MY TAMGOTCHI APP

Cartoon a cartoon dog with its tongue out

Description automatically generated

Cartoon a dog in a tub

Description automatically generated

IMAD ASSIGNMENT 2

MOHAMED LACHPORIA

Contents

[MY TAMAGOTCHI APP 2](#_Toc165895543)

[Pictures that I used within my Tamagotchi app. 2](#_Toc165895544)

[Activity\_main.xml 3](#_Toc165895545)

[Figure1 3](#_Toc165895546)

[Figure 2 4](#_Toc165895547)

[MainActivity.kt 5](#_Toc165895548)

[Figure 3 5](#_Toc165895549)

[Activity\_main2.xml 6](#_Toc165895550)

[Figure 4 6](#_Toc165895551)

[Figure5 7](#_Toc165895552)

[MainActivity2,kt 8](#_Toc165895553)

[Figure 6 8](#_Toc165895554)

[Figure7 9](#_Toc165895555)

[User Interfaces 10](#_Toc165895556)

[References 11](#_Toc165895557)

# MY TAMAGOTCHI APP

I have spent considerable time and effort creating a fully functional and interactive game application for Android devices, utilizing the Kotlin programming language. The app is designed to serve as a Tamagotchi-style game, where users can take care of and nurture their virtual pets by feeding them, playing with them, and keeping them healthy and happy. The game includes various levels, characters, and challenges that users can unlock and explore as they progress through the game. The app has been thoroughly tested to ensure that it runs smoothly on a wide range of Android devices and provides an engaging and enjoyable gaming experience for users of all ages.

## Pictures that I used within my Tamagotchi app.

Cartoon a dog in a tub

Description automatically generatedA cartoon of a dog eating food

Description automatically generated

Cartoon a cartoon dog holding a ball

Description automatically generatedCartoon a cartoon dog with its tongue out

Description automatically generatedA cartoon dog holding a bowl

Description automatically generated

# Activity\_main.xml

A screenshot of a computer program

Description automatically generated

## Figure1

In the provided XML code snippet, you have defined the layout for an Android activity using ConstraintLayout, which is a flexible layout manager provided by the Android SDK. Let's break down the elements and attributes used in this layout:ConstraintLayout: This is the root layout element, which allows you to create complex layouts by specifying constraints between child views. It provides flexibility in arranging UI elements relative to each other.TextView: This element represents a text view widget, which is used to display text on the screen. In this case, the text "Hello World!" is displayed. Constraints are applied to position the text view in the center of the screen both vertically and horizontally.layout\_width and layout\_height: Define the width and height of the text view.

app:layout\_constraintBottom\_toBottomOf, app:layout\_constraintEnd\_toEndOf, app:layout\_constraintStart\_toStartOf, app:layout\_constraintTop\_toTopOf: These attributes specify constraints to position the text view relative to the parent ConstraintLayout.app:layout\_constraintHorizontal\_bias, app:layout\_constraintVertical\_bias: These attributes define the horizontal and vertical bias of the text view within its constraints. ImageView: This element represents an image view widget, which is used to display images on the screen. In this case, an image is loaded from the drawable resources and displayed in the image view. Constraints are applied to position the image view on the screen.android:id: This attribute assigns an ID to the image view, which can be used to reference it programmatically.

android:layout\_width and android:layout\_height: Define the width and height of the image view.android:layout\_marginStart: Defines the margin from the start (left) edge of the screen.android:src: Specifies the image resource to be displayed in the image view.app:layout\_constraintStart\_toStartOf: This attribute specifies a constraint to position the start (left) edge of the image view relative to the start (left) edge of the parent ConstraintLayout.Overall, the provided XML layout defines a simple user interface with a text view and an image view. The text view displays the text "Hello World!" at the center of the screen, while the image view displays an image loaded from the drawable resources. Constraints are applied to position both views within the parent ConstraintLayout.

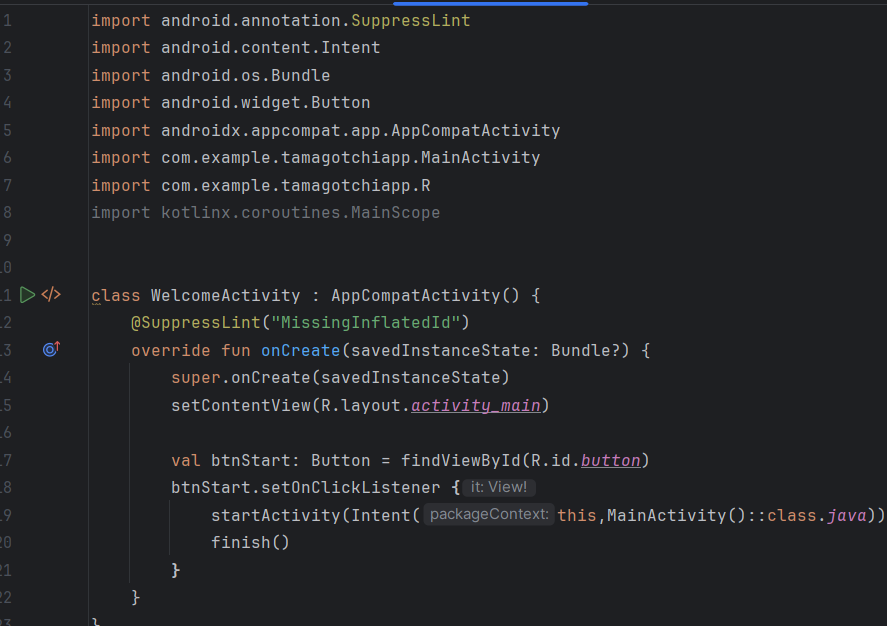


## Figure 2

Button Element: This XML element represents a button widget, an essential component of the user interface that enables users to trigger actions or initiate processes within the application. The button is labeled "Get Started," indicating its purpose to prompt users to commence an activity or operation. The following attributes are utilized to define the appearance and positioning of the button within the layout:android:id: This attribute assigns a unique identifier to the button, enabling programmatic access to it within the application's code.android:layout\_width and android:layout\_height: These attributes specify the width and height of the button, determining its size on the screen.android:text: This attribute specifies the text displayed on the button, conveying its intended action or purpose to the user.app:layout\_constraintEnd\_toEndOf and app:layout\_constraintStart\_toStartOf: These attributes establish constraints that anchor the button's horizontal position to the respective end and start edges of the parent ConstraintLayout, ensuring it spans the width of the screen.tools:layout\_editor\_absoluteY: This attribute provides a preview-only absolute Y-coordinate for the button's vertical positioning within the layout during design-time visualization in Android Studio's layout editor.tools:ignore: This attribute instructs the layout editor to disregard missing constraints during design-time rendering, allowing for a more flexible and iterative design process.

# MainActivity.kt

## Figure 3



WelcomeActivity is a pivotal component of the Tamagotchi Android application, serving as the initial interface for users. Structured as an AppCompatActivity class, it ensures compatibility across diverse Android devices. Annotated with @SuppressLint("MissingInflatedId"), it suppresses lint warnings related to missing IDs in layout files, maintaining code cleanliness. In the onCreate() method, setContentView(R.layout.activity\_main) inflates the UI layout defined in activity\_main.xml, likely containing various welcoming UI elements. The initialization of the start button (btnStart) and its click listener via findViewById() and setOnClickListener() respectively, enable programmatic interaction. Upon button click, startActivity() is triggered with an explicit intent targeting MainActivity, facilitating the transition to the main application functionality. Finally, finish() ensures the closure of WelcomeActivity, optimizing memory usage and maintaining activity stack cleanliness. WelcomeActivity embodies effective UI design principles, offering users a seamless transition into the app experience. Its careful implementation exemplifies the importance of thoughtful design and functionality in Android app development, enabling developers to enhance user satisfaction and drive application success.

# Activity\_main2.xml

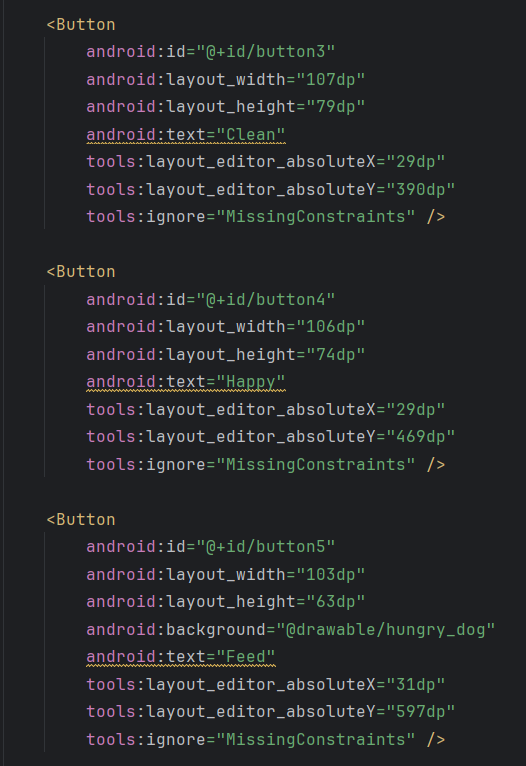
## Figure 4

A screen shot of a computer program

Description automatically generated

In the XML code provided, a layout for the MainActivity of an Android application is defined using ConstraintLayout, a powerful layout manager in Android development. This layout specifies the arrangement and properties of UI elements, facilitating the visual presentation of the app's main screen. The root element, ConstraintLayout, serves as the container for UI elements and allows for flexible positioning and sizing through constraints. Within this layout, two UI elements are defined: an ImageView and a Button. The ImageView (imageView2) is configured to display an image, presumably a visual representation relevant to the application. Its width and height are set to specific dimensions (409dp and 256dp, respectively), and constraints are applied to ensure it spans the entire width of the parent layout while maintaining its aspect ratio. This element likely serves a decorative or illustrative purpose, enhancing the visual appeal of the MainActivity.

The Button (button2) is defined with text "Hunger" and specified dimensions (102dp width, 74dp height). However, this Button lacks constraints, as indicated by the tools:ignore="MissingConstraints" attribute. Constraints define the position of UI elements relative to each other or the parent layout, ensuring consistent and responsive layout behavior across different screen sizes and orientations. The absence of constraints may result in layout issues, such as elements overlapping or being positioned incorrectly on certain devices.In summary, the provided XML layout code establishes the visual structure of the MainActivity in an Android application. It includes an ImageView for displaying an image and a Button for user interaction, though proper constraint definitions are necessary to ensure optimal layout behavior across various device configurations.

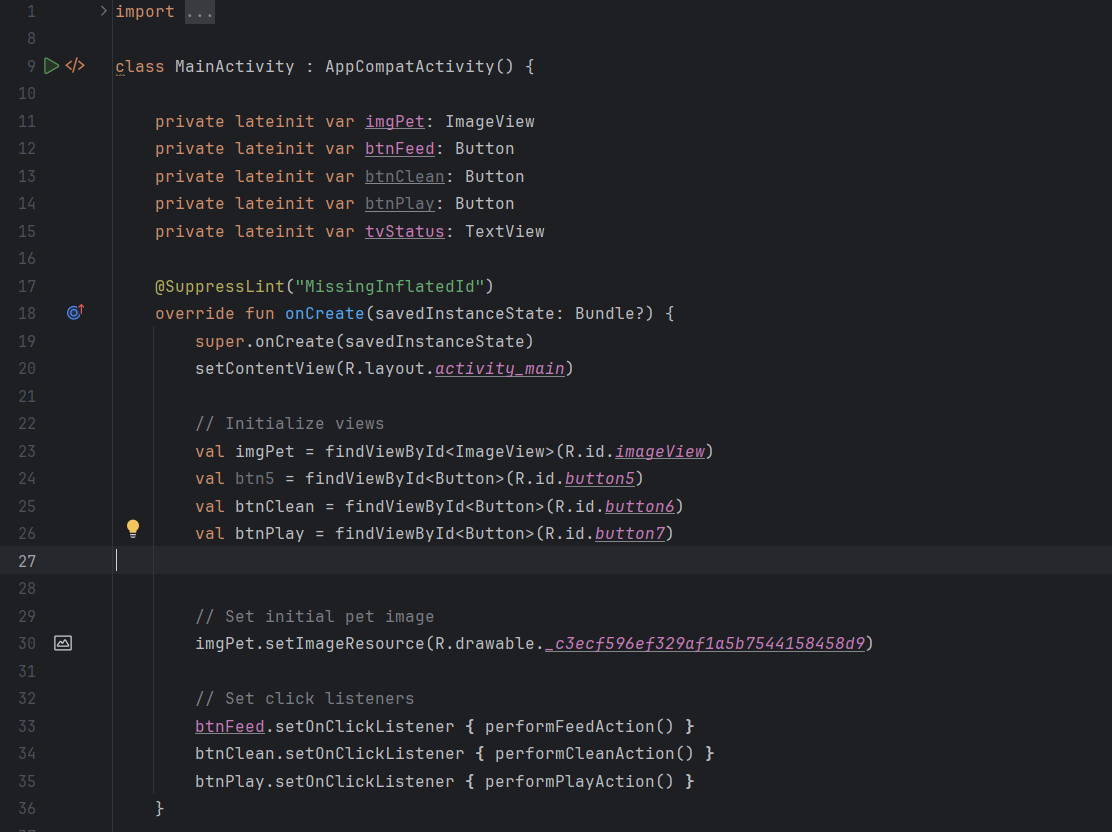


## Figure5

In the provided XML code snippet, three Button elements are defined for the layout of an Android application screen. Each Button is assigned a unique ID and configured with specific dimensions and text. The first Button (button3) is labeled "Clean" and has a width of 107dp and height of 79dp. It is positioned at coordinates (29dp, 390dp) within the layout. However, like the subsequent Buttons, it lacks proper constraint definitions, which are crucial for ensuring consistent layout behavior across different devices and screen sizes. The second Button (button4) is labeled "Happy" and has dimensions 106dp width and 74dp height. It is positioned at coordinates (29dp, 469dp) within the layout. The third Button (button5) is labeled "Feed" and has a custom background defined by the @drawable/hungry\_dog resource. It has dimensions 103dp width and 63dp height and is positioned at coordinates (31dp, 597dp) within the layout. While these Buttons define the visual elements and interaction points of the application screen, the absence of constraint definitions may lead to layout issues on certain devices. It's essential to incorporate proper constraints to ensure the responsiveness and adaptability of the layout across diverse Android devices.

# MainActivity2,kt

## Figure 6



In the provided code snippet, an implementation of the MainActivity class in an Android application is outlined. MainActivity serves as the central activity where users interact with the Tamagotchi application.

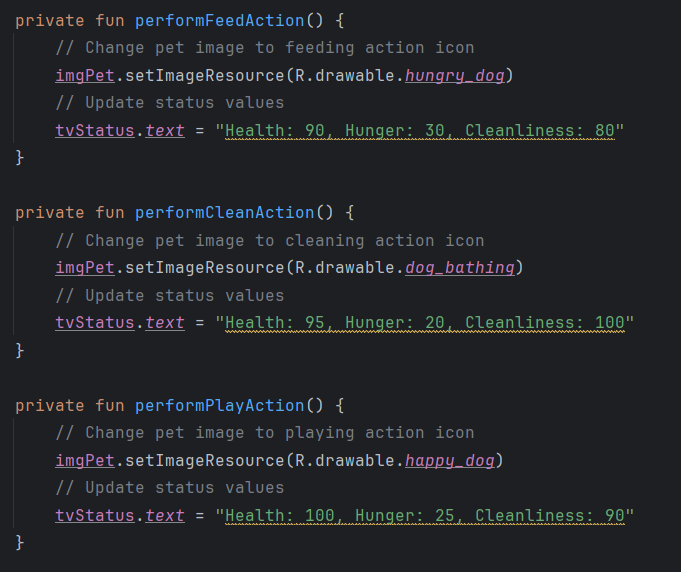
The class extends AppCompatActivity, indicating its compatibility with the Android support library and ensuring consistent behavior across different Android versions and devices.

Several UI elements are initialized within MainActivity, including an ImageView (imgPet) for displaying the pet's image, and three Button widgets (btnFeed, btnClean, btnPlay) for user interaction. Additionally, a TextView (tvStatus) may be used for displaying the pet's status, though it's not initialized in the provided code.

The @SuppressLint("MissingInflatedId") annotation suppresses lint warnings related to potentially missing IDs in layout files, enhancing code readability while acknowledging any potential issues.

Within the onCreate() method, the layout defined in activity\_main.xml is set as the content view using setContentView(R.layout.activity\_main), ensuring that the UI elements are displayed correctly.

Views are initialized using findViewById(), associating them with their respective XML elements. However, there's a potential issue with variable shadowing in the initialization of views, where local variables are created with the same names as class properties.Initial configurations are applied to the pet's image using imgPet.setImageResource(), setting its initial image to a drawable resource. Click listeners are set for the three buttons, specifying actions to be performed when each button is clicked. However, there seems to be a discrepancy where btnFeed is used without being initialized in the provided code, potentially causing a runtime error. In summary, MainActivity is a crucial component of the Tamagotchi application, responsible for managing user interaction and updating the UI based on user input. Proper initialization and configuration of UI elements are essential for ensuring a smooth user experience and robust functionality.



## Figure7

In the provided code snippet, three private functions are defined within the MainActivity class of an Android application: performFeedAction(), performCleanAction(), and performPlayAction(). Each function encapsulates a specific action related to interacting with a virtual pet represented by an ImageView (imgPet) and updates its status displayed in a TextView (tvStatus).

In performFeedAction(), the pet's image is changed to a feeding action icon, and the status values reflecting the pet's health, hunger, and cleanliness are updated accordingly.

performCleanAction() updates the pet's image to a cleaning action icon and adjusts the status values to reflect the pet's improved cleanliness and health, with a decrease in hunger.

performPlayAction() changes the pet's image to a playing action icon and updates the status values to indicate the pet's optimal health, slightly decreased hunger, and maintained cleanliness.

These functions enable users to interact with the virtual pet by simulating feeding, cleaning, and playing actions, dynamically updating the UI to reflect the pet's state. By encapsulating these actions within separate functions, the code achieves modularity and readability, enhancing maintainability and ease of debugging.

# User Interfaces

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

# References

Image 1 (figure1) [cute-fun-dog-cartoon-vector-6833014.jpg (780×1080) (vectorstock.com)](https://cdn.vectorstock.com/i/1000v/30/14/cute-fun-dog-cartoon-vector-6833014.jpg)

Image 2 (hungry dog) [R.d52b5afc6ede1b87250bdbd50fedd873 (1291×1690) (bing.com)](https://th.bing.com/th/id/R.d52b5afc6ede1b87250bdbd50fedd873?rik=SZV0INaOLE3nKw&riu=http%3a%2f%2fthumbs.dreamstime.com%2fz%2fhungry-dog-hand-drawn-cartoon-bowl-awaiting-dinner-41453522.jpg&ehk=ch2qlL4N%2fw9zWpECnRM%2fQadY60feeLiqjL0OQRzP5ek%3d&risl=&pid=ImgRaw&r=0)

Image 3(playful dog) [OIP.CFKC0hBuOO71azkCr70iHQHaH- (474×510) (bing.com)](https://th.bing.com/th/id/OIP.CFKC0hBuOO71azkCr70iHQHaH-?rs=1&pid=ImgDetMain)

Image 4 (bathing ) [cartoon-dog-having-a-bath-vector-27004515.jpg (1000×919) (vectorstock.com)](https://cdn1.vectorstock.com/i/1000x1000/45/15/cartoon-dog-having-a-bath-vector-27004515.jpg)