LAPORAN PROJECT

PEMROGRAMAN BERORIENTASI OBJEK

PROJECT POMOCAT

\

DISUSUN OLEH:

Aprilia Wulandari	**L0124040**
Aisyah Nurul Sholikhah	**L0124085**
Calista Salsabila	**L0124092**
Keisha Nasywa Syakira	**L0124102**

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

2025

BAB I
PENDAHULUAN

1.1 Latar Belakang

Aktivitas seperti mengerjakan tugas, belajar, dan menyelesaikan pekerjaan dalam jangka waktu panjang sering kali menimbulkan kejenuhan serta penurunan tingkat konsentrasi. Kondisi tersebut berdampak pada menurunnya efektivitas penyelesaian pekerjaan dan dapat menghambat produktivitas individu. Untuk mengatasi masalah tersebut, berbagai metode manajemen waktu dikembangkan guna membantu seseorang mempertahankan fokus dan mengelola energi kerja secara lebih terstruktur.

Salah satu teknik yang telah terbukti efektif dan banyak digunakan adalah metode Pomodoro. Metode ini membagi waktu kerja ke dalam siklus fokus terkontrol, misalnya 25 menit bekerja intensif yang kemudian diikuti jeda istirahat selama 5 menit. Pola tersebut membantu menjaga ritme kerja, mengurangi kelelahan mental, dan meningkatkan kemampuan individu dalam mempertahankan konsentrasi dalam periode yang konsisten.

Namun, penggunaan *timer* Pomodoro konvensional sering kali tidak cukup untuk mempertahankan motivasi jangka panjang. Banyak pengguna kesulitan konsisten karena kurangnya elemen umpan balik atau insentif yang membuat proses menjadi lebih menarik. Tantangan inilah yang melatarbelakangi pengembangan PomoCat, sebuah aplikasi timer berbasis metode Pomodoro yang dipadukan dengan pendekatan gamifikasi melalui karakter virtual pet.

Aplikasi Pomopet tidak hanya berfungsi sebagai pengatur waktu, tetapi juga menyediakan sistem *reward*, pemantauan progres, dan interaksi dengan *pet* sebagai bentuk motivasi tambahan. Dengan mengintegrasikan teknik Pomodoro, ta*sk management*, dan unsur permainan, aplikasi ini diharapkan mampu membantu pengguna mempertahankan fokus, membangun kebiasaan kerja yang lebih konsisten, serta meningkatkan produktivitas dalam aktivitas sehari-hari.

1.2 Permasalahan

Pada penerapannya, penggunaan *timer* Pomodoro konvensional tidak selalu mampu mempertahankan disiplin dan konsistensi pengguna. *Timer* hanya berfungsi

sebagai alat hitung waktu tanpa memberikan insentif tambahan ataupun keterlibatan emosional. Selain itu, proses pencatatan dan penyelesaian tugas harian sering kali berjalan terpisah dari penggunaan *timer*, sehingga alur kerja menjadi tidak terintegrasi dan progres sulit dipantau secara menyeluruh.

Dari sisi pengalaman pengguna, tidak adanya *feedback* yang bersifat menarik membuat aktivitas kerja terasa monoton. Padahal, pengguna cenderung lebih termotivasi ketika terdapat elemen visual, reward, atau interaksi yang memberi rasa pencapaian. Hal ini menunjukkan adanya kebutuhan akan sistem yang tidak hanya menghitung waktu, tetapi juga mampu meningkatkan keterlibatan dan motivasi selama bekerja. Berdasarkan kondisi tersebut, masalah utama yang ingin diselesaikan adalah:

a. Minimnya motivasi jangka panjang saat menggunakan *timer* Pomodoro biasa.

b. Terpisahnya manajemen tugas dari mekanisme *timer* sehingga progres kurang terhubung.

c. Tidak adanya elemen gamifikasi yang memberikan reward langsung terhadap perilaku produktif.

d. Kurangnya sistem umpan balik yang membuat proses kerja terasa lebih menarik dan terarah.

1.3 Penyelesaian

Solusi yang dikembangkan untuk mengatasi permasalahan ini adalah membangun sebuah aplikasi manajemen aktivitas hewan peliharaan berbasis Java dengan menerapkan konsep *abstract class*, *interface*, dan *multithreading*. Pendekatan ini dipilih untuk memastikan sistem memiliki struktur yang terorganisasi, fleksibel untuk dikembangkan, serta mampu memproses beberapa aktivitas secara paralel tanpa mengganggu responsivitas aplikasi.

Pertama, aplikasi dirancang menggunakan abstract class Pet sebagai dasar bagi setiap jenis hewan. Pendekatan ini memberikan standar atribut dan perilaku minimal, sehingga pencatatan data hewan menjadi lebih konsisten. Selain itu, interface *Task* digunakan untuk mendefinisikan berbagai tugas perawatan seperti memberi makan

dan bermain. Pemisahan ini menghasilkan struktur kode yang modular dan mudah diperluas apabila jenis tugas bertambah di kemudian hari.

Selanjutnya, permasalahan pemantauan waktu diselesaikan melalui implementasi kelas TimerModel, yang bertugas mengelola hitungan waktu setiap aktivitas. Pemisahan logika waktu ke dalam kelas khusus mencegah konflik fungsi dan memastikan setiap proses dapat berjalan dengan akurat tanpa mengganggu alur kerja lainnya.

Untuk menjaga aplikasi tetap responsif, sistem menerapkan *multithreading* pada beberapa proses, khususnya *timer* dan eksekusi tugas yang berjalan di latar belakang. Dengan adanya pemrosesan paralel, pengguna tetap dapat melakukan *input* atau mengelola hewan peliharaan tanpa terjadi *freeze* atau hambatan pada antarmuka.

Terakhir, seluruh komponen diintegrasikan melalui MainController, yang mengatur koordinasi antara data hewan, tugas, dan *timer*. Struktur kontrol terpusat ini memastikan alur eksekusi lebih teratur serta mempermudah pengembangan maupun pemeliharaan aplikasi.

BAB II
DASAR TEORI

2.1 Konsep Object Oriented Programming

Pemrograman Berorientasi Objek (*Object-Oriented Programming*/OOP) merupakan paradigma pengembangan perangkat lunak yang memodelkan sistem sebagai kumpulan objek yang saling berinteraksi. Setiap objek merepresentasikan entitas yang memiliki keadaan (atribut) dan perilaku (metode). Dalam proyek PomoCat, pendekatan ini digunakan untuk membangun arsitektur aplikasi yang modular, terstruktur, serta mudah dikembangkan lebih lanjut.

2.1.1 Class dan Object

Class berperan sebagai *blue print* yang mendefinisikan atribut dan metode dari suatu entitas. *Object* adalah instans konkret dari *class* tersebut yang hidup selama eksekusi program. Pada PomoCat, beberapa class utama yang membentuk logika aplikasi antara lain:

- Task digunakan untuk merepresentasikan tugas yang dikelola pengguna, lengkap dengan status dan parameter waktunya.
- Pet digunakan untuk merepresentasikan karakter virtual yang bereaksi terhadap progres pengguna.
- TimerModel digunakan untuk menangani logika perhitungan waktu, termasuk durasi pomodoro, istirahat, dan transisi status.

Relasi antarclass ini menciptakan struktur aplikasi yang jelas, di mana tiap objek memiliki tanggung jawab terpisah namun saling mendukung.

2.1.2 Enkapsulasi (Encapsulation)

Enkapsulasi adalah prinsip menyatukan data dan operasi yang bekerja pada data tersebut dalam satu unit, sekaligus membatasi akses langsung dari luar. Manfaatnya yaitu menjaga integritas data, karena perubahan hanya dapat dilakukan melalui metode yang telah ditentukan dan mengurangi ketergantungan, sehingga perubahan di dalam class tidak merusak modul lain.

Di PomoCat, atribut internal seperti status *timer*, jumlah sesi, atau kondisi *pet* dienkapsulasi dalam *class* masing-masing dan hanya dapat dimanipulasi melalui metode yang telah digunakan.

2.1.3 Pewarisan (Inheritance)

Pewarisan memungkinkan sebuah *class* baru (*child*) memperoleh atribut dan *method* dari *class* lain (*parent*), sehingga kode yang umum dapat digunakan kembali. Prinsip ini mempermudah ekstensi perilaku tanpa menyalin kode dan penataan hierarki logis antar entitas.

2.1.4 Polimorfisme (Polymorphism)

Polimorfisme memberikan kemampuan bagi objek untuk merespons pesan yang sama dengan cara yang berbeda, bergantung pada tipe objeknya. Polimorfisme diterapkan melalui dua *method*, yaitu *method overriding* dan *method overloading*.

2.1.5 Abstraksi (Abstraction)

Abstraksi digunakan untuk menyederhanakan kompleksitas sistem dengan hanya menampilkan detail yang relevan kepada pengguna dan menyembunyikan logika rumit di belakang layar.

2.2 Konsep Pengembangan Aplikasi

2.2.1 Arsitektur Model-View-Controller (MVC)

PomoCat dibangun menggunakan pola desain MVC yang memisahkan aplikasi menjadi tiga komponen utama:

1. **Model**

   Merepresentasikan data dan logika bisnis (Task.java, Pet.java, TimerModel.java).

2. **View**

   Merepresentasikan antarmuka pengguna atau GUI (*file* FXML seperti main_view.fxml, welcome.fxml, serta *file* CSS seperti style.css).

3. **Controller**

   Menghubungkan Model dan View, serta menangani *input* pengguna (MainController.java). Pemisahan ini memudahkan pemeliharaan kode (*maintainability*) dan pengembangan fitur baru.

2.2.2 JavaFX dan Data Binding

JavaFX merupakan *framework* antarmuka grafis yang menyediakan komponen UI modern, sistem *event* yang konsisten, serta dukungan bawaan untuk pemisahan antara logika bisnis dan tampilan. Salah satu fitur yang

paling relevan dalam konteks proyek PomoCat adalah penggunaan JavaFX Properties dan Data Binding.

JavaFX Properties (seperti StringProperty, IntegerProperty, atau BooleanProperty) memungkinkan setiap perubahan nilai dapat terdeteksi secara otomatis oleh elemen UI maupun komponen lain dalam aplikasi.

Data Binding menyediakan mekanisme untuk menghubungkan nilai antar komponen, misalnya antara model (seperti TimerModel) dan elemen tampilan (seperti label penghitung waktu). Ketika nilai pada model berubah, UI diperbarui tanpa perlu pemanggilan manual.

Pendekatan ini menciptakan alur data yang reaktif dan konsisten, mengurangi kerentanan terhadap bug akibat sinkronisasi data, serta memisahkan tanggung jawab antara model, *controller*, dan *view* secara lebih tegas. Pada PomoCat, binding digunakan untuk menghubungkan nilai penghitung waktu dengan tampilan *timer*, mengikat status *task* dengan daftar tampilan tugas, dan melakukan *update* reaksi *pet* secara otomatis ketika progres pengguna berubah. Dengan demikian, JavaFX tidak hanya bertindak sebagai *framework* GUI, tetapi juga sebagai mekanisme yang memastikan perubahan logika bisnis langsung tercermin dalam antarmuka pengguna.

2.2.3 Persistensi Data (File Handling)

Persistensi data adalah kemampuan sebuah aplikasi untuk menyimpan dan mempertahankan informasi sehingga data tersebut tetap tersedia meskipun aplikasi sudah ditutup atau perangkat dimatikan. Tanpa mekanisme ini, seluruh data yang tersimpan di memori selama aplikasi berjalan akan hilang begitu proses berhenti. Dalam konteks PomoCat, persistensi diperlukan untuk menjaga agar daftar *task* pengguna tidak hilang setiap kali aplikasi ditutup, serta progres dan status timer tetap tercatat,

Untuk memenuhi kebutuhan itu, proyek ini menggunakan pendekatan file handling berbasis teks. Teknik ini merupakan bentuk persistensi paling dasar, di mana data diubah menjadi representasi tekstual, kemudian ditulis ke dalam *file* di penyimpanan lokal. Ketika aplikasi dijalankan kembali, maka *file* dibaca, data diparsing, dan objek-objek aplikasi direkonstruksi dari isi *file* tersebut. Pendekatan ini cukup untuk aplikasi sederhana yang tidak membutuhkan query kompleks, transaksi, atau efisiensi penyimpanan tingkat lanjut.

BAB III
ARSITEKTUR SISTEM DAN PENJELASAN CODE

3.1 Fitur Utama

3.2 Penjelasan Code

3.2.1 Task.java

```java
package com.pomodoro.model;

import javafx.beans.property.*;

public class Task {
    private final StringProperty title;
    private final BooleanProperty isCompleted;

    public Task(String title) {
        this.title = new SimpleStringProperty(title);
        this.isCompleted = new SimpleBooleanProperty(false);
    }

    public StringProperty titleProperty() { return title; }
    public BooleanProperty isCompletedProperty() { return isCompleted; }

    public String getTitle() { return title.get(); }
    public boolean isCompleted() { return isCompleted.get(); }

    @Override
    public String toString() {
        return getTitle();
    }

    //Supaya bisa ganti nama tugas kalau mau edit
    public void setTitle(String title) { this.title.set(title); }

    //Supaya checkbox bisa mengubah status true/false
    public void setCompleted(boolean completed) { this.isCompleted.set(completed); }
}
```

Kode di atas digunakan untuk membuat kelas model Task yang merepresentasikan satu tugas dalam aplikasi PomoCat. Kelas ini menyimpan dua informasi utama yaitu judul tugas “title” dan status selesai “isCompleted”. Keduanya menggunakan JavaFX Property (StringProperty dan Boolean Property) agar dapat terhubung langsung dengan elemen UI seperti *TextField*, *Label*, atau *CheckBox*, sehingga perubahan data otomatis terlihat pada tampilan.

Selain itu, kelas ini menggunakan *getter*, *setter*, serta *method* *“*toString()” yang mengembalikan judul tugas. *Method* *“*setTitle()” digunakan untuk mengedit nama tugas, sementara itu “setCompleted()” untuk mengubah status tugas ketika *checkbox* ditandai.

3.2.2 Pet.java

```java
package com.pomodoro.model;

import javafx.beans.property.*;

public class Pet {
    private IntegerProperty happiness = new SimpleIntegerProperty(0);
    private IntegerProperty coins = new SimpleIntegerProperty(0);

    public IntegerProperty happinessProperty() { return happiness; }
    public IntegerProperty coinsProperty() { return coins; }

    public void earnReward(int amount) {
        coins.set(coins.get() + amount);
    }

    public boolean play(int cost, int happinessGain) {
        if (coins.get() >= cost) {
            coins.set(coins.get() - cost);
            increaseHappiness(happinessGain);
            return true;
        }
        return false;
    }

    public boolean increaseHappiness(int amount) {
        int current = happiness.get() + amount;

        if (current >= 100) {
            //Kalau penuh reset ke 20
            happiness.set(20);
            coins.set(coins.get() + 100); //Bonus Prestige
            return true;
        } else {
            happiness.set(current);
            return false;
        }
    }

    public void decreaseHappiness() {
        if (happiness.get() > 0) happiness.set(happiness.get() - 10);
    }
}
```

Kode di atas digunakan untuk membuat kelas model Pet yang mewakili hewan virtual dalam aplikasi. Kelas ini menyimpan dua atribut yang menggunakan JavaFX Property, yaitu “happiness” dan “coins”, di mana nilai keduanya terhubung langsung ke elemen UI, seperti label atau progress bar, sehingga perubahan data otomatis terlihat di tampilan.

Kelas ini juga meggunakan *method “*earnReward()” untuk mendapatkan hadiah, “play()” untuk bermain dengan biaya koin yang telah didapatkan, “increaseHappiness()” untuk menaikkan kebahagiaan, serta “decreaseHappiness()” untuk mengurangi kebahagian. Jika *happiness* mencapai atau melebihi 100, nilainya akan di-*reset ke 20* dan *user* menerima bonus 100 koin sebagai *Prestige reward*.

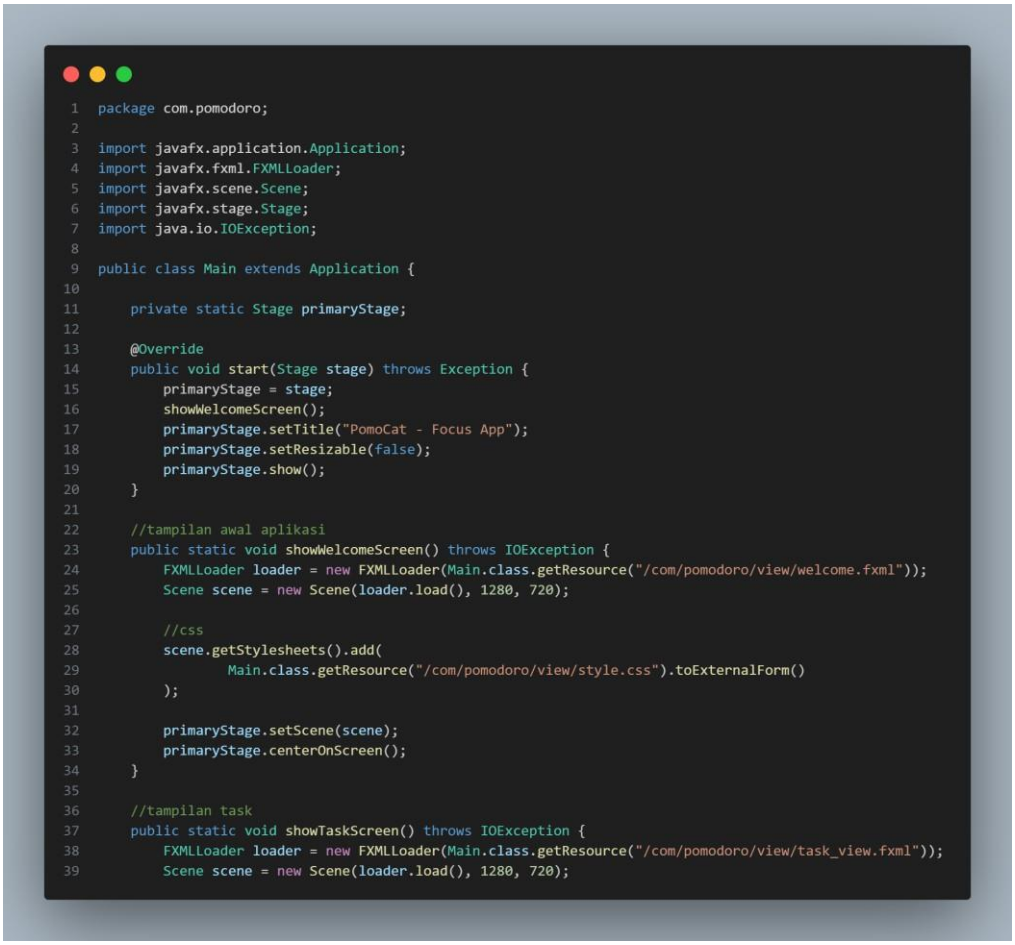
3.2.3 TimerModel.java

```java
package com.pomodoro.model;

import javafx.beans.property.*;

public class TimerModel {
    // Default awal (bisa diubah nanti)
    private int currentStartTime = 25 * 60;

    private IntegerProperty timeSeconds = new SimpleIntegerProperty(currentStartTime);
    private BooleanProperty isRunning = new SimpleBooleanProperty(false);

    public IntegerProperty timeSecondsProperty() { return timeSeconds; }
    public BooleanProperty isRunningProperty() { return isRunning; }

    public int getTimeSeconds() { return timeSeconds.get(); }
    public boolean isRunning() { return isRunning.get(); }

    public void setRunning(boolean running) { this.isRunning.set(running); }

    // --- TAMBAHAN BARU: FITUR CHOOSE TIME ---
    public void setStartTime(int seconds) {
        this.currentStartTime = seconds;
        this.timeSeconds.set(seconds); // Update waktu saat ini juga
    }

    public void decrementTime() {
        if (timeSeconds.get() > 0) {
            timeSeconds.set(timeSeconds.get() - 1);
        }
    }

    public void reset() {
        timeSeconds.set(currentStartTime); // Reset kembali ke waktu yang dipilih user
        setRunning(false);
    }

    public enum SessionMode {
        WORK, BREAK
    }
}
```

Kode di atas digunakan untuk membuat kelas model TimerModel yang mengatur penghitung waktu dalam aplikasi PomoCat. Kelas ini menyimpan dua atribut, yaitu “timeSeconds” untuk menyimpan waktu dalam detik dan “isRunning” untuk menyimpan status *timer* yang keduanya menggunakan JavaFX Property sehingga perubahannya langsung muncul pada UI seperti label atau progress bar. Nilai *timer* awal disimpan dalam “currentStartTime” dan dapat dipilih sesuai pilihan *user*.

Selain, terdapat beberapa *method* yang digunakan, seperti “setStartTime()” untuk mengatur waktu mulai, “decrementTime()” utnuk menghitung waktu mundur, dan “reset()” untuk mereset *timer* kembali ke waktu awal. Terdapat juga *enum* “SessionMode” yang membedakan mode kerja dan istirahat.

3.2.4 Main.java



```java
package com.pomodoro;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;
import java.io.IOException;

public class Main extends Application {

    private static Stage primaryStage;

    @Override
    public void start(Stage stage) throws Exception {
        primaryStage = stage;
        showWelcomeScreen();
        primaryStage.setTitle("PomoCat - Focus App");
        primaryStage.setResizable(false);
        primaryStage.show();
    }

    //tampilan awal aplikasi
    public static void showWelcomeScreen() throws IOException {
        FXMLLoader loader = new FXMLLoader(Main.class.getResource("/com/pomodoro/view/welcome.fxml"));
        Scene scene = new Scene(loader.load(), 1280, 720);

        //css
        scene.getStylesheets().add(
                Main.class.getResource("/com/pomodoro/view/style.css").toExternalForm()
        );

        primaryStage.setScene(scene);
        primaryStage.centerOnScreen();
    }

    //tampilan task
    public static void showTaskScreen() throws IOException {
        FXMLLoader loader = new FXMLLoader(Main.class.getResource("/com/pomodoro/view/task_view.fxml"));
        Scene scene = new Scene(loader.load(), 1280, 720);
```

Kode di atas digunakan sebagai titik awal eksekusi aplikasi PomoCat. Kelas Main menyimpan satu “primaryStage” yang dipakai untuk menampilkan tiga halaman utama, yaitu welcome screen, task screen, dan timer screen. Setiap halaman dimuat menggunakan FXMLLoader, diberi ukuran tetap 1280x720, serta memakai stylesheet (style.css) yang sama. Selain mengatur navigasi antarlayar, kelas ini juga menjalankan aplikasi melalui method “main()” dan menampilkan layar pertama saat aplikasi dibuka.

3.2.5 task_view.fxml

a. Root Container

VBox digunakan sebagai kontainer utama pada tampilan. Komponen–komponen di dalamnya disusun secara vertikal dari atas ke bawah, sehingga struktur layout menjadi terorganisasi dan mudah diatur.

Atribut fx:controller pada berkas FXML digunakan untuk menghubungkan tampilan dengan kelas pengontrol pada Java. Melalui keterhubungan ini, elemen antarmuka dapat berinteraksi dengan logika aplikasi yang diimplementasikan dalam MainController.

3.2.5.1 Combobox

ComboBox digunakan sebagai elemen pemilih (dropdown) untuk menentukan durasi sesi timer, seperti 10 detik, 25 menit, atau 50 menit. Komponen ini memungkinkan pengguna memilih nilai waktu tanpa perlu melakukan input manual.

Atribut fx:id berfungsi sebagai pengenal unik bagi elemen FXML. Identitas ini menghubungkan komponen ComboBox dengan variabel yang dideklarasikan di kelas pengontrol melalui anotasi @FXML. Dengan demikian, elemen antarmuka dapat diakses dan dikendalikan secara langsung dari MainController.

3.2.5.2 TextField & Button

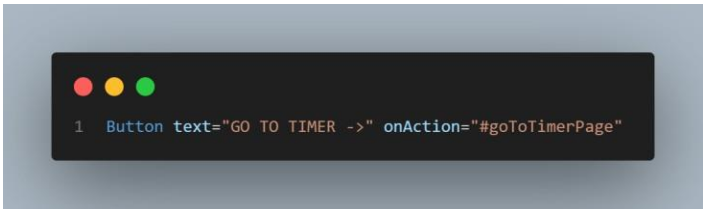
TextField digunakan sebagai komponen *input* untuk memasukkan nama tugas. Elemen ini menyediakan area teks satu baris yang memungkinkan pengguna mengetikkan data secara langsung ke dalam sistem.

Atribut onAction mendefinisikan metode yang dipanggil ketika sebuah aksi terjadi pada komponen, misalnya ketika pengguna menekan tombol Enter. Nilai #handleAddTask menghubungkan event tersebut dengan metode penanganan kejadian di kelas pengontrol, sehingga proses penambahan tugas dapat dijalankan secara otomatis.

3.2.5.3 ListView

ListView berfungsi sebagai komponen tampilan daftar yang dapat digulir. Elemen ini menampilkan kumpulan objek tugas dalam bentuk baris–baris vertikal, sehingga pengguna dapat melihat seluruh tugas yang tersimpan dengan mudah. Atribut fx:id="taskListView" memberikan identitas unik pada komponen tersebut. Melalui identitas ini, kelas pengontrol dapat mengakses dan memanipulasi isi ListView, termasuk menampilkan data tugas yang berasal dari struktur data bersama (sharedTasks).

3.2.5.4 Tombol Navigasi (Pindah Halaman)

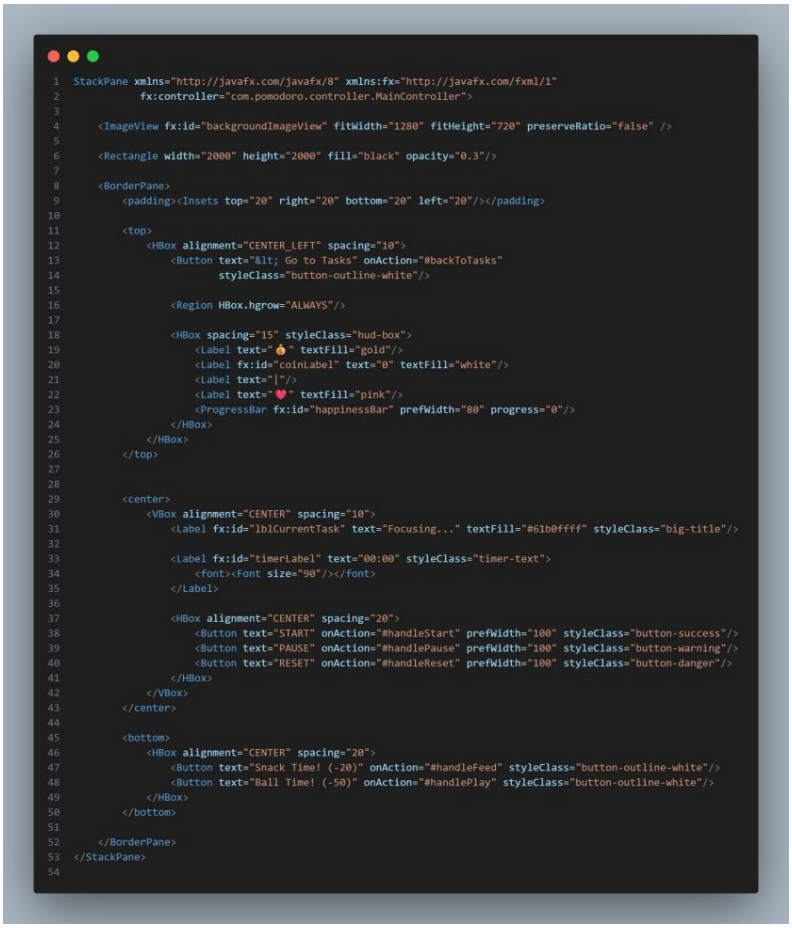


```
1  Button text="GO TO TIMER ->" onAction="#goToTimerPage"
```

Atribut “onAction=”#goToTimerPage” digunakan untuk mangaitkan tombol dengan metode “goToTimePage()” pada *controller*, yang berfungsi melakukan navigasi menuju halaman *timer* ketika tombol diaktifkan oleh *user*.

3.2.6 timer_view.fxml

Kode ini merepresentasikan halaman utama yang digunakan untuk memantau proses fokus. Halaman ini menampilkan durasi sesi, status timer, serta elemen terkait lainnya yang berfungsi sebagai *dashboard* selama pengguna menjalankan aktivitas fokus.



```
StackPane xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1"
          fx:controller="com.pomodoro.controller.MainController">

    <ImageView fx:id="backgroundImageView" fitWidth="1280" fitHeight="720" preserveRatio="false" />

    <Rectangle width="2000" height="2000" fill="black" opacity="0.3"/>

    <BorderPane>
        <padding><Insets top="20" right="20" bottom="20" left="20"/></padding>

        <top>
            <HBox alignment="CENTER_LEFT" spacing="10">
                <Button text="&lt; Go to Tasks" onAction="#backToTasks"
                        styleClass="button-outline-white"/>

                <Region HBox.hgrow="ALWAYS"/>

                <HBox spacing="15" styleClass="hud-box">
                    <Label text="🔥" textFill="gold"/>
                    <Label fx:id="coinLabel" text="0" textFill="white"/>
                    <Label text="|"/>
                    <Label text="❤" textFill="pink"/>
                    <ProgressBar fx:id="happinessBar" prefWidth="80" progress="0"/>
                </HBox>
            </HBox>
        </top>


        <center>
            <VBox alignment="CENTER" spacing="10">
                <Label fx:id="lblCurrentTask" text="Focusing..." textFill="#61b0ffff" styleClass="big-title"/>

                <Label fx:id="timerLabel" text="00:00" styleClass="timer-text">
                    <font><Font size="90"/></font>
                </Label>

                <HBox alignment="CENTER" spacing="20">
                    <Button text="START" onAction="#handleStart" prefWidth="100" styleClass="button-success"/>
                    <Button text="PAUSE" onAction="#handlePause" prefWidth="100" styleClass="button-warning"/>
                    <Button text="RESET" onAction="#handleReset" prefWidth="100" styleClass="button-danger"/>
                </HBox>
            </VBox>
        </center>

        <bottom>
            <HBox alignment="CENTER" spacing="20">
                <Button text="Snack Time! (-20)" onAction="#handleFeed" styleClass="button-outline-white"/>
                <Button text="Ball Time! (-50)" onAction="#handlePlay" styleClass="button-outline-white"/>
            </HBox>
        </bottom>

    </BorderPane>
</StackPane>
```

e. Konsep Layering (StackPane)

- **Layer 1 (Background)**

  Lapisan pertama berupa ImageView yang menampilkan animasi GIF karakter kucing dalam berbagai keadaan (belajar, tidur, atau makan). Lapisan ini berfungsi sebagai elemen visual utama pada halaman Timer.

- **Layer 2 (Filter Gelap)**

  Lapisan kedua menggunakan elemen Rectangle berwarna hitam dengan nilai opacity sebesar 0.3. Lapisan ini berfungsi sebagai overlay untuk mereduksi intensitas visual gambar latar, sehingga elemen teks di atasnya tetap memiliki tingkat keterbacaan yang tinggi.

- **Layer 3 (Kontrol)**

  Lapisan ketiga terdiri dari sebuah BorderPane yang memuat komponen kontrol seperti tombol dan teks status. Lapisan ini merupakan area interaktif tempat pengguna melakukan pengoperasian timer dan elemen terkait lainnya.

f. Bagian Atas (<top>) - HUD & Navigasi

- **Tombol Kembali**

  Elemen <Button text="< Go to Tasks" … /> berfungsi sebagai kontrol navigasi untuk kembali ke halaman daftar tugas. Atribut onAction="#backToTasks" menghubungkan tombol ini dengan metode pengontrol yang menghentikan proses timer dan melakukan perpindahan tampilan ke halaman Task.

- **Spacer**

  Elemen <Region HBox.hgrow="ALWAYS" /> digunakan sebagai ruang fleksibel dalam layout. Komponen ini memperluas dirinya secara horizontal sehingga mendorong elemen di sisi kiri dan kanan ke posisi masing-masing, memastikan tata letak tetap proporsional.

- **Status Player (HUD)**

  Bagian ini menampilkan informasi status pengguna dalam bentuk jumlah Coins dan tingkat Happiness. Elemen-elemen ini

ditampilkan sebagai bagian dari antarmuka yang memantau perkembangan aktivitas gamifikasi.

- **fx:id="coinLabel" & fx:id="happinessBar”**

  Atribut fx:id pada elemen tersebut menghubungkan komponen tampilan dengan variabel di kelas pengontrol. Melalui keterhubungan ini, nilai Coins dan Happiness dapat diperbarui secara otomatis berdasarkan perubahan status objek sharedPet.

g. Bagian Tengah ( ) - Fokus Utama

- **Judul Tugas**

  Komponen teks ini digunakan untuk menampilkan informasi mengenai tugas yang sedang dikerjakan. Nilainya dapat berupa status umum seperti “Focusing…” atau nama tugas yang dipilih pada halaman sebelumnya.

- **Angka Timer**

  Elemen teks dengan ukuran font besar (sekitar 90 piksel) yang menampilkan waktu hitung mundur. Nilai pada label ini diperbarui secara berkala melalui mekanisme Timeline pada kelas pengontrol.

- **Tombol Kontrol**

  Tiga tombol yang disusun sejajar—Start, Pause, dan Reset—berfungsi sebagai kontrol utama untuk mengoperasikan timer. Masing-masing tombol terhubung dengan metode logika pada pengontrol yang menangani perubahan status timer.

h. Bagian Bawah (<bottom>) - Interaksi Pet

- **"Snack Time!**"

  Tombol ini terhubung dengan metode #handleFeed pada kelas pengontrol. Ketika diaktifkan, sistem menjalankan proses pemberian makanan kepada karakter virtual dan mengurangi nilai Coins sebesar 20.

- **"Ball Time!"**

  Tombol ini terhubung dengan metode #handlePlay. Aksi ini memicu proses interaksi bermain dengan karakter virtual dan mengurangi nilai Coins sebesar 50.

3.2.7 welcome.fxml

File FXML ini merepresentasikan tampilan Welcome Screen pada aplikasi PomoCat. Halaman ini berfungsi sebagai titik awal interaksi pengguna dengan aplikasi, menampilkan identitas visual aplikasi serta menyediakan tombol untuk melanjutkan ke tahap berikutnya dalam proses penggunaan

a. **Wadah Utama & Latar Belakang**

- StackPane digunakan sebagai kontainer dasar pada halaman. Komponen ini memungkinkan penumpukan elemen secara berlapis sehingga struktur tampilan dapat diatur dengan fleksibel
- Atribut style style="-fx-background-color: #2c3e50;" digunakan untuk menetapkan warna latar belakang elemen. Nilai #2c3e50 menghasilkan tampilan dengan nuansa gelap yang konsisten dan mendukung karakter visual halaman sambutan.

b. **Tata Letak Vertikal (VBox)**

- Atribut alignment mengatur posisi seluruh elemen di dalam kontainer agar terpusat secara horizontal maupun vertikal.
- Atribut spacing menetapkan jarak antar elemen di dalam kontainer sebesar 20 piksel, sehingga tata letak tetap rapi dan tidak saling berhimpitan

c. Logo Maskot (ImageView)

Elemen gambar menampilkan berkas animasi welcomeCat.gif dengan ukuran 300×300 piksel. Komponen ini

berfungsi sebagai elemen visual utama pada halaman sambutan dan merepresentasikan identitas karakter aplikasi.

d. Judul & Slogan (Label)

- Elemen teks menampilkan judul “PomoCat” dengan ukuran font 48 dan gaya cetak tebal. Komponen ini berfungsi sebagai identitas utama halaman sambutan.
- Elemen teks kedua menampilkan deskripsi singkat aplikasi dengan warna abu-abu muda (#bdc3c7). Pengaturan warna ini memastikan tingkat keterbacaan yang baik terhadap latar belakang gelap.

e. TombolMulai (Button)

Atribut onAction menentukan metode yang dieksekusi ketika tombol diaktifkan. Nilai #goToTaskPage mengaitkan tombol tersebut dengan metode pada kelas pengontrol yang berfungsi melakukan navigasi dari halaman sambutan menuju halaman Task.

f. Desain (CSS Inline):

- **fx-background-color#f1c40f;**

  Atribut ini mengatur warna latar belakang tombol menggunakan kode warna #f1c40f. Pemilihan warna tersebut memberikan kontras yang jelas terhadap latar belakang halaman sehingga elemen tombol tampil menonjol secara visual.

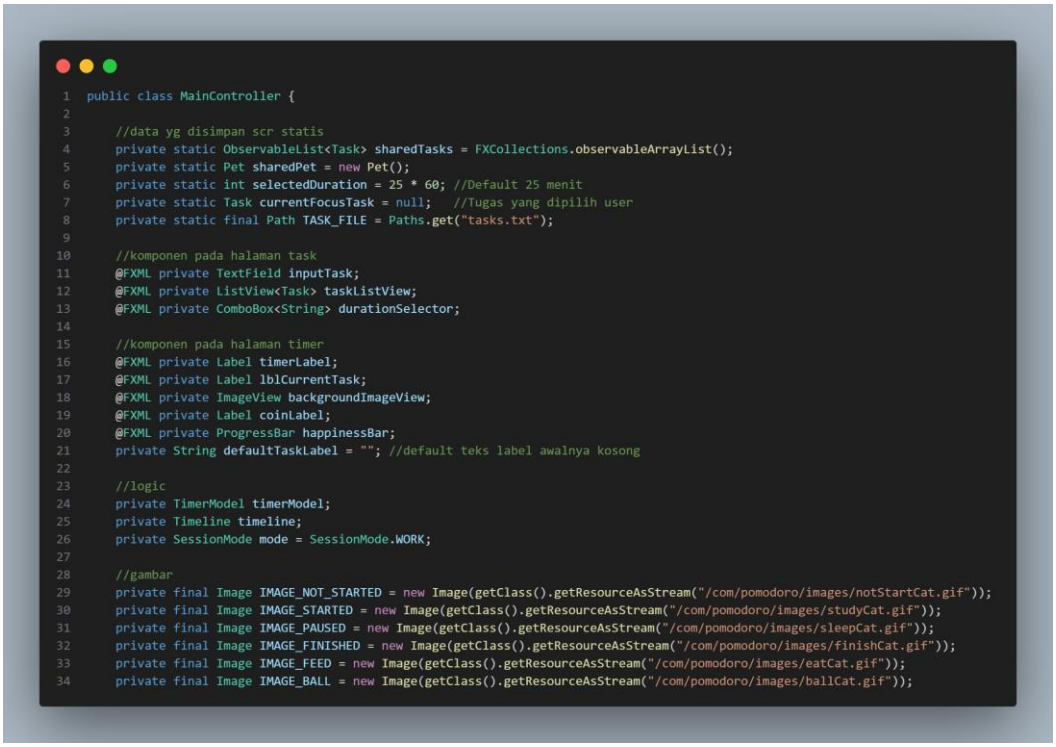
- **fx-background-radius: 30;**

  Properti ini menetapkan radius sudut sebesar 30 piksel, menghasilkan bentuk tombol dengan sudut membulat atau menyerupai kapsul.

- **fx-cursor: hand;**

  Atribut ini mengubah tampilan kursor menjadi ikon tangan saat pengguna mengarahkan pointer ke tombol, menandakan bahwa elemen tersebut bersifat interaktif.

3.2.8 MainController.java



```java
public class MainController {

    //data yg disimpan scr statis
    private static ObservableList<Task> sharedTasks = FXCollections.observableArrayList();
    private static Pet sharedPet = new Pet();
    private static int selectedDuration = 25 * 60; //Default 25 menit
    private static Task currentFocusTask = null;   //Tugas yang dipilih user
    private static final Path TASK_FILE = Paths.get("tasks.txt");

    //komponen pada halaman task
    @FXML private TextField inputTask;
    @FXML private ListView<Task> taskListView;
    @FXML private ComboBox<String> durationSelector;

    //komponen pada halaman timer
    @FXML private Label timerLabel;
    @FXML private Label lblCurrentTask;
    @FXML private ImageView backgroundImageView;
    @FXML private Label coinLabel;
    @FXML private ProgressBar happinessBar;
    private String defaultTaskLabel = ""; //default teks label awalnya kosong

    //logic
    private TimerModel timerModel;
    private Timeline timeline;
    private SessionMode mode = SessionMode.WORK;

    //gambar
    private final Image IMAGE_NOT_STARTED = new Image(getClass().getResourceAsStream("/com/pomodoro/images/notStartCat.gif"));
    private final Image IMAGE_STARTED = new Image(getClass().getResourceAsStream("/com/pomodoro/images/studyCat.gif"));
    private final Image IMAGE_PAUSED = new Image(getClass().getResourceAsStream("/com/pomodoro/images/sleepCat.gif"));
    private final Image IMAGE_FINISHED = new Image(getClass().getResourceAsStream("/com/pomodoro/images/finishCat.gif"));
    private final Image IMAGE_FEED = new Image(getClass().getResourceAsStream("/com/pomodoro/images/eatCat.gif"));
    private final Image IMAGE_BALL = new Image(getClass().getResourceAsStream("/com/pomodoro/images/ballCat.gif"));
```

Kode di atas digunakan sebagai *controller* utama aplikasi PomoCat yang menghubungkan antarmuka FXML dengan logika aplikasi. *Controller* mengatur tiga elemen utama yaitu daftar tugas, timer Pomodoro, dan *pet virtual*. Data tugas, *pet*, dan durasi *timer* disimpann secara statis agar tetap konsisten saat berpindah halaman. Saat aplikasi dijalankan, *controller* menginisialisasi tampilan, memuat *task* dari *file*, mengatur *binding* UI seperti *coins*, *happiness bar*, serta pilihan durasi *timer*.

Selain itu, controller menangani berbagai interaksi user, seperti menambah dan menghapus tugas, memulai, menjeda, atau mereset timer, berpindah halaman, serta menjalankan sesi kerja dan istirahat secara otomatis. Controller juga mengatur reward untuk pet, menampilkan animasi GIF, serta menyimpan perubahan tugas ke file.

3.2.9 style.css

File CSS digunakan untuk mengatur tampilan visual seluruh antarmuka aplikasi PomoCat. *Stylesheet* ini menentukan gaya global seperti warna tipografi, bayangan, betuk komponen, serta efek interaksi. Aturan-aturan di dalamnya diterapkan pada berbagai elemen seperti tombol, kartu, *input*, HUD, hingga teks besar pada halaman *timer*.

a. .root

Kode di atas digunakan untuk mengatur font default aplikasi menjadi Inter, Segoe UI, atau *sans-serif*. Semua label dan tombol diberikan efek *drop shadow* pada berbagai komponen sehingga tampak lebih menonjol dan tidak terlihat *flat*.

b.card

Elemen menggunakan warna putih dengan tingkat transparansi sekitar 85%. Pengaturan ini menghasilkan efek semi-transparan yang menyerupai frosted glass pada latar belakang. Properti radius sebesar 15 piksel diterapkan pada sudut-sudut elemen, menghasilkan bentuk dengan sudut membulat dan tampilan yang lebih halus. Bayangan lembut diterapkan pada bagian bawah elemen untuk memberikan efek elevasi, sehingga komponen tampak terangkat dari latar belakang.

c. Tombol (.button variants)

Primary (Biru) digunakan untuk aksi utama dalam alur interaksi, misalnya tombol “Go to Timer”. Success (Hijau) digunakan erepresentasikan konfirmasi, seperti “Add Task” dan “Start”. Warning (Oranye) digunakan untuk penundaan atau intervensi sementara, misalnya “Pause”. Danger (Merah) Digunakan untik mereset status, seperti “Delete” atau “Reset”.

d. Efek Hover

– **Efek Hover pada Tombol**

Ketika kursor pengguna berada di atas elemen tombol, komponen tersebut menampilkan efek cahaya berwarna cyan yang halus. Efek ini berfungsi sebagai umpan balik visual langsung untuk menegaskan bahwa tombol berada dalam kondisi aktif dan dapat diinteraksikan, sehingga meningkatkan kejelasan serta responsivitas antarmuka.

– **HUD & Timer**

Komponen hud-box, yang memuat indikator koin dan hati, menggunakan latar berwarna putih dengan tingkat transparansi sekitar 25%. Penggunaan transparansi ini menjaga agar informasi tetap mudah terbaca tanpa sepenuhnya menutupi citra latar belakang. Pendekatan ini mempertahankan keseimbangan antara estetika visual dan kejelasan informasi, terutama pada elemen yang sifatnya kritis dalam pengamatan waktu nyata.

– **Efek Bayangan pada Tampilan Waktu**

Elemen penunjuk waktu diberikan bayangan hitam dengan tingkat opasitas tinggi yang diletakkan langsung di belakang teks berwarna putih. Tujuannya adalah mempertahankan keterbacaan angka timer secara konsisten, terutama ketika citra latar memiliki tingkat kecerahan tinggi atau dominasi warna terang.

- **.big-title**

  Komponen judul menggunakan efek visual bergaya neon. Teks ditampilkan dalam warna biru muda dengan garis tepi (stroke) yang mempertegas bentuk huruf. Selain itu, lapisan cahaya putih yang mengelilingi teks berfungsi untuk meningkatkan kontras dan memberikan penekanan visual tanpa mengurangi keterbacaan.

e. Gaya Tombol Transparan Berbingkai

Tombol ini tidak menggunakan warna latar, melainkan mempertahankan tampilan transparan dengan tingkat opasitas rendah. Sebagai penegas bentuk, tombol diberi garis tepi berwarna putih. Pendekatan ini dipilih agar komponen tetap terlihat fungsional tanpa menarik perhatian secara berlebihan, sekaligus menjaga agar elemen visual latar—terutama gambar kucing—tetap menjadi fokus utama antarmuka.

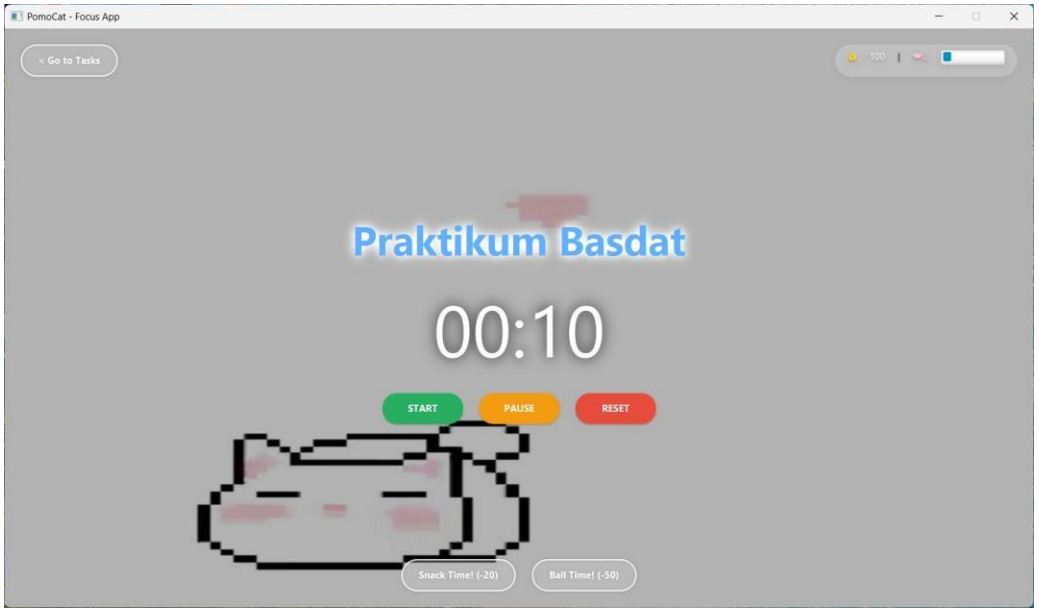
f. Gaya Tombol Transparan Berbingkai

Tombol ini tidak menggunakan warna latar, melainkan mempertahankan tampilan transparan dengan tingkat opasitas rendah. Sebagai penegas bentuk, tombol diberi garis tepi berwarna putih. Pendekatan ini dipilih agar komponen tetap terlihat fungsional tanpa menarik perhatian secara berlebihan, sekaligus menjaga agar elemen visual latar—terutama gambar kucing—tetap menjadi fokus utama antarmuka.

BAB IV
FITUR UTAMA

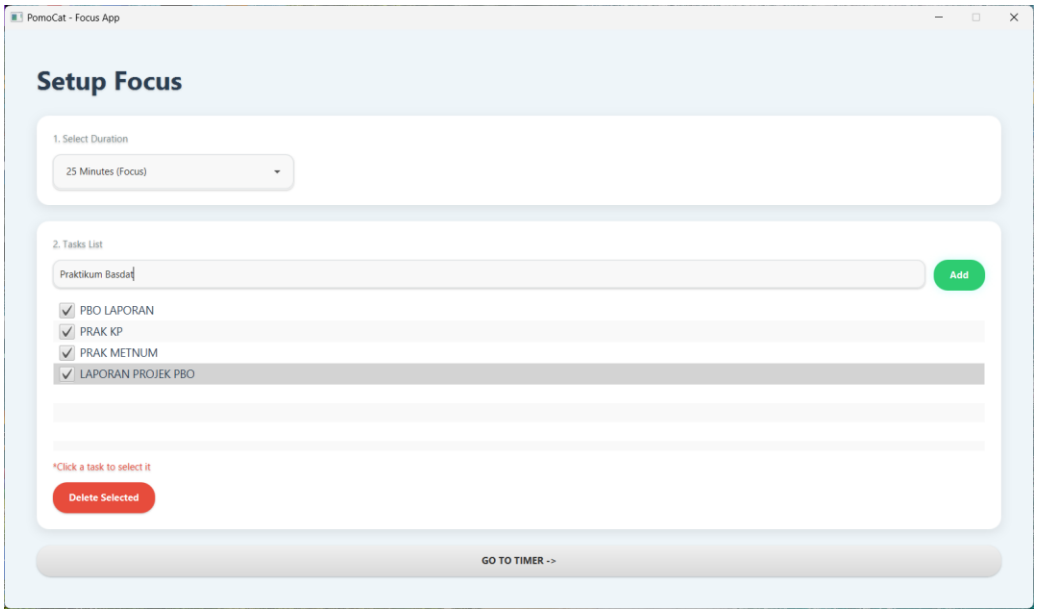
4.1 Pomodoro Timer

Aplikasi menyediakan fitur timer yang dapat berjalan secara mandiri menggunakan multithreading. Timer ini digunakan untuk mensimulasikan waktu pengerjaan tugas atau aktivitas tertentu, mirip dengan metode Pomodoro (durasi kerja + istirahat). Fitur ini didukung oleh TimerModel dan ditambah oleh thread terpisah.



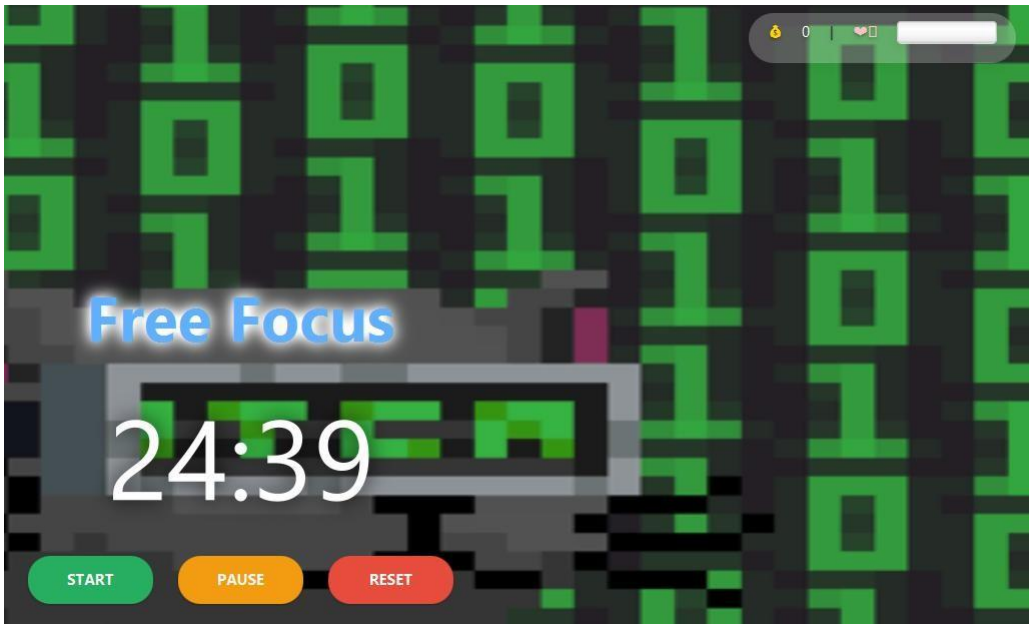
4.2 Task Management

Aplikasi memungkinkan pengguna membuat dan menjalankan tugas tertentu melalui antar muka Task. Setiap tugas memiliki durasi yang dapat dijalankan menggunakan timer, dan efeknya mempengaruhi kondisi pet. Fitur ini didukung oleh task interface dan implementasi di MainController.



4.3 Virtual Pet System

Pengguna dapat mengelola hewan peliharaan virtual yang memiliki atribut tertentu (misalnya energi, kebersihan, mood, dsb tergantung atribut di kelas Pet). Aktivitas yang dijalankan melalui task akan mengubah status pet. Fitur ini didukung oleh pet abstract class.



4.4 Real Time Status Update

Ketika timer dan task berjalan, status pet atau status tugas diperbarui secara real-time tanpa menghentikan proses lainnya. Ini membuat aplikasi terasa hidup dan responsif. Fitur ini didukung oleh multithreading di timer dan controller.

4.5 Activity Coordination via Controller

MainController bertindak sebagai pusat kendali yang mengatur hubungan antara Timer, Pet, dan Task. Fitur ini membuat aplikasi mampu menjalankan banyak aktivitas tanpa bentrok. Fitur ini didukung oleh MainController.

BAB V
EVALUASI DAN PENUTUP

5.1 Tantangan dalam Pengembangan (Challege)

Selama proses pengembangan, beberapa tantangan teknis muncul, terutama karena aplikasi memadukan beberapa konsep OOP dan multithreading:

- Sinkronisasi antara logika timer dengan aktivitas. Timer berjalan pada thread terpisah, sehingga kontrol alur dan komunikasi antara TimerModel dengan MainController memerlukan perhatian khusus agar tidak terjadi kondisi balapan (race condition) atau status tidak konsisten.
- Perancangan hierarki OOP yang tepat. Menentukan atribut dan perilaku yang akan ditempatkan dalam abstract class Pet membutuhkan analisis, karena keputusan awal mempengaruhi fleksibilitas pengembangan subclass di masa depan.
- Implementasi interface task. Tantangan muncul dalam memastikan setiap implementasi Task benar-benar memiliki efek yang relevan terhadap objek Pet, serta menjaga agar sistem tetap extensible tanpa melanggar prinsip Open-Closed.
- Pengelolaan state yang berubah seiring waktu. Aktivitas dan timer dapat mengubah kondisi Pet secara bersamaan. Mengatur urutan perubahan state dan validasi nilai menjadi krusial agar tidak menimbulkan perilaku aplikasi yang salah.
- Controller sebagai pusat koordinasi. MainController menanggung banyak tanggung jawab, sehingga perancangan metode dan alur kontrol harus sangat hati-hati agar tidak menimbulkan kompleksitas yang berlebihan di satu kelas.

5.2 Peluang dalam Pengembangan (Opportunities)

Meskipun aplikasi masih berada pada tahap dasar, arsitektur yang sudah dibangun memberikan banyak kesempatan untuk dikembangkan menjadi sistem yang lebih besar dan kompleks. Beberapa peluang pengembangan antara lain:

- Pengembangan menjadi aplikasi produktivitas nyata. Timer Pomodoro dapat dijadikan fitur utama dengan UI modern, statistik produktivitas, dan integrasi fitur to-do list.
- Gamifikasi perawatan pet. Dengan struktur OOP yang sudah rapi, aplikasi dapat dikembangkan menjadi game ringan berbasis peningkatan level, mini-games, atau sistem kebutuhan seperti energi, mood, dan kesehatan.
- Integrasi dengan platform mobile atau web. Logika backend yang sudah modular memungkinkan migrasi ke Android, Web, atau desktop GUI.
- Penerapan pola desain (Design Pattern). Beberapa pattern seperti Observer, Strategy, atau Factory Method bisa ditambahkan untuk meningkatkan skalabilitas dan kualitas arsitektur.
- Kolaborasi dengan backend / database. Sistem pelacakan progres dan status Pet dapat disimpan secara permanen, sehingga aplikasi berkembang dari sekadar demo teknis menjadi sistem yang sesungguhnya.

5.3 Kesimpulan

Proyek pengembangan aplikasi PomoPet berhasil menerapkan konsep-konsep utama Pemrograman Berorientasi Objek (PBO), yaitu abstract class, interface, dan multithreading, dalam sebuah simulasi perawatan hewan peliharaan yang terintegrasi dengan mekanisme timer.

Struktur sistem terdiri dari beberapa komponen inti, yaitu kelas abstrak Pet sebagai dasar objek peliharaan, interface Task sebagai standar perilaku aktivitas, TimerModel sebagai pengatur waktu berbasis thread, serta MainController sebagai pengendali utama alur aplikasi.

Implementasi ini menunjukkan bahwa pemanfaatan prinsip OOP dapat menciptakan arsitektur program yang modular, fleksibel, dan mudah dikembangkan. Proyek ini juga memperlihatkan bagaimana multithreading dapat digunakan untuk memisahkan proses timer dari interaksi aplikasi lain sehingga program tetap responsif. Dengan demikian, aplikasi PomoPet bukan hanya memenuhi kebutuhan fungsional yang ditugaskan, tetapi juga memberikan pengalaman simulasi yang terstruktur dan dinamis.

5.4 Saran

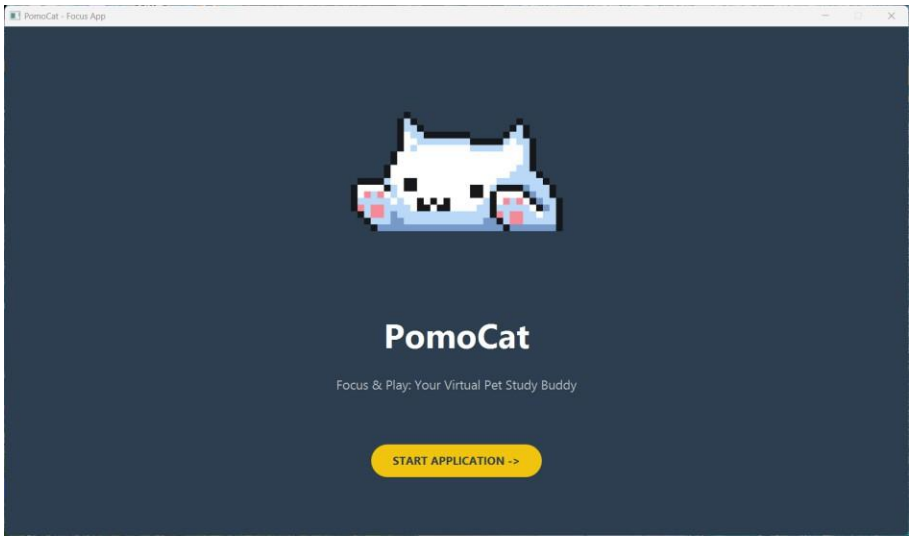
Beberapa pengembangan lanjutan dapat dilakukan agar aplikasi menjadi lebih komprehensif dan siap digunakan oleh pengguna akhir. Saran yang dapat diberikan antara lain:

- Menambahkan antarmuka pengguna (GUI) agar interaksi lebih intuitif dan menarik.
- Memperluas jenis hewan peliharaan, misalnya menambahkan subclass Pet seperti Cat, Dog, atau Bird.
- Membuat sistem penyimpanan data, baik berbasis file maupun database, agar status peliharaan dapat disimpan.
- Mengembangkan sistem reward, misalnya koin atau level-up, untuk meningkatkan motivasi pengguna.
- Menambah pengaturan timer, seperti pengaturan interval kerja-istirahat (custom Pomodoro).

BAB VI
LAMPIRAN

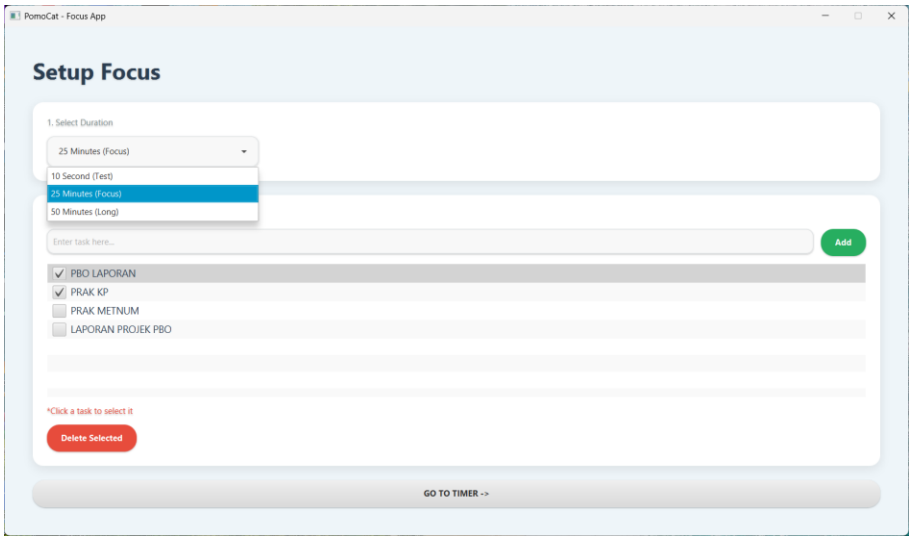
6.1 UI awal

Tampilan awal saat program dibuka

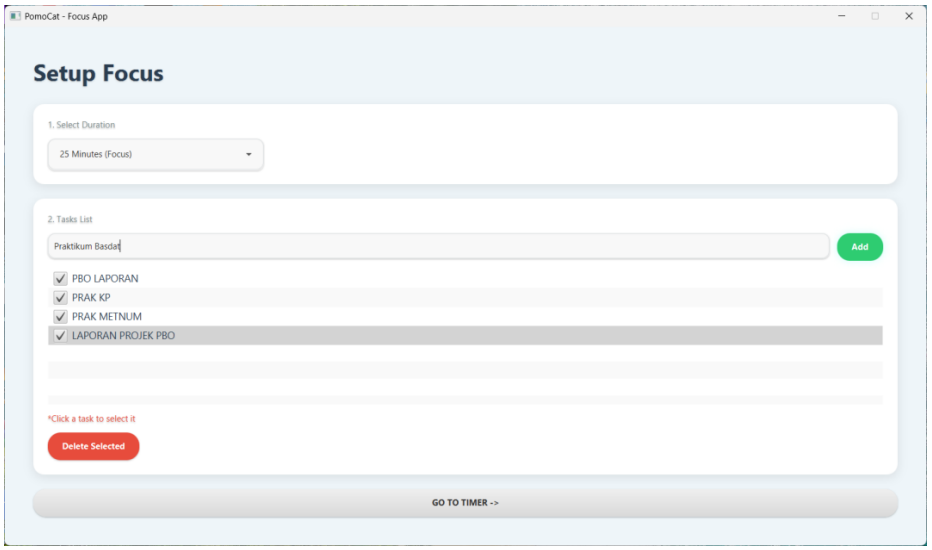


6.2 Setup Focus and Task Manager

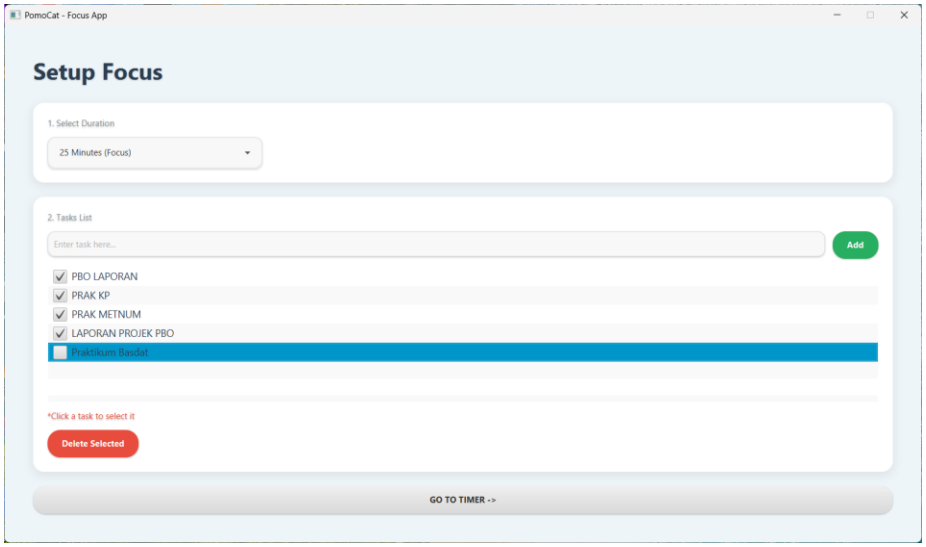
Pengguna dapat memilih durasi pomodoro timer



Setiap task yang sudah selesai dapat diberikan tanda berupa checklist pada kotak disamping deskripsi task

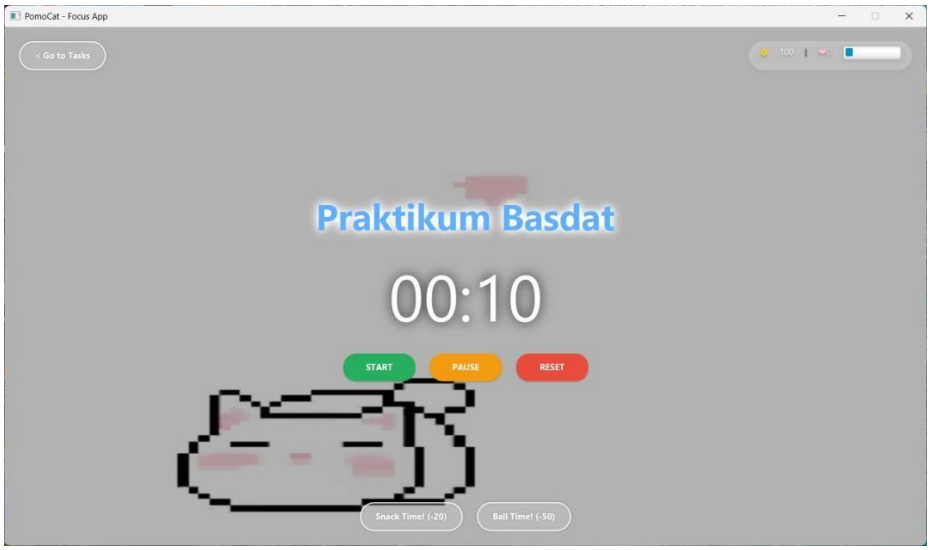


Menambahkan daftar task terbaru yang ingin dikerjakan

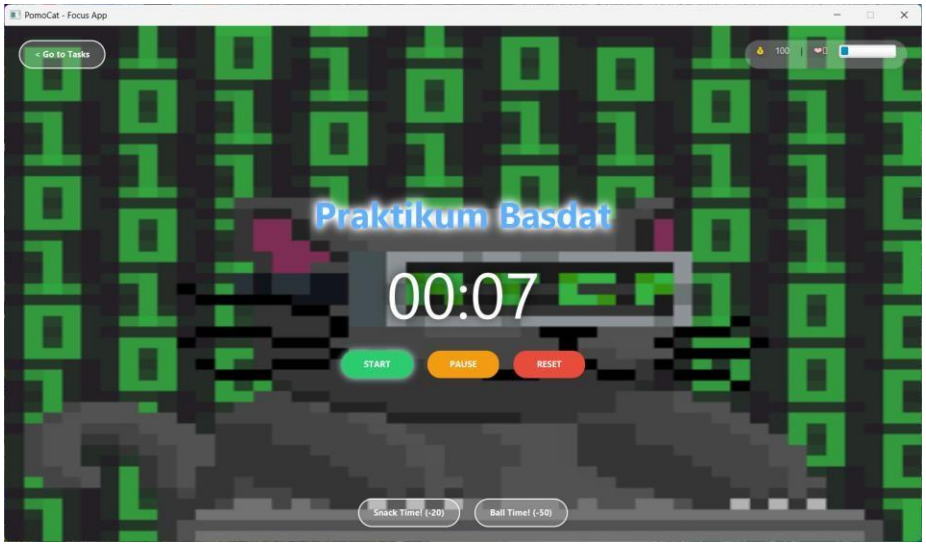


6.3 Pomodoro Timer

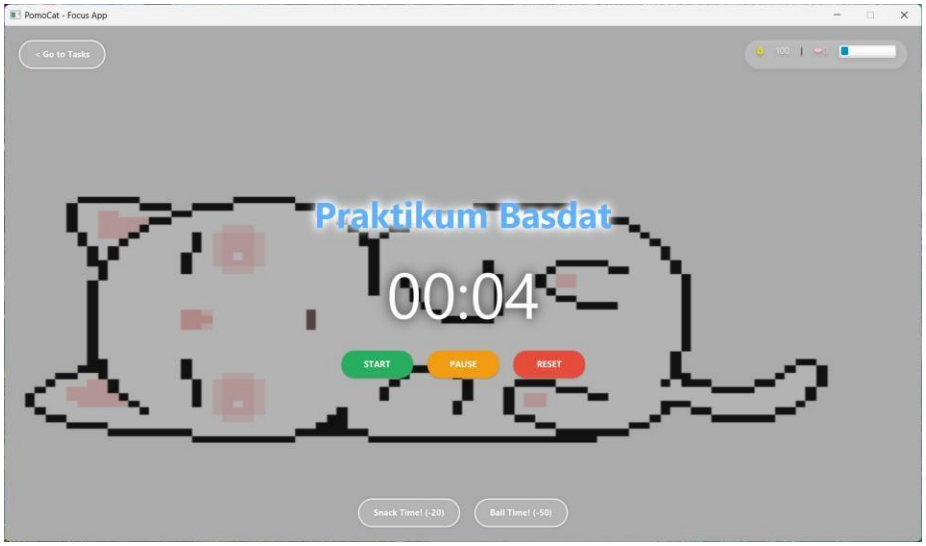
Tampilan awal timer sebelum dimulai



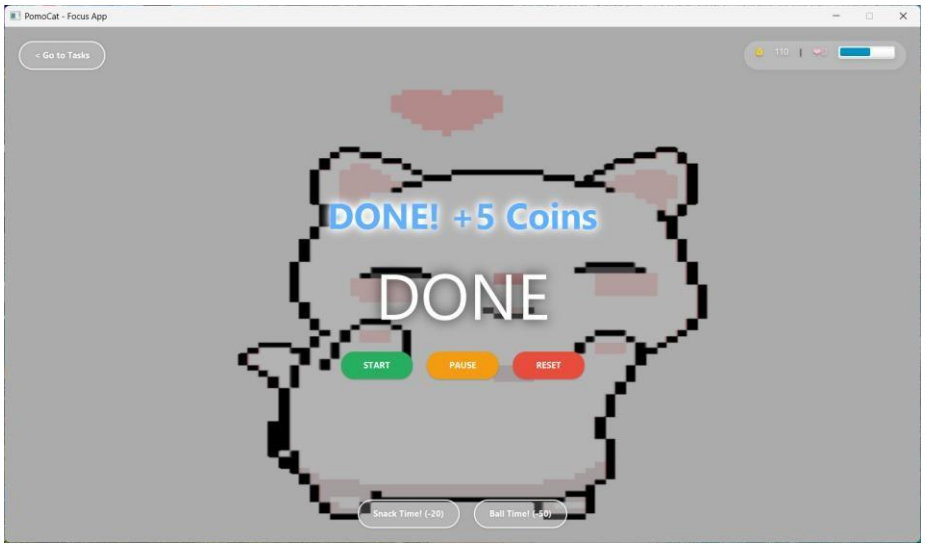
Tampilan setelah timer dimulai



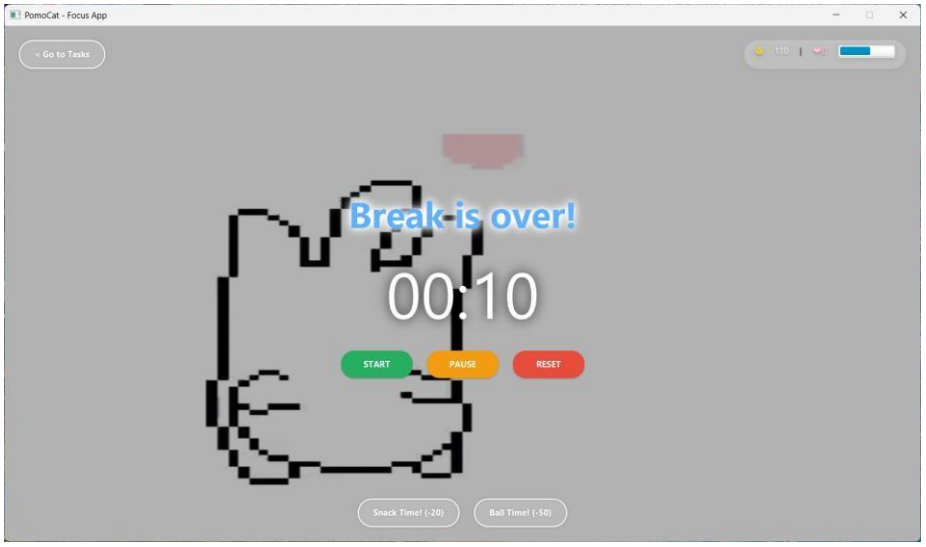
Tampilan saat timer di pause



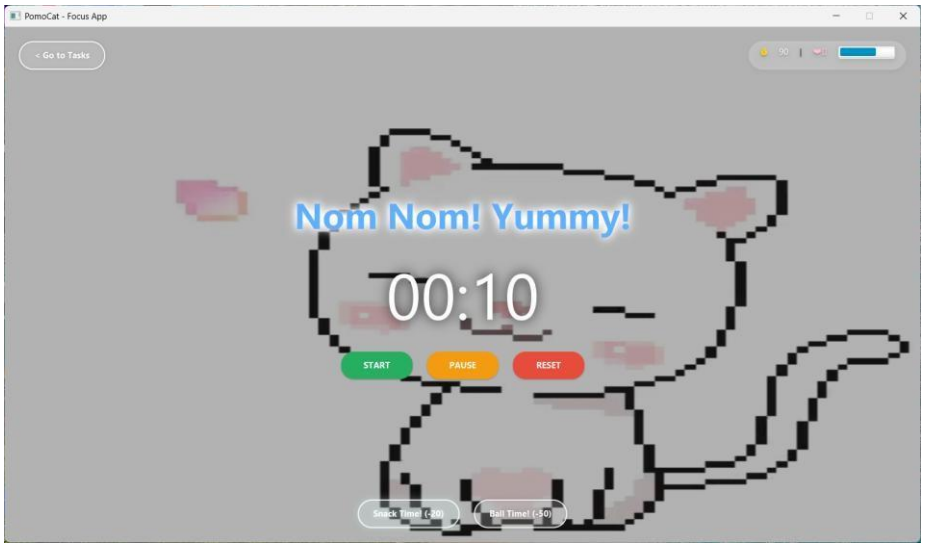
Tampilan saat waktu timer selesai, pengguna mendapatkan tambahan 5 poin



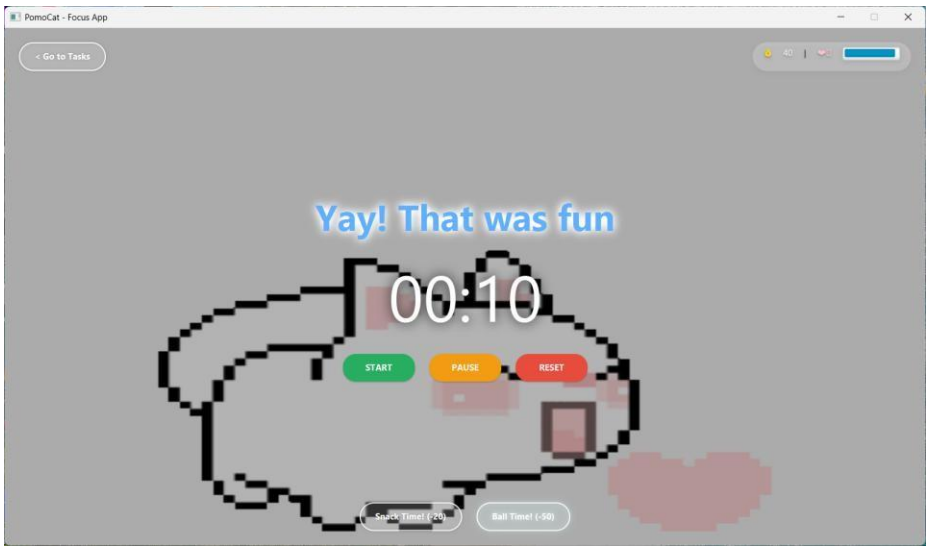
Akan ada waktu istirahat sebelum akan kembali melanjutkan timernya



Tampilan untuk snack time yang membutuhkan 20 coins



Tampilan untuk ball time yang membutuhkan 50 coins


PomoCat - Focus App
< Go to Tasks
Yay! That was fun
00:10
START
PAUSE
RESET
Snack Time! (+20)
Ball Time! (-50)