BAB II

TINJAUAN PUSTAKA

2.1 Penilitian Terkait

Smart Timer Application merupakan aplikasi Kesehatan Digital berbasis *mobile.* Aplikasi ini digunakan untuk mengatur waktu penggunaan aplikasi sesuai dengan kemauan pengguna, melihat waktu penggunaan *smartphone*, pengingat durasi penggunaan *smartphone*, dan notifikasi peringatan untuk menutup aplikasi jika sudah pada batas yang ditentukan pengguna. Beberapa *platform* telah membuat aplikasi yang berguna untuk memberikan informasi mengenai penggunaan *smartphone* diantaranya telah terdapat di *Google Play Store* seperti aplikasi Kesehatan Digital, *Action Dash*: Kesehatan Digital, dan *Stay Free* – Kesehatan Digital. Aplikasi tersebut secara umum telah memberikan fitur berupa waktu penggunaan *smartphone*, riwayat peluncuran aplikasi, riwayat pemberitahuan, mode tidur, mode fokus, dan batas penggunaan aplikasi. Sama halnya dengan beberapa pengabdian berikut yang menjadi referensi penelitian yang akan dilakukan.

Pada penelitian yang dilakukan oleh Arief Witjaksono dkk meliputi Tes refraksi subyektif dan obyektif, penelitian tentang prevalensi kelainan refraksi, nasehat dan edukasi terhadap kesehatan mata terutama dampak bermain *game* dalam jangka panjang terhadap kesehatan mata.

Studi ini menemukan bahwa jika terlalu sering melihat ke depan monitor, mata dapat rusak karena radiasi yang terpapar dari layar monitor atau karena melihatnya terlalu lama: sinar-X, sinar ultraviolet, gelombang mikro,

dan radiasi elektromagnetik yang sangat lemah.

Radiasi frekuensi rendah (frekuensi sangat rendah/VLF) dan radiasi elektromagnetik frekuensi sangat rendah (frekuensi sangat rendah/ELF) terdeteksi di kornea mata dan diteruskan ke lensa kristal, lensa mata sekolah. Secara fisiologis, anak-anak yang lebih kecil sangat terkena dampaknya karena saraf mereka rusak [8].

Selain itu, pada penelitian yang dilakukan oleh Windy Patadungan dkk melalui observasional analitik menggunakan pendekatan *cross sectional*, Hasil penelitian ini menunjukkan bahwa intensitas (jam) paparan cahaya *smartphone* per hari tidak berpengaruh terhadap ketajaman penglihatan atau ketajaman penglihatan (p-value = 0,769 > 0,05). Kurangnya efek ini diduga disebabkan oleh adanya jeda waktu dan waktu istirahat otot mata saat sehari-hari menggunakan *smartphone*, sehingga dapat mencegah kelelahan mata [4].

2.2 Landasan Teori

2.2.1. Aplikasi

Aplikasi adalah jenis perangkat lunak komputer yang secara langsung memanfaatkan kemampuan komputer untuk menjalankan tugas-tugas yang diinginkan oleh pengguna. Dalam konteks ini, aplikasi dapat dianggap sebagai penghubung yang mengambil perintah dari pengguna komputer, kemudian mengolahnya melalui perangkat keras untuk melakukan Tindakan yang diminta [9].

2.2.2 Mobile

Mobile adalah sebuah sistem perangkat lunak yang memungkinkan pengguna untuk melakukan aktivitas secara *mobile* dengan menggunakan perangkat PDA (*Personal Digital Assistant*) atau asisten digital pada telepon genggam atau seluler. Android dan iOS adalah dua sistem operasi *mobile* yang dominan di pasar saat ini. Aplikasi *mobile* juga dikenal dengan sebutan *web app, online app, iPhone app,* atau *smartphone app* [10].

2.2.3 Aplikasi *Mobile*

Aplikasi *mobile* berasal dari gabungan dua kata, yaitu "aplikasi" dan "*mobile*". Secara umum, aplikasi merujuk kepada program yang dirancang untuk melakukan suatu fungsi tertentu bagi pengguna atau aplikasi lainnya. Sementara itu, "*mobile*" mengacu pada perpindahan dari satu tempat ke tempat lain. Jadi, secara keseluruhan, aplikasi *mobile* adalah program siap pakai yang dirancang untuk menjalankan fungsi tertentu dan dapat diinstal pada perangkat *mobile* [11].

2.2.4 Monitoring

Monitoring adalah proses pengumpulan dan analisis informasi berdasarkan indikator yang telah ditetapkan secara sistematis dan berkelanjutan terhadap kegiatan atau program tertentu. Tujuannya adalah untuk memungkinkan tindakan korektif yang diperlukan untuk meningkatkan program atau kegiatan tersebut. Monitoring juga dapat diartikan sebagai kesadaran tentang informasi yang ingin diketahui, di

mana pemantauan dilakukan secara teratur untuk memungkinkan pengukuran terhadap progres menuju tujuan atau sebaliknya [12].

2.2.5 *Flutter* SDK

Gambar 2. 1 Logo *Flutter*

Flutter adalah *platform* yang memungkinkan pengembang untuk membuat aplikasi yang dapat berjalan di berbagai *platform* hanya dengan menggunakan satu basis kode (*codebase*). Dengan *Flutter* , pengembang dapat membuat aplikasi yang dapat digunakan di *platform mobile* seperti Android dan iOS, serta di *web* dan *desktop*.

Flutter memiliki dua komponen penting, yaitu, *Software Development Kit (SDK)* dan juga *Framework User Interface* .

a) *Software Development Kit (SDK)* adalah kumpulan alat atau perangkat lunak yang berfungsi untuk memungkinkan pengembang membuat aplikasi yang dapat dijalankan di berbagai *platform*.
b) *Framework UI* adalah kumpulan komponen antarmuka pengguna (UI) yang disediakan oleh suatu *framework* atau *library*. Komponen UI ini mencakup berbagai elemen seperti teks, tombol,

navigasi dan lainnya yang memungkinkan pengembang untuk membangun antarmuka pengguna aplikasi.

Flutter adalah *platform* yang gratis dan *open source*. Untuk menggunakan *Flutter* , Anda perlu mempelajari bahasa pemrograman Dart. Berbeda dengan *framework front-end* pada umumnya yang menggunakan *JavaScript* sebagai bahasa pemrogramannya, *Flutter* menggunakan Dart.

Flutter pertama kali dikembangkan oleh *Google* sejak 2015 dan resmi diluncurkan pada Desember 2018. Pada tahun 2019, popularitas *Flutter* mulai meningkat pesat dan banyak pengembang yang beralih menggunakan *Flutter* untuk mengembangkan aplikasi mereka [13].

2.2.6 Android Studio

Gambar 2. 2 Logo Android Studio

Android Studio adalah *tools Integrated Development Environment (IDE)* resmi yang dikembangkan oleh *Google* dan *JetBrains* untuk memudahkan pengembangan aplikasi Android. IDE ini memiliki berbagai komponen yang lengkap, termasuk *source code editor , compiler,* dan *debugger* [14].

2.2.7 Visual Studio Code

Gambar 2. 3 Logo *Visual Studio Code*

Visual Studio Code adalah aplikasi *editor* teks gratis yang dikembangkan oleh *Microsoft*. Aplikasi ini dapat digunakan untuk berbagai bahasa pemrograman tanpa perlu mengganti *editor* . *Visual Studio Code* dapat dijalankan di berbagai *platform* seperti *Windows, Linux,* dan *macOS*. Keunggulan *Visual Studio Code* adalah kemampuannya untuk memudahkan *programmer* saat berpindah antara bahasa pemrograman tanpa harus mengganti aplikasi *editor* . Pengguna tidak perlu memahami dan mengkonfigurasi ulang *tools* di aplikasi *editor* yang baru. *Visual Studio Code* juga memberikan kebebasan kepada penggunanya dalam hal tema, *debugger*, *extension*, dan fitur lainnya [15].

2.2.8 UML (Unified Modelling Language)

Unified Modeling Language adalah metode dalam rekayasa perangkat lunak yang digunakan untuk menggambarkan alur dan cara kerja sistem, fungsi, tujuan, dan mekanisme kontrol dari suatu sistem. Dalam rekayasa perangkat lunak, terutama dalam analisis dan perancangan sistem informasi, penggunaan konsep pemrograman berorientasi objek telah menjadi umum. Konsep ini melibatkan

pandangan sistem sebagai objek yang menggabungkan data dan proses atau bisa bekerja secara mandiri dalam satu set sistem (*package*).

Metodologi desain sistem informasi mencakup empat model UML yang paling efektif menggambarkan desain sistem, yaitu *Use case Diagram, Class Diagram, Sequence Diagram,* dan *Behavioral State Machine Diagram.* Keempat teknik pemodelan UML ini juga dikenal sebagai empat teknik pemodelan dasar (inti) berbasis UML. Keempat teknik pemodelan UML ini umum digunakan dalam proyek berorientasi objek [16].

Dalam Perancangan beriorentasi objek berbasiskan *UML* menggunakan alat bantu sebagai berikut:

1. *Use case Diagram*

   Use case diagram merupakan model perilaku sistem informasi yang dibuat. *Use case* digunakan untuk menentukan kapabilitas apa saja yang ada pada suatu sistem informasi dan siapa saja yang berhak menggunakan kapabilitas tersebut. Tabel berikut menunjukkan simbol-simbol yang digunakan dalam diagram *use case*.

   Tabel 2. 1 Simbol *Use case*

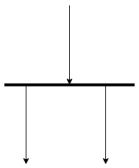
No	Simbol	Keterangan
1		*Use case* : Abstraksi dan interaksi antarasistem dan *actor*.

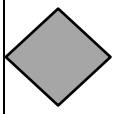
2		*Actor* : Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasidengan *use case.*
3		Asosiasi antara *actor* dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan data.
4		Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem.
5	<<include>>	*Include*, Menunjukkan bahwa satu *use case* mengikutsertakan atau mencakup fungsionalitas dari *use case* lain(*use case* inkludir)
6	<<extends>>	*Extend*, merupakan perluasan dari *use case* lain jika kondisi tertentu terpenuhi.

2. *Activity Diagram*

Activity Diagram menggambarkan alur kerja atau aktivitas suatu sistem atau proses bisnis. Tabel berikut menunjukkan simbol-simbol yang digunakan dalam diagram aktivitas.

Tabel 2. 2 Simbol *Activity Diagram*

No	Simbol	Keterangan
1		*Start Point*, titik awal dari suatu aktivitas atau proses dalam diagram
2		*End Point*, titik akhir dari suatu aktivitas atau proses dalam diagram alur kerja.
3		*Activities*, menggambar Tindakan atau langkah-langkah yang dilakukan dalam suatu proses atau kegiatan bisnis.
4		*Fork/*percabangan digunakan untuk menunjukan kegiatan yang dikakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.

5		*Decision Points*, menggambarkan alur kerja dimana pengambilan keputusan terjadi (biasanya "*true"* atau "*false").*
6		*Swimlane,* pembagian *Activity Diagram* untuk menunjukkan siapa yang bertanggung jawab atas setiap aktivitas.

2.2.9 Pengujian Sistem

1. *White Box Testing*

White Box Testing digunakan berdasarkan kode sumber untuk menentukan bagian kode mana yang tidak berfungsi dengan benar. Keuntungan *white box testing* mencakup kemampuan untuk menghapus bagian kode tersembunyi yang tidak relevan dan melakukan pengujian komprehensif karena setiap bagian struktur atau logika diperiksa. Ini membantu mengoptimalkan kode dan memungkinkan memulai *white box testing* bahkan ketika GUI masih dalam pengembangan. Dalam melakukan pengujian pada aplikasi *Smart Timer* dengan metode *white box testing* penguji akan melihat kode secara keseluruhan kemudian melakukan testing terhadap setiap kode yang ada jika kode tidak berfungsi maka bisa dihapus atau digantikan dengan kode yang difungsikan sebagaimana mestinya [17].

2. *Black Box Testing*

Black Box Testing disebut sebagai pengujian perilaku. Struktur *interior*, logika perangkat lunak yang diuji tidak diketahui oleh penguji. Penguji didasarkan kepada spesifikasi kebutuhan dan tidak perlu dilakukan analisis kode. Kelebihan *black box testing* diantaranya membantu dalam hal penemuan aspek yang tidak terpenuhi dari spesifikasi kebutuhan yang diberikan dalam pengembangan perangkat lunak. Sedangkan kekurangan *black box testing* diantaranya pengujian tidak dapat dilakukan sepenuhnya dikarenakan pengetahuan penguji terbatas tentang perangkat lunak yang diuji [17]. Dalam pengujian pada aplikasi *Smart Timer* menggunakan metode *black box testing* penguji hanya melihat *interface* aplikasi kemudian dilakukan pengujian pada fungsi dari spesifikasi aplikasi.