# Report

The Tamagotchi app simulates the experience of owning a pet. The app gives users the same responsibilities that a real per own would have, such as, feeding, cleaning, playing with it. This is all done to ensure that the pet stays happy and healthy.

The objective of this assignment is to create our own Tamagotchi app using Android Studios. Featuring a welcome page that leads users into the game. Where the user can interact with their virtual pet by feeding, cleaning, and playing with the pet. This is all done to allow the user to focus on the pet’s cleanliness, hunger, and health status.

## Background

## User Interface

For this Tamagotchi app I used a minimalistic layout. Where there is only a title component, image component, and a button component.

The thyme that I have decided to go with incorporates colours that compliment the pet. Which in this case are darker colour, such as, …..

On start up of the app, the landing page will display a image of the users pet, a German Shepard. This image will be randomly selected from a selection of 5 images, which will then be displayed. These images serve to welcome the user to app and with functionality of randomising the images, this gives the app a lively appearance.

These are the following images that will be randomly selected:



## Welcome Screen Logic

The welcome screen features an image view component that will be assigned a random image to display.

// Create all the components

val petImage = findViewById<ImageView>(R.id.imageView) // imageView component

val btnGetStarted = findViewById<Button>(R.id.btnGetStarted) // button component

//Select a random image

var randomImage = (1..5).random()// generates a random value between 1 and 5, then assigns to random variable.

// The switch statement will assign a petImage resource based on the value generated above

when(randomImage){

1 -> petImage.setImageResource(R.drawable.home1)// assign the petImage to the home1 resource

2 -> petImage.setImageResource(R.drawable.home2)// assign the petImage to the home2 resource

3 -> petImage.setImageResource(R.drawable.home3)// assign the petImage to the home3 resource

4 -> petImage.setImageResource(R.drawable.home4)// assign the petImage to the home4 resource

5 -> petImage.setImageResource(R.drawable.home5)// assign the petImage to the home5 resource

}

btnGetStarted.setOnClickListener{

val intent = Intent(this, HomeScreen::class.java) // this will initialize a new activity

startActivity(intent) // start the activity that was initialized

}

## Second Screen Logic

The second screen has similar logic to the first screen. Where the difference can be having to also display the current counter and checking that the counter does not exceed 10 clicks.

// initialise all counter variables

var feedCounter = 0 // will count how many time the feed button will be clicked

var playCounter = 0// will count how many time the play button will be clicked

var cleanCounter = 0// will count how many time the clean button will be clicked

// Create Feed button functionality

btnFeed.setOnClickListener {

if(feedCounter < 10){// Will ensure that the counter stops incrementing when it reaches 10

feedCounter++ // increment the counter by 1

}

var random = (1..5).random() // generate a random number between 1 and 5

// The switch statement will assign a petImage resource based on the value generated above

when(random){

1 -> petImage.setImageResource(R.drawable.eating1)// assign the petImage to the eating1 resource

2 -> petImage.setImageResource(R.drawable.eating2)// assign the petImage to the eating2 resource

3 -> petImage.setImageResource(R.drawable.eating3)// assign the petImage to the eating3 resource

4 -> petImage.setImageResource(R.drawable.eating4)// assign the petImage to the eating4 resource

5 -> petImage.setImageResource(R.drawable.eating5)// assign the petImage to the eating5 resource

}

// Display all the counter variables

tvFeedResult.text = feedCounter.toString()

tvPlayResult.text = playCounter.toString()

tvCleanResult.text = cleanCounter.toString()

}

// create play button functionality

btnPlay.setOnClickListener {

playCounter++

var random = (1..5).random()// generate a random number between 1 and 5

// The switch statement will assign a petImage resource based on the value generated above

when(random){

1 -> petImage.setImageResource(R.drawable.playing1)// assign the petImage to the playing1 resource

2 -> petImage.setImageResource(R.drawable.playing2)// assign the petImage to the playing2 resource

3 -> petImage.setImageResource(R.drawable.playing3)// assign the petImage to the playing3 resource

4 -> petImage.setImageResource(R.drawable.playing4)// assign the petImage to the playing4 resource

5 -> petImage.setImageResource(R.drawable.playing5)// assign the petImage to the playing5 resource

}

// Display all the counter variables

tvFeedResult.text = feedCounter.toString()

tvPlayResult.text = playCounter.toString()

tvCleanResult.text = cleanCounter.toString()

}

btnClean.setOnClickListener {

cleanCounter++

var random = (1..5).random()// generate a random number between 1 and 5

// The switch statement will assign a petImage resource based on the value generated above

when(random){

1 -> petImage.setImageResource(R.drawable.clean1)// assign the petImage to the clean1 resource

2 -> petImage.setImageResource(R.drawable.clean2)// assign the petImage to the clean2 resource

3 -> petImage.setImageResource(R.drawable.clean3)// assign the petImage to the clean3 resource

4 -> petImage.setImageResource(R.drawable.clean4)// assign the petImage to the clean4 resource

5 -> petImage.setImageResource(R.drawable.clean5)// assign the petImage to the clean5 resource

}

// Display all the counter variables

tvFeedResult.text = feedCounter.toString()

tvPlayResult.text = playCounter.toString()

tvCleanResult.text = cleanCounter.toString()

}

// Button to go back to the starting page

btnBackToStart.setOnClickListener{

val intent = Intent(this, MainActivity::class.java)// this will initialize a new activity

startActivity(intent)// start the activity that was initialized

}

// a button to clear all the fields

btnClear.setOnClickListener {

val newValue = 0 // create a variable to reassign all the values

tvFeedResult.text = newValue.toString()

tvCleanResult.text = newValue.toString()

tvPlayResult.text = newValue.toString()

// reset all the counters

playCounter = newValue

feedCounter = newValue

cleanCounter = newValue

}

# Design Considerations

For this app I have gone for a minimal User Interface (UI). This design choice was chosen as it allows for easier legibility of content. This design made it possible to create consistency with my fonts, spacing, colour of buttons, and font weights such as bolding texts, and consistent font size such as 20px.

## Consistency

We must consider uniform design decisions within our app, such as, fonts, colours and button styles says Babich, N. (2016).

## Legible Content

Smartphones have relatively small screens, which means that one of the challenges of mobile design is to fit a lot of information on a small UI says Babich, N. (2016).

The following image shows what the final app layout will look like.

# GitHub Utilization

GitHub was an integral part of the development of this app. GitHub allowed me to document my code using a README file. This has helped me in explaining the purpose of my code. Allowing others to understand the project with ease.

# GitHub Actions

In this project we made use of GitHub Actions to build and test our projects before submitting, to make sure that everything was up to standard.

# Conclusion

This was a good indication of my programming abilities and highlighted all the areas that I need to work on to become a better developer. Such as, Logic design, naming conventions and commenting.

I am looking forward to utilizing more tools in future such as GitHub Actions. So that I can understand how my application will work within the real world.

I think one of my biggest setbacks during this project was my time management. In future I will be creating list of all the to-do items for any project. This will hopefully allow me to have structure within my work and allow me to stay on track with my tasks.

## References List

Open AI. (2024). ChatGPT 3.5

Babich, N. (2016). Mobile UX Design: Key Principles. [online] Medium. Available at: https://uxplanet.org/mobile-ux-design-key-principles-dee1a632f9e6 [Accessed 06 Apr. 2024].

www.uxdesigninstitute.com. (2023). Your ultimate guide to mobile app design. [online] Available at: https://www.uxdesigninstitute.com/blog/ultimate-guide-to-mobile-app-design/ [Accessed 06 Apr. 2024].