasi Android kesehatan mental ini adalah *blackbox testing* dan pengujian langsung oleh pengguna. *Blackbox testing* dilakukan dengan cara membandingkan masukan dan keluaran dari aplikasi, dengan masukan yang sudah ditentukan apakah aplikasi atau perangkat lunak dapat memberikan keluaran yang sesuai dan tepat, jika aplikasi dapat memberikan keluaran yang tepat maka pengujian dinyatakan berhasil, sebaliknya jika aplikasi tidak memberikan keluaran yang tepat maka pengujian dinyatakan gagal. Setelah *blackbox testing* berhasil, dilanjutkan dengan pengujian pengguna, dimana aplikasi yang akan diuji diberikan langsung ke pengguna dan melakukan survei dengan kuesioner untuk memperoleh informasi apakah aplikasi dapat memenuhi tujuan yang sudah ditentukan.

3. Hasil dan Pembahasan

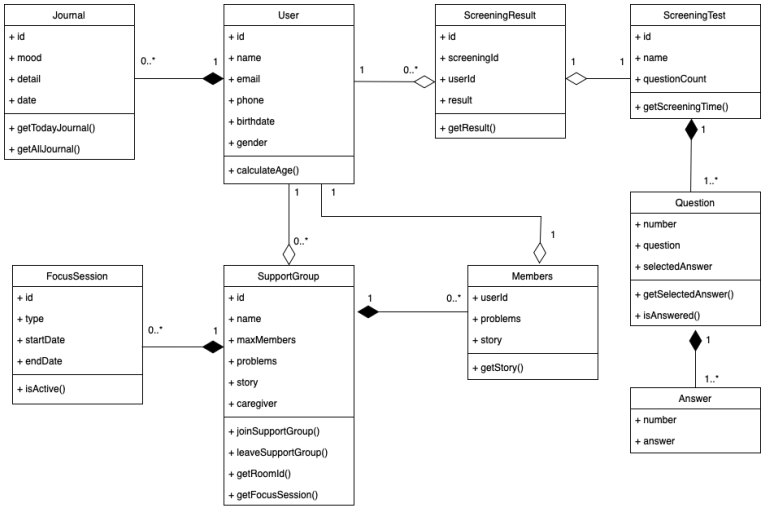
3.1 Desain Sistem

Dalam *use case diagram* tersebut terdapat 9 *use case* yang menggambarkan apa saja yang dapat dilakukan oleh sistem yaitu membuat *support group*, mencari *support group*, bergabung ke dalam c*hat room support group*, keluar dari *support group*, membuat *focus session*, melihat *focus session*, melakukan tes psikologi, menggunakan *mood tracker*, dan melihat konten kesehatan mental. Aktor yang berinteraksi dengan use case tadi adalah *User*. *Use Case Diagram* dapat dilihat pada Gambar 2.



Gambar 2. Use Case Diagram.

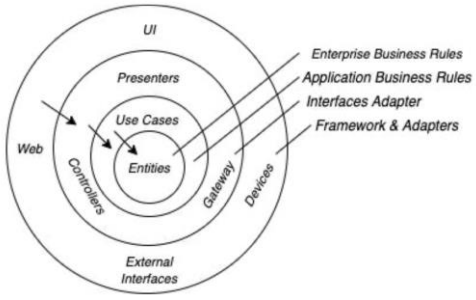
Class diagram menggambarkan hubungan kelas-kelas dalam sistem dan bagaimana kelas tersebut berinteraksi dengan kelas lainnya. Adapun kelas yang digunakan dalam pengembangan aplikasi ini adalah *Journal*, *User*, *SupportGroup*, *Members*, *FocusSession*, *ScreeningResult*, *ScreeningTest*, *Question*, dan *Answer*. *Class diagram* untuk rancangan aplikasi dapa dilihat pada Gambar 3.



Gambar 3. Class Diagram.

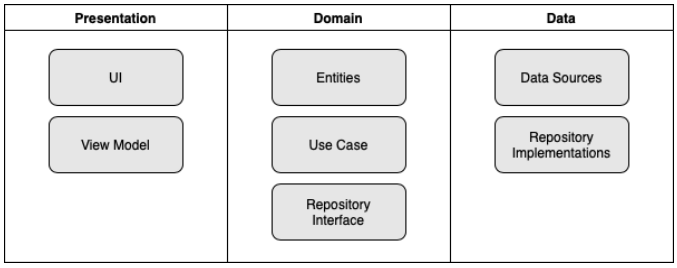
3.2. Penerapan *Clean Architecture*

Aplikasi dikembangkan menggunakan prinsip *clean architecture* dengan tujuan untuk membuat sistem mampu beradaptasi dengan kedinamisan kebutuhan bisnis dengan upaya yang minimal[8], [9]. Dalam penerapan prinsip *clean architecture* dilakukan pemisahan komponen pada aplikasi menjadi komponen yang independen dan modular. Komponen aplikasi yang independen memudahkan pengembang dalam mencari *bug* atau *error* dan pengembang dapat memperbaiki *bug* atau *error* tersebut tanpa mempengaruhi komponen lainnya. Seperti yang terlihat pada Gambar 4, *clean architecture* memiliki 4 level *layer* di antaranya *entities*, *use case/interactor*, *interface adapter*, dan *frameworks*.



Gambar 4. *Clean Architecture Layer*s.

Seperti yang sudah dijelaskan sebelumnya bahwa *clean architecture* memiliki 4 level *layer*, selanjutnya *layer-layer* tersebut dikelompokan menjadi 3 bagian yaitu *presentation*, *domain*, dan *data* [10]seperti pada Gambar 5. *Presentation layer* akan berfungsi untuk mengatur tampilan, *domain layer* berfungsi untuk mengatur proses bisnis, dan *data layer* untuk mengatur data sources. Seperti yang terlihat pada Gambar 5, pada *layer* data terdapat implementasi dari *repository interface* yang ada pada *layer domain*, hal ini dilakukan untuk mengaplikasikan *dependency rule*. *Dependency rule* akan dijelaskan pada bagian selanjutnya.



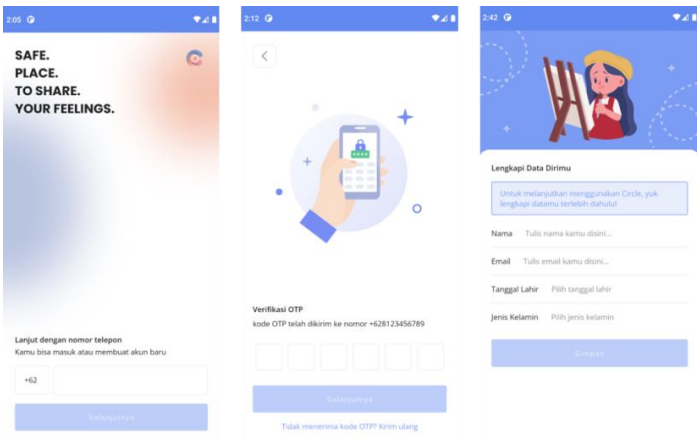
Gambar 5. Pembagian Project.

Dependency rule adalah aturan yang mengatur ketergantungan antara satu *layer* ke *layer* yang lainnya. Pada *clean architecture*, *layer presentation* dan *layer data* harus bergantung kepada *layer domain*, bukan sebaliknya. Pada *layer presentation* akan menampilkan data yang didapat dari *use case* yang ada pada *domain layer*, hal ini akan menyebabkan *presentation layer* bergantung kepada *domain layer*. Selanjutnya *domain* akan mengakses data pada *data layer*[8]. Untuk mengakses *data layer* tanpa bergantung pada *layer* tersebut dapat dilakukan dengan cara menerapkan *dependency inversion* yakni dengan membuat *abstraction interface*, hal ini akan membuat data *layer* tergantung pada *interface* tersebut. *Dependency rule* akan membuat proses bisnis yang ada pada *domain layer* menjadi bagian yang independen. Saat *domain layer* menjadi independen dan tidak tergantung pada *layer* lainnya, maka jika terjadi perubahan pada *presentation layer* atau *data layer* tidak akan mempengaruhi proses bisnis. Saat *dependency rule* diterapkan, aplikasi akan menjadi independen, *maintainable*, dan *testable*.

Gambar 6. *Dependency Rule.*

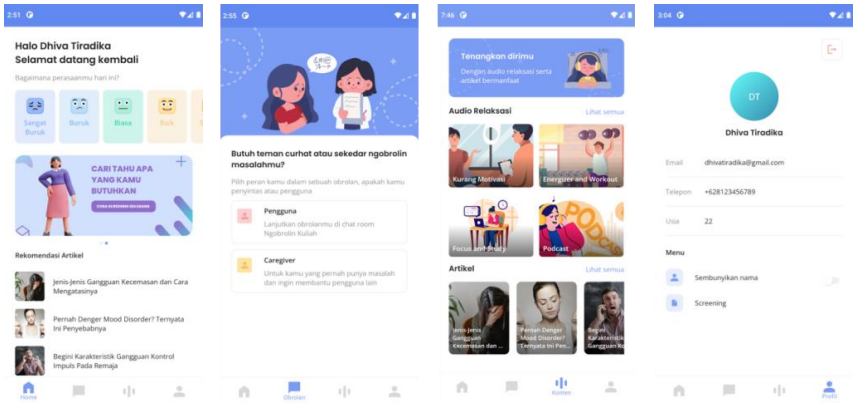
3.3 Implementasi Sistem

Pada implementasi sistem akan dijelaskan hasil dari pengembangan aplikasi Android kesehatan mental menggunakan Kotlin dengan menerapkan prinsip *clean architecture*. Pada bagian ini, hasil dari implementasi sistem akan dijelaskan dalam bentuk tampilan dari aplikasi Android serta deskripsi dari masing-masing tampilan tersebut.



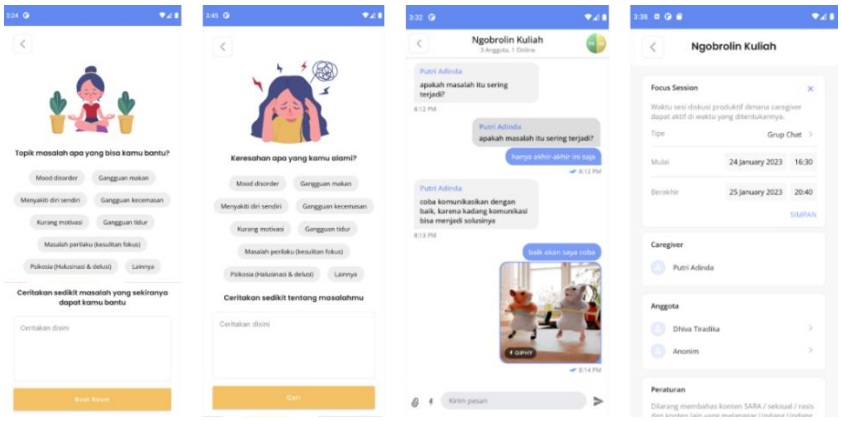
Gambar 7. Halaman Autentikasi.

Saat pengguna pertama kali menggunakan aplikasi mereka perlu masuk terlebih dahulu. Pengguna diminta memasukan nomor telepon mereka, lalu pengguna perlu memasukan 6 digit kode *OTP* yang telah dikirimkan ke nomor telepon yang pengguna masukan. Selanjutnya pengguna diminta untuk melengkapi data diri mereka. Halaman proses autentikasi dapat dilihat pada Gambar 7.



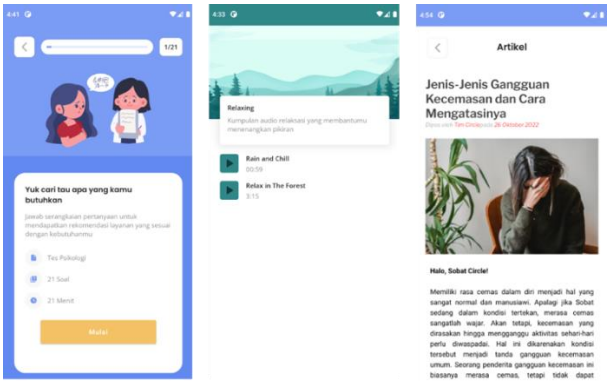
Gambar 8. Halaman Utama.

Saat pengguna berhasil masuk menggunakan nomor telepon, mereka akan diarahkan ke halaman utama, dimana pada halaman utama terdapat 4 menu yaitu *home*, obrolan, konten, dan profil. Pada halaman *home* pengguna dapat menggunakan *mood tracker* dan melihat *banner* informasi terkini. Pengguna dapat menggunakan fitur *support group* pada halaman obrolan, untuk menikmati konten kesehatan mental pengguna dapat masuk ke halaman konten. Pada halaman profil pengguna dapat melihat data diri mereka serta memilih beberapa menu seperti pilihan untuk anonim dan fitur *screening*.



Gambar 9. Fitur Support Group.

Pengguna dapat masuk ke dalam *support group* sebagai anggota maupun *caregiver*, mereka dapat memilih peran itu pada halaman obrolan. Setelah masuk ke dalam *support group*, pengguna dapat mulai berdiskusi pada grup chat support group. Pada support group pengguna juga dapat menggunakan fitur-fitur seperti mengirim gambar, mengirim *GIF*, memberikan reaksi, membalas pesan, melaporkan pesan, menyematkan pesan yang dipilih, serta menggunakan *focus session*.



Gambar 10. Fitur Tambahan.

Selain menggunakan fitur *support group*, pengguna juga dapat menggunakan fitur tambahan seperti *screening* untuk mengetahui layanan apa yang pengguna butuhkan, fitur konten seperti audio relaksasi dan artikel kesehatan mental.

3.4 Pengujian Sistem

Pengujian sistem dengan *blackbox testing* dilakakukan dengan tujuan untuk memastikan sistem yang dibuat telah sesuai dan dapat mencapai tujuan bisnis yang telah ditentukan sebelumnya. Pengujian sistem juga dilakukan untuk mendeteksi serta menemukan *error* dan *bug* pada sistem yang telah dibuat. Pengujian *blackbox testing* dilakukan pada 9 proses yaitu proses *login*, proses *signup*, proses pembuatan *support group*, proses pencarian *support group*, bergabung pada *support group*, pembuatan *focus session*, proses keluar dari *support group*, pembuatan *focus session*, proses *screening*, proses konten, dan proses *mood tracker*.

Pengujian pembuatan *support group* dilakukan dengan memastikan hasil yang didapatkan sesuai dengan hasil yang diharapkan. Hasil pengujian sistem pada proses pembuatan *support group* dapat dilihat pada Tabel 1.

Tabel 1. Pengujian Proses Pembuatan Menu Support Group.

No	Kelas Uji	Hasil yang diharapkan	Hasil Pengujian	Keterangan
1	Input nama kosong	Button selanjutnya tidak dapat ditekan	Button selanjutnya tidak dapat ditekan	Sesuai
2	Button selanjutnya ditekan	Sitem mengarahkan ke halaman support group detail	Sitem mengarahkan ke halaman support group detail	Sesuai
3	Item topik masalah belum dipilih dan input masalah kosong	Button buat room tidak dapat ditekan	Button buat room tidak dapat ditekan	Sesuai
4	Item topik masalah sudah dipilih dan input masalah kosong	Button buat room tidak dapat ditekan	Button buat room tidak dapat ditekan	Sesuai
5	Item topik masalah belum dipilih dan input masalah terisi	Button buat room tidak dapat ditekan	Button buat room tidak dapat ditekan	Sesuai
6	Item topik masalah sudah dipilih dan input masalah terisi	Button buat room tidak dapat ditekan	Button buat room tidak dapat ditekan	Sesuai

Berdasarkan hasil pengujian pada Tabel 1, seluruh button telah berfungsi sesuai dengan fungsinya, sehingga hasil dari *blackbox testing* adalah semua keluaran kelas uji sesuai dengan hasil yang diharapkan.

4. Kesimpulan

Kesimpulan hasil perekayasaan pada penelitian dengan penerapan prisnsip *Clean Architecture* pada aplikasi android kesehatan mental menggunakan kotlin yaitu telah dihasilkan aplikasi android kesehatan mental yang dirancang menggunakan *Unified Modeling Language (UML)* dan dikembangkan menggunakan bahasa pemograman *Kotlin* dengan menerapkan prinsip *Clean Architecture* dan telah diuji dengan metode *blackbox testing* dengan hasil seluruh kelas uji sudah sesuai dengan fungsinya masing – masing.

Daftar Pustaka:

[1] D. Ayuningtyas, M. Misnaniarti, and M. Rayhani, "Analisis Situasi Kesehatan Mental Pada Masyarakat di Indonesia dan Strategi Penanggulangannya," *Jurnal Ilmu Kesehatan Masyarakat*, vol. 9, no. 1, Oct. 2018, doi: 10.26553/jikm.2018.9.1.1-10.

[2] I. A. Ridlo, D. Administrasi, K. Kesehatan, K. Masyarakat, and U. Airlangga, "Pandemi COVID-19 dan Tantangan Kebijakan Kesehatan Mental di Indonesia", doi: 10.20473/jpkm.v5i12020.155-164.

[3] W. A. Radiani and J. Ahmad Yani Km, "Kesehatan Mental Masa Kini dan Penanganan Gangguannya Secara Islami," 2019. [Online]. Available: http://jurnal.uin-antasari.ac.id/index.php/jils/article/view/2659

[4] L. S. Rahmat Fadillah, "Perancangan Aplikasi Mobile Learning Berbasis Android di SMK Negeri 6 Padang," *Jurnal Vokasional Teknik Elektronika dan Informatika*, vol. 7, 2019.

[5] S. Arfida, H. Wibowo, and A. F. Setya, "Penerapan Teknologi Android Terhadap Aplikasi Panduan Penggunaan Software Adobe Audition," *IJCCS*, vol. x, No.x, pp. 1–5.

[6] Muhamad Riyan, "Penggunaan Media Pembelajaran Berbasis Aplikasi Android pada Pembelajaran Teks Eksposisi," *Journal DIKSI*, 2021.

[7] S. Wirayudha, T. Wuruk Pribadi, and Moh. S. Arif, "Perancangan Aplikasi Berbasis Android untuk Aktivitas Manajemen Material Galangan Kapal Baru," *Jurnal Teknik ITS*, vol. 6, 2017.

[8] M. Hilmy Bad Rudduja and R. E. Putra, "Penerapan *Clean Architecture* pada Aplikasi Pemesanan Makanan menggunakan Metode Slope One Algorithm," *Journal of Informatics and Computer Science*, p. 2022.

[9] A. Rahman Fajri and S. Rani, "Penerapan Design Pattern MVVM dan *Clean Architecture* pada Pengembangan Aplikasi Android (Studi Kasus: Aplikasi Agree Partner)."

[10] A. W. Subagio and F. Muttaqin, "Tekno Jurnal Teknologi Elektro dan Kejuruan Arif Widiasan Subagio, Faisal Muttaqin | Penerapan *Clean Architecture* pada Pengembangan Sistem Payment Point Online… Penerapan *Clean Architecture* pada Pengembangan Sistem Payment Point Online Bank," 2022. [Online]. Available: http://journal2.um.ac.id/index.php/tekno