RANCANG BANGUN SISTEM MANAJEMEN WAKTU DENGAN TEKNIK POMODORO MENGGUNAKAN ARSITEKTUR MVVM

Ihsan Nurul Iman, Purwantoro, Aries Suharso
Program Studi Teknik Informatika S1, Fakultas Ilmu Komputer
Universitas Singaperbangsa Karawang, Telukjambe Timur, Karawang, Indonesia
1910631170086@student.unsika.ac.id

ABSTRAK

Manajemen waktu merupakan hal yang krusial untuk menyelesaikan lebih banyak tugas dengan lebih efektif. Namun, manajemen waktu yang baik merupakan hal yang sulit digapai banyak orang. Terlebih, di era maraknya internet dan media sosial, para pekerja sering mengalami interupsi dan kesulitan untuk fokus pada pekerjaannya. Oleh sebab itu, diperlukan adanya sistem manajemen waktu yang baik. Penelitian ini bertujuan untuk merancang sistem manejemen waktu dengan salah satu teknik manajemen waktu terpopuler bernama Teknik Pomodoro. Teknik yang dicetuskan Francesco Cirillo ini telah terbukti efektif oleh berbagai studi baik untuk produktivitas maupun kesehatan. Dalam penelitian ini digunakan metodologi penelitian *waterfall* yang terdiri dari tahap analisis, desain rancangan sistem, implementasi, pengujian, dan pemeliharaan. Sementara itu, aplikasi ini dirancang dengan arsitektur *Model-View-ViewModel* (MVVM), sebuah arsitektur yang dirancang untuk memisahkan logika bisnis dengan tampilkan aplikasi. Hasil penelitian ini adalah sebuah aplikasi manajemen waktu dengan Teknik Pomodoro berbasis MVVM yang berjalan di perangkat *desktop*. Melalui pengujian beta, Aplikasi ini telah diuji coba oleh sejumlah pengguna secara langsung. Hasilnya, aplikasi berjalan sesuai ekspektasi dan memenuhi kebutuhan pengguna berdasarkan metode penilaian yang digunakan.

Kata kunci: *Teknik Pomodoro, WPF, MVVM, manajemen waktu, aplikasi desktop*

1. PENDAHULUAN

Manajemen waktu merupakan hal yang krusial dalam kehidupan sehari-hari. Manajemen waktu yang baik merupakan kunci untuk menyelesaikan tugas secara efektif dan efisien. Namun, memiliki manajemen waktu yang baik merupakan hal yang sulit dicapai bagi sebagian orang.

Menurut sebuah penelitian [1], pekerja pada umumnya terkena gangguan (interupsi) sebanyak enam hingga delapan kali sehari yang menyita sekitar 28% waktu produktif mereka. Penelitian lain menyebutkan bahwa para pekerja terkena interupsi ssekitar 40% dari waktu kerja mereka. Ketika mereka kembali ke pekerjaan mereka setelah mengalami interupsi, dibutuhkan waktu sekitar 25 menit untuk kembali ke keadaan kognitif awal [2].

Oleh sebab itu, memiliki kemampuan manajemen waktu yang baik sangatlah penting. Untuk tujuan tersebut, ada banyak teknik manajemen waktu yang dikembangkan. Salah satu teknik terpopuler adalah Teknik Pomodoro yang dicetuskan oleh Francesco Cirilio pada akhir 1980-an. Teknik ini merupakan sebuah metode manajemen waktu yang dapat membantu meningkatkan produktivitas seseorang dengan cara membagi waktu menjadi interval fokus yang teratur.

Teknik Pomodoro merupakan pilihan yang tepat karena menyesuaikan dengan psikologi manusia. Penelitian yang dipimpin oleh Profesor Psikologi Alejandro Lleras dari University of Illinois menyimpulkan bahwa istirahat sejenak dari mengerjakan tugas dapat secara signifikan meningkatkan fokus seseorang pada pekerjaannya untuk periode waktu yang lebih lama [3]. Selain itu, mengerjakan sebuah tugas selama berjam-jam bisa menyebabkan kejenuhan yang akut. Dalam keadaan tersebut, seseorang bisa memiliki pemikiran yang buntu [4]. Teknik pomodoro yang membagi waktu kerja menjadi kecil-kecil serta diselingi waktu istirihat pendek memungkinkan seseorang menghindari isu-isu di atas. Tak hanya itu, duduk selama berjam-jam dapat menyebabkan berbagai isu kesehatan. Sebuah studi menyarankan bahwa setiap setengah jam duduk, seseorang disarankan melakukan lari-lari kecil selama lima menit [5] . Saat memasuki sesi istirahat, pengguna Teknik Pomodoro dapat melakukan aktivitas kecil sembari meringankan kinerja badan dan otak, seperti halnya berlari-lari kecil. Pendek kata, Teknik Pomodoro sangat sesuai dengan kesimpulan dari penelitian-penelitian yang telah disebutkan.

Penelitian ini bertujuan untuk merancang sebuah aplikasi *desktop* manajemen waktu berbasis Pomodoro menggunakan kerangka kerja *Windows Presentation Foundation (WPF). WPF* merupakan sebuah kerangka kerja *(framework)* untuk menciptakan aplikasi *desktop* yang dirancang oleh Microsoft. *Framework* ini sangat erat kaitannya dengan sebuah arsitektur bernama *Model-View-ViewModel* (MVVM), oleh sebab itu aplikasi ini dirancang menggunakan arsitektur tersebut. MVVM merupakan sebuah model arsitektur turunan *Model-View-Controller (MVC)* yang diciptakan untuk memisahkan logika bisnis dengan tampilan aplikasi.

2. TINJAUAN PUSTAKA

2.1. Teknik Pomodoro

Teknik Pomodoro merupakan sebuah metode manajemen waktu untuk membantu meningkatkan produktivitas seseorang dengan cara membagi waktu ke dalam interval fokus yang teratur. Teknik yang dicetus Francesco Cirilio ini membagi pekerjaan menjadi periode fokus terbatas (biasanya 25 menit) yang diikuti oleh istirahat pendek (biasanya lima menit). Selama periode fokus terbatas, pengguna tidak boleh melakukan hal lain selain mengerjakan tugas yang telah ditetapkan sebelumnya. Jika tugas sebelumnya telah selesai dikerjakan, pengguna beralih mengerjakan tugas lain. Setiap empat kali menyelesaikan periode fokus terbatas (mendapatkan empat buah pomodoro), pengguna teknik ini beristirahat selama 15 hingga 30 menit. Dalam aplikasinya, angka-angka yang disebutkan tadi bukanlah tolok ukur yang mutlak, namun merupakan rekomendasi dari penemu teknik ini [6].

2.2. C#

C# (dibaca "C sharp") adalah sebuah bahasa pemrograman yang dikembangkan oleh Microsoft. C# mendukung paradigma *static typing, type-safe programming, component-oriented, functional, imperative, declarative,* dan *generic* [7].

2.3. Basis Data

Basis data adalah sekumpulan informasi atau data terorganisasi yang biasa disimpan di dalam suatu sistem komputer untuk tujuan diakses, dikelola, dan diolah secara efisien. Pada umumnya, data pada basis data menggunakan *structured query language (SQL)* atau sering juga disebut basis data relasional. Dalam struktur ini, data dimodelkan dengan baris dan kolom dalam sebuah tabel [8]. Dengan pemodelan tersebut, data dapat ditambah, dihapus, diubah, dan dibaca. Dinamakan relasional karena tabel-tabel pada basis data SQL dapat berelasi satu sama lain untuk menangani tipe data yang lebih kompleks.

2.4. Pemaketan Perangkat Lunak

Pemaketan perangkat lunak *(software packaging)* adalah proses pengumpulan, konfigurasi, dan penyediaan sebuah perangkat lunak dalam sebuah paket yang dapat diinstal dengan mudah di sistem operasi yang ditargetkan. Paket perangkat lunak ini umumnya berisi semua berkas, dependensi, dan instruksi yang diperlukan untuk menginstal dan menjalankan sebuah perangkat lunak.

2.5. Windows Presentation Foundation

Windows Presentation Foundation (WPF) adalah sebuah kerangka kerja *(framework)* pengembangan aplikasi desktop yang diperkenalkan Microsoft sebagai bagian dari platform .NET pada tahun 2006. WPF memisahkan antara antarmuka pengguna *(user interface)* dengan logika bisnis *(business logic)* [9]. WPF memungkinkan pengembang untuk membuat aplikasi desktop yang interaktif dan menarik. WPF menyediakan seperangkat fitur pengembangan aplikasi yang lengkap seperti *Extensible Application Markup Language (XAML)*, control, data binding, layout, grafis 2D dan 3D, animasi, style, template, dokumen, media, teks, dan *typography* [10].

2.6. Model-View-ViewModel (MVVM)

Model-View-ViewModel (MVVM) merupakan variasi dari *Model-View-Controller* (MVC) yang ditemukan oleh arsitek Microsoft Ken Cooper dan Ted Peters [11]. Tujuan dari MVVM adalah memisahkan tanggung jawab pengembang aplikasi dengan desainer. Dengan penerapan MVVM, aspek *View* atau tampilan visual dapat sepenuhnya dikerjakan seorang desainer tanpa campur tangan dari seorang programmer. Ini dimungkinkan dengan adanya bahasa markup seperti XAML serta fitur data binding yang memisahkan logika bisnis (business logic) dari *View* [12]. Selain itu, salah satu hambatan ketika menjalankan unit testing adalah hubungan yang terlalu erat antara logika bisnis dengan *View*. Dengan terpisahnya logika bisnis dari *View*, MVVM memungkinkan *unit testing* untuk berjalan dengan lebih mudah [13].

2.7. SQLite

SQLite adalah sebuah RDBMS yang didistribusikan dalam bentuk *library*. SQLite merupakan anggota dari keluarga embedded database yaitu DBMS yang penggunaannya terintegrasi ke dalam aplikasi. RDBMS merupakan mesin basis data yang paling sering diterapkan dan telah banyak digunakan di berbagai jenis sistem operasi dan perangkat [14].

2.8. Pengujian Unit *(Unit Testing)*

Menurut Ian Sommerville (2021), *unit testing* adalah sebuah proses pengujian pada komponen-komponen atau unit-unit program, seperti pada fungsi *(method)* atau kelas objek *(object class).* Pengujian ini harus berfokus pada fungsionalitas suatu program.

Tujuan *unit testing* adalah memvalidasi bahwa setiap unit yang ditulis pada sumber kode bekerja sesuai harapan. Unit testing biasa dilakukan oleh seorang programmer atau *quality assurance (QA) tester* [15].

2.9. Pengujian Alpha *(Alpha Testing)*

Pengujian alpha *(alpha testing)* adalah pengujian yang berfokus pada struktur internal dan cara kerja aplikasi. Tujuan dari pengujian ini adalah memastikan bahwa fungsi-fungsi dasar aplikasi berjalan dengan baik, meskipun mungkin masih terdapat sejumlah eror, cacat *(bug),* atau fitur-fitur yang belum rampung. Alpha testing biasa dilakukan oleh pihak internal seperti pengembang aplikasi itu sendiri. Setelah pengujian alpha, perangkat lunak biasanya akan diuji dengan pengujian beta *(beta testing)* [16].

2.10. Pengujian Beta *(Alpha Testing)*

Pengujian beta *(beta testing)* mirip dengan *alpha testing*. Hanya saja, pengujian ini dilakukan dengan melibatkan langsung pengguna aplikasi yang sesungguhnya [16]. Dengan kata lain, pengembang aplikasi akan mengumpulkan sejumlah sampel dan meminta responden untuk mencoba produk secara langsung dan melaporkan apakah produk sudah berfungsi dengan baik. Selain itu, pengujian ini dilakukan untuk meminta umpan balik dari pengguna untuk mengidentifikasi kekurangan-kekurangan pada aplikasi. Jumlah responden dalam pengujian ini bergantung sepenuhnya dengan kompleksitas dan tujuan proyek. Untuk proyek sederhana yang dikelola pribadi, empat hingga lima responden sudah cukup untuk pengujian beta [17].

2.11. Metodologi Waterfall

Alur hidup sistem *pengembangan (system development life cycle)* merupakan sebuah kerangka kerja pada rekayasa perangkat lunak *(software engineering)* yang menjelaskan tahap-tahap yang terlibat dalam pengembangan sebuah sistem informasi. Kerangka kerja ini didesain untuk menstruktur, merencanakan, dan mengontrol proses pengembangan sebuah sistem [18].

Salah satu model SDLC yang populer adalah waterfall. Dalam waterfall, proyek dipecah menjadi fase-fase yang berurutan. Setiap fase bergantung dengan keterselesaian fase sebelumnya. Fase-fase ini pada umumnya adalah pengumpulan data, desain rancangan sistem, implementasi, pengujian, dan pemeliharaan. Meskipun demikian, fase-fase ini bisa berbeda menyesuaikan kebutuhan pengembang aplikasi [19].

2.12. Unified Modelling Language (UML)

Menurut Fowler (2004), *Unified Modeling Language (UML)* adalah salah satu notasi grafis yang menggambarkan konsep-konsep dari suatu bahasa. UML membantu menggambarkan dan merancang sistem perangkat lunak, terutama sistem perangkat lunak yang dibangun dengan gaya berorientasi objek (OO).

UML memiliki banyak jenis diagram yang membantu menjelaskan konsep dari suatu bahasa. Dari sekian banyak diagram tersebut, beberapa di antaranya yang populer adalah *class diagram*, *activity diagram*, *use case diagram*, dan *sequence diagram*.

3. METODE PENELITIAN

Penelitian ini akan menggunakan metode pengembangan perangkat lunak dengan model *waterfall*. Dalam model ini, pengerjaan proyek akan dibagi menjadi tahap-tahap yang dikerjakan secara sistematis. Tahap-tahap yang penulis anut dalam proyek ini meliputi analisis, desain rancangan sistem, implementasi, pengujian, dan pemeliharaan. Tidak ada aturan baku dalam model *waterfall*. Meski dikerjakan secara sistematis, penulis dapat secara fleksibel kembali ke tahap-tahap sebelumnya apabila terdapat kekurangan atau sesuatu yang terlewat.

a. Pada tahap analisis, akan diindentifikasikan kebutuhan-kebutuhan yang perlu ada pada aplikasi ini.
b. Pada tahap desain rancangan sistem, akan dilakukan beberapa jenis desain untuk menentukan tingkah laku *(behavior)* dan tampilan aplikasi. Pertama-tama, akan dilakukan desain diagram UML dengan diagram kasus penggunaan *(use case diagram)*. Kemudian, akan dilakukan desain tampilan aplikasi dengan Figma.
c. Pada tahap implementasi, akan dilakukan perancangan aplikasi menggunakan hal-hal yang sudah dipersiapkan pada tahap analisis dan desain.
d. Pada tahap pengujian, aplikasi yang sudah dirancang akan diuji dengan metode pegujian beta untuk memastikan bahwa aplikasi sudah berjalan dengan baik.
e. Terakhir, pada tahap pemeliharaan, proyek yang berhasil dibangun akan dipublikasikan sebagai proyek berlisensi terbuka ke dalam repositori GitHub. Selain itu, aplikasi ini akan dipublikasikan pada sebuah halaman web *static* dengan GitHub Pages agar aplikasi ini dapat digunakan oleh khalayak umum.

4. HASIL DAN PEMBAHASAN

4.1. Analisis

Pada tahap analisis dilakukan analisis kebutuhan-kebutuhan pengguna aplikasi ini. Hasil dari analisis ini adalah sebagai berikut.

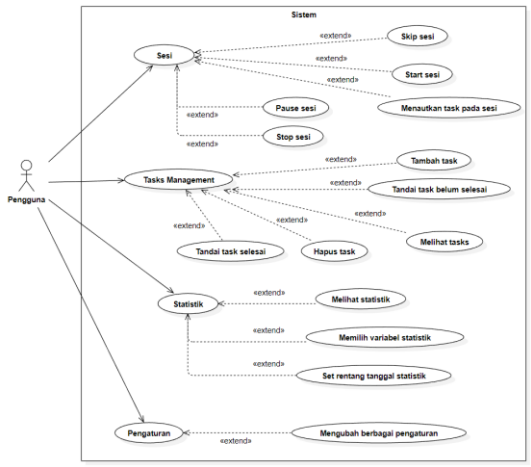
a. Pengguna dapat memulai (start), menghentikan sementara (pause), melanjutkan (resume), mengulang (reset/stop), dan pergi ke sesi selanjutnya (skip) saat menjalankan suatu sesi.
b. Pengguna dapat menambah, mengedit, dan menghapus tugas (task).
c. Pengguna dapat memindahkan suatu tugas yang belum selesai (in-progress) menjadi selesai (completed), demikian sebaliknya.
d. Pengguna dapat mengatur tugas (task) untuk sesi yang sedang berjalan.
e. Pengguna dapat mengubah variabel-variabel pada teknik Pomodoro.
f. Pengguna dapat melihat hasil statistik penggunaan aplikasi untuk mengetahui progres serta hasil penggunaan teknik Pomodoro ini.
g. Pengguna dapat memulai (start) dan menghentikan sementara (pause) sesi yang berlangsung dengan hotkey.
h. Pengguna dapat mengatur sejumlah pengaturan yang memudahkan dan membuat nyaman penggunaan aplikasi (mengatur jenis alarm, volume suara, durasi alarm, push notification, dan tema aplikasi).

i. dapat mengatur jenis alarm, volume suara, dan durasi alarm yang digunakan saat suatu sesi berakhir
j. Pengguna dapat me¬-reset pengaturan ke pengaturan awal.
k. Penulis memutuskan untuk menghitung satu sesi sebagai satu hari. Oleh sebab itu, statistik juga akan ditunjukkan dalam periode minimal satu hari.

4.2. Desain Rancangan Sistem

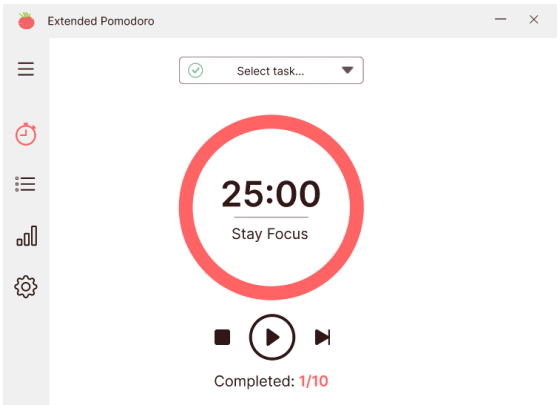
Pada tahap ini dilakukan dua jenis desain yaitu desain sistem dengan UML *(Unified Modelling Language)* dan desain antarmuka *(user interface design).* Tahap ini membantu memperjelas tampilan dan tingkah laku *(behavior)* pada aplikasi yang akan dirancang nantinya.

Pada desain UML, dilakukan desain menggunakan diagram kasus penggunaan *(use case diagram).* Diagram ini berguna untuk mendeskripsikan penggunaan sistem (Fowler, 2004). Hasilnya sebagai berikut.



Gambar 1. Kasus penggunaan aplikasi

Setelah itu, dilakukan desain antarmuka. Dalam hal ini digunakan sebuah alat desain antarmuka bernama Figma. Berikut contoh hasil desain aplikasi pada tahap ini.



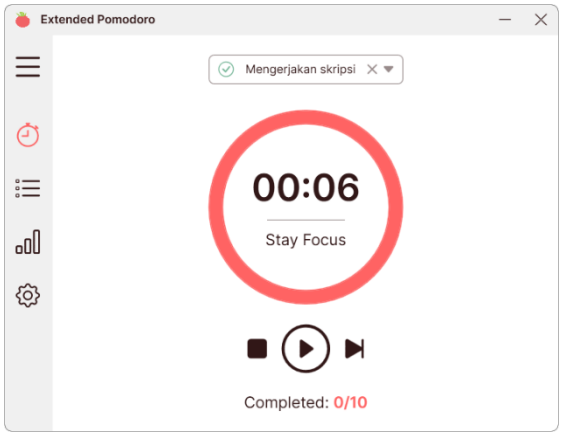
Gambar 2. Desain halaman sesi

4.3. Implementasi

Pada tahap implementasi dilakukan perancangan aplikasi ini berdasarkan hasil analisis dan desain yang telah dilakukan. Hasil dari perancangan aplikasi ini adalah sebagai berikut.

4.4. Halaman sesi

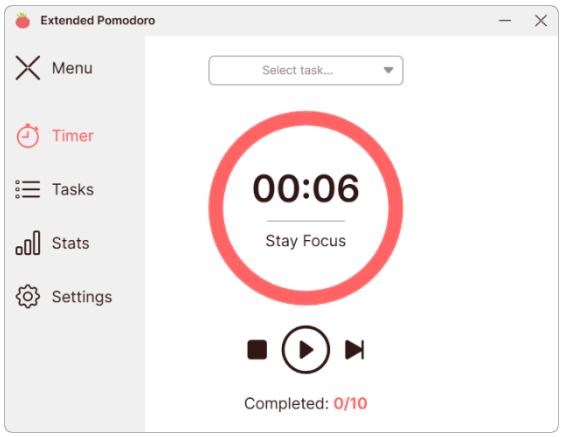
Pada halaman sesi terdapat waktu fokus, waktu istirahat pendek, dan waktu istirahat panjang. Pada waktu fokus, pengguna melakukan suatu tugas selama durasi waktu tertentu. Sementara pada waktu lainnya, pengguna beristirahat selama durasi waktu tertentu sesuai konfigurasi yang dilakukan.



Gambar 3. Halaman sesi

4.5. Halaman menu

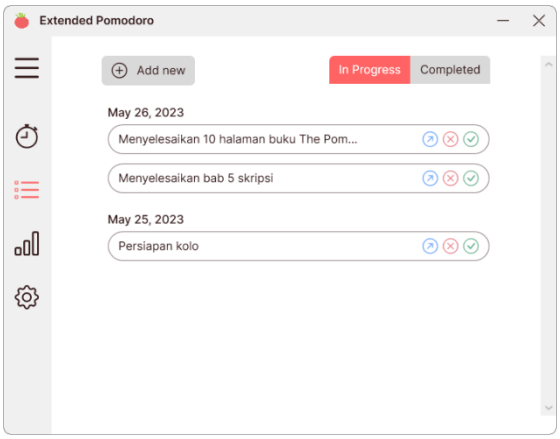
Menu digunakan untuk navigasi pengguna. Dalam aplikasi ini, terdapat dua kondisi menu yaitu tertutup dan terbuka. Menu dapat dibuka dengan mengklik tombol hamburger menu dan dapat ditutup kembali dengan mengklik tombol silang.



Gambar 4. Halaman menu

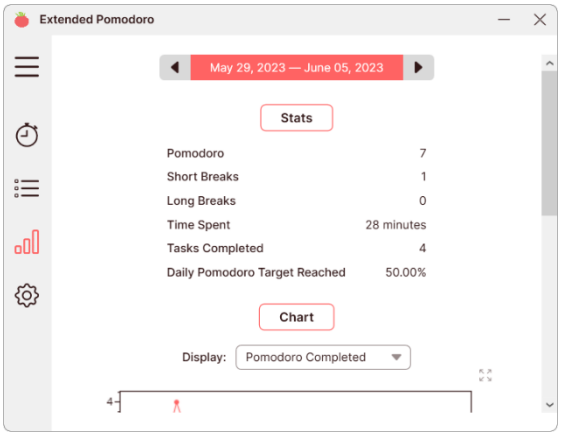
4.6. Halaman tugas-tugas

Halaman tugas-tugas adalah tempat bagi pengguna untuk melakukan sejumlah operasi pada tugas-tugas seperti melihat, menambah, mengedit, dan menghapus tugas. Tugas-tugas disortir berdasarkan tanggal pembuatan.

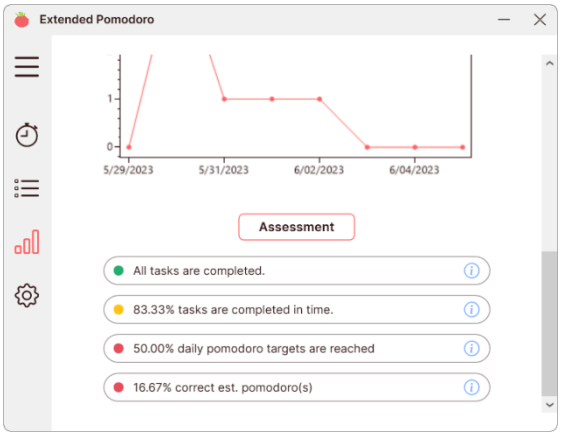


Gambar 5. Halaman tugas-tugas

4.7. Halaman statistik



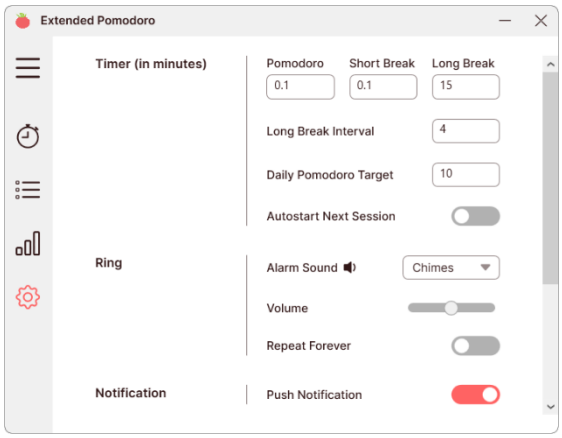
Gambar 6. Halaman statistik



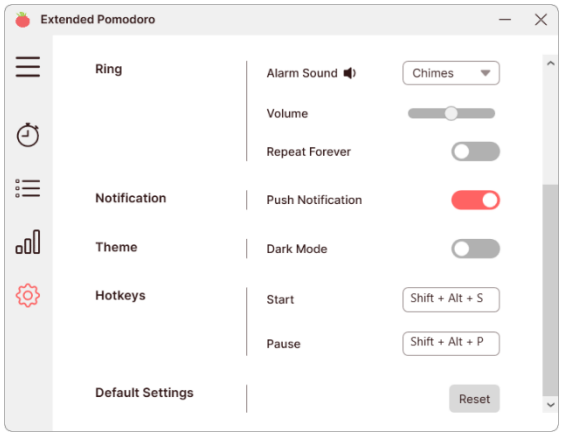
Gambar 7 Halaman statistik (gulir ke bawah)

4.8. Halaman pengaturan

Halaman pengaturan merupakan tempat bagi pengguna untuk mengubah suatu pengaturan (contohnya mengubah durasi waktu fokus atau interval istirahat panjang).



Gambar 8. Halaman pengaturan



Gambar 9. Halaman pengaturan (gulir ke bawah)

4.9. Pengujian

Pertama-tama, aplikasi diuji dengan pengujian alpha yaitu penulis menguji aplikasi sendiri untuk memastikan aplikasi berjalan sesuai ekspektasi. Setelah semua operasional diuji dan berjalan sebagaimana mestinya, dilakukan pengujian beta.

Pada pengujian beta, dilakukan pengumpulan sejumlah responden untuk mencoba menggunakan aplikasi ini. Adapun medium atau wadah yang digunakan untuk pengujian ini adalah wawancara secara daring *(online)* melalui Google Meet. Wawancara dilakukan dengan pertama-tama memberikan penjelasan dan instruksi penggunaan aplikasi kepada pengguna. Setelahnya, diajukan sejumlah kuisioner yang terdiri dari lima buah pertanyaan dengan skala Likert untuk meminta penilaian dari pengguna. Di akhir sesi, pengguna juga diminta untuk memberikan masukan *(feedback)* untuk perbaikan dan pengembangan aplikasi ini ke depannya.

Responden terdiri dari empat orang dengan latar belakang mahasiswa IT dan pekerja media. Tabel 1 menampilkan hasil kuesioner ini.

Tabel 1. Hasil kuisoner

Responden	Nomor Pertanyaan					Total	Maks
	1	2	3	4	5		
R1	5	5	5	5	4	24	25
R2	5	5	3	4	3	20	25
R3	5	4	4	5	4	22	25
R4	5	4	5	5	3	22	25
Jumlah						88	100

Tabel 2 menampilkan pertanyaan-pertanyaan yang diajukan pada setiap wawancara dengan responden.

Tabel 2. Pertanyaan-pertanyaan pada pengujian beta

No.	Pertanyaan
1	Apakah instalasi aplikasi ini mudah?
2	Apakah aplikasi ini mudah dimengerti?
3	Apakah aplikasi ini dapat memudahkan Anda dalam manajemen waktu?
4	Apakah tampilan aplikasi ini mudah dinavigasi?
5	Apakah Anda terbantu dengan aplikasi ini?

Berdasarkan penilaian yang telah diberikan seperti pada tabel 4.21, didapatkan jumlah skor 66 dari 75 jumlah skor maksimal. Skor tersebut akan dimasukkan ke dalam rumus sebagai berikut.

$$Y = \frac{\sum(N \,.\, R)}{Skor\ ideal} \times 100\%$$

$$Y = \frac{88}{100} \times 100\% = 88\%$$

Hasil perhitungan persentanse adalah 88% dari 100%. Hasil ini kemudian diinterpretasikan seperti pada tabel 3 sebagai berikut.

Tabel 3. Interpretasi skala Likert

No.	Persentase Kelayakan	Interpretasi
1.	81% –100%	Sangat setuju
2.	61% –80%	Setuju
3.	41% –60%	Cukup setuju
4.	21% –40%	Kurang setuju
5.	0% –20%	Tidak setuju

Hasil interpretasi yang didapatkan adalah sangat setuju. Artinya, secara umum aplikasi ini dapat berjalan dengan baik dan memenuhi kebutuhan penggunanya.

4.10. Pemeliharaan

Setelah melalui tahap pengujian, aplikasi melalui tahap pemeliharaan. Pada tahap ini, aplikasi akan dipaketkan dengan sebuah alat pemaketan bernama *Inno Setup*. Hasil dari pemaketan ini adalah dua buah paket instalasi untuk perangkat Windows berbasis 32 bit dan 64 bit dengan versi 1.0.0.

Setelah aplikasi dipaketkan, langkah selanjutnya adalah memublikasikan aplikasi ke GitHub sebagai proyek berlisensi terbuka *(open-source project).* Repositori proyek ini dapat diakses pada tautan berikut https://github.com/ihsansfd/extended-pomodoro. Terakhir, aplikasi dipublikasi ke sebuah halaman statis sehingga dapat diunduh dan dicoba langsung oleh calon pengguna aplikasi ini dengan mudah. Hasil halaman statis ini dapat diakses pada tautan berikut https://ihsansfd.github.io/extended-pomodoro-web/.

5. KESIMPULAN DAN SARAN

Proyek ini dirancang dengan metodologi penelitian waterfall yang terdiri dari lima tahap yaitu analisis, desain rancangan sistem, implementasi, pengujian, dan pemeliharaan. Pada analisis, dilakukan analisis kebutuhan-kebutuhan yang diperlukan untuk kelancaran perancangan proyek. Pada desain rancangan sistem, dilakukan desain tampilan aplikasi dengan Figma dan kelakukan aplikasi dengan UML. Pada implementasi, aplikasi dirancang dengan sebuah kerangka kerja bernama Windows Presentation Foundation dengan menerapkan arsitektur MVVM. Pada pengujian, aplikasi yang telah dirancang diuji melalui pengujian alpha dan pengujian beta. Terakhir, pada pemeliharaan, aplikasi yang sudah lulus uji dipublikasikan ke repositori GitHub dan sebuah halaman statis sebagai proyek berlisensi terbuka. Selain itu, aplikasi terus diperbarui dan dipelihara apabila ditemukan kekurangan-kekurangan dari masukan-masukan pengguna.

Pengujian beta *(beta testing)* dilakukan dengan wawancara langsung yang melibatkan tiga orang responden dan menghasilkan hasil skor 88%. Berdasarkan skala Likert, dapat disimpulkan bahwa hasil skor tersebut menandakan aplikasi dapat berjalan dengan baik dan sesuai dengan kebutuhan pengguna.

Saran-saran yang dapat diberikan untuk pengembangan aplikasi ini selanjutnya yaitu sebagai berikut. Menggunakan sebuah kerangka kerja yang lintas platform sehingga aplikasi ini dapat berjalan di platform-platform yang lebih banyak, tidak hanya di Windows. Menerapkan sistem penyimpanan awan *(cloud storage)* sehingga data-data pengguna dapat diakses kapan saja dan di mana saja. Objek-objek penilaian pada halaman statistik yang sekiranya berguna bagi pengguna dapat ditambahkan lagi. Menerapkan desain aplikasi yang dapat diperbesar *(maximizable).*

DAFTAR PUSTAKA

[1] J. B. Spira and J. B. Feintuch, "The Cost of Not Paying Attention: How Interruptions Impact Knowledge Worker Productivity Chief Analyst," Basex, 2005

[2] M. Czerwinski, E. Cutrell and E. Horvitz, "Instant Messaging: Effects of Relevance and Timing," Microsoft Research, 2000

[3] U. o. I. a. Urbana-Champaign, "Brief diversions vastly improve focus, researchers find," ScienceDaily, 2011. [Online]. Available: https://www.sciencedaily.com/releases/2011/02/110208131529.htm. [Accessed 14 03 2023]

[4] K. Weir, "Never a dull moment," 2013. [Online]. Available: https://www.apa.org/monitor/2013/07-08/dull-moment. [Accessed 14 03 2023].

[5] K. Diaz, A. T. Duran, C. P. Friel, C. P. Friel, C. P. Friel and Y. K. Cheung, "Breaking Up Prolonged Sitting to Improve Cardiometabolic Risk: Dose-Response Analysis of a Randomized Cross-Over Trial," 2023. [Online]. Available: https://journals.lww.com/acsm-msse/Abstract/9900/Breaking_Up_Prolonged_Sitting_to_Improve.200.aspx. [Accessed 14 03 2023].

[6] F. Cirillo, The Pomodoro Technique (The Pomodoro), 2006

[7] E. International, "C# Language Specification," ECMA, 2006

[8] Oracle, "What Is a Database?," Oracle, [Online]. Available: https://www.oracle.com/database/what-is-database/. [Accessed 15 03 2023].

[9] A. Nathan, Windows Presentation Foundation Unleashed, Pearson, 2006

[10] Microsoft, "A tour of the C# language," Microsoft, 2023. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Accessed 14 03 2023]

[11] J. Smith, "Patterns - WPF Apps With The Model-View-ViewModel Design Pattern," [Online]. Available: https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern. [Accessed 16 03 2023]

[12] J. Gossman, "Introduction to Model/View/ViewModel pattern for building WPF apps," 2005. [Online]. Available: https://learn.microsoft.com/en-us/archive/blogs/johngossman/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps. [Accessed 16 03 2023]

[13] J. Smith, "WPF vs. Windows Forms," Microsoft, 2007. [Online]. Available: https://joshsmithonwpf.wordpress.com/2007/09/05/wpf-vs-windows-forms/. [Accessed 15 03 2023]

[14] "SQLite," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/SQLite. [Accessed 15 03 2023]

[15] Guru99, "Unit Testing Tutorial – What is, Types & Test Example," Guru99, 2023. [Online]. Available: https://www.guru99.com/unit-testing-guide.html. [Accessed 15 03 2023]

[16] S. S. Indonesia, "Memahami Apa Bedanya Beta Testing & Alpha Testing," Startup Studio Indonesia, 2021. [Online]. Available: https://startupstudio.id/beta-testing/. [Accessed 15 03 2023]

[17] R. Rachlin, "How many beta testers do I need to create the perfect app?," UberTesters, 2016. [Online]. Available: https://ubertesters.com/blog/how-many-beta-testers-do-i-need-to-create-the-perfect-app/. [Accessed 10 07 2023]

[18] CMS, "SELECTING A DEVELOPMENT APPROACH," CMS, 2005. [Online]. Available: https://web.archive.org/web/20190828154643/https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf. [Accessed 15 03 2023]

[19] D. A. Hughey, "The Traditional Waterfall Approach," [Online]. Available: https://www.umsl.edu/~hugheyd/is6840/waterfall.html. [Accessed 15 03 2023]