**Module Title:** Advanced Internet and Mobile Computing

**Module Code:** IS3S664

**Module Leader/Tutor:** Daniel Cunliffe

**Assessment Title:** Noughts and Crosses game Android App

Student Name: Dilukshika Sivanathan

Student No: 30073395



Table of Contents

[Introduction 3](#_Toc187429759)

[Base Code Overview 3](#_Toc187429760)

[Enhancements Implemented 4](#_Toc187429761)

[Implementation Details 5](#_Toc187429762)

[High-Level Functionality 5](#_Toc187429763)

[Detailed Enhancements 5](#_Toc187429764)

[1. Timer Integration 5](#_Toc187429765)

[2. Sound Effects 6](#_Toc187429766)

[3. Enhanced UI 7](#_Toc187429767)

[4. Result Dialog 8](#_Toc187429768)

[Testing Plan 10](#_Toc187429769)

[Objective 10](#_Toc187429770)

[Test Cases 10](#_Toc187429771)

[Testing Results 11](#_Toc187429772)

[Conclusion 12](#_Toc187429773)

[Reference 13](#_Toc187429774)

# Introduction

This report provides a detailed analysis of the development process for a Tic Tac Toe game application for Android. It outlines the base functionality of the code, enhancements made, the approach to implementation, and a test plan for validating the final application.

# Base Code Overview

The application is developed on Android Studio using **Kotlin**. The base code provided aims at implementing the basic functionality of a Tic Tac Toe game that includes:

Core Features:

* A 3x3 grid displaying the game board, which has been created using a TableLayout.
* The ability of two players ('X' and 'O') to take turns in marking cells.
* Methods to check for win conditions or draw scenarios.
* Score tracking of each player.

Technologies Used:

* ConstraintLayout for UI
* CountDownTimer for time-based turns
* MediaPlayer for game sounds

Base Functionality:

* Game starts with Player 'X'.
* A player clicks a cell in order to mark it.
* Check winning or draw condition just after every move.

# Enhancements Implemented

The project enhances the base application with the following features:

1. Visual Enhancements:

* Improved User Interface design through custom styles and colors, with a background image.
* Implement a timer that enforces a time limit for each player's turn.

1. Audio Features:

* Sound effects for winning, draw, and timer alerts.

1. Scoreboard:

* Display scores of Player X and Player O.
* Reset button integrated to reset scores.

1. Dynamic Timer:

* Count time for every turn with alerts for low times.
* Turn change based on time expiry.

1. Dialog Boxes:

* Dialog boxes popping up for game outcome-instances of win/draw, with the option to start a new game.

1. More Interaction:

* Buttons to restart the game or reset the entire score.

# Implementation Details

## High-Level Functionality

The Tic Tac Toe game represents the user-friendly way of playing against an opponent. The application handles scores, tight turn logic, and different feedbacks with sounds and views to make participation more engaging. Core logic:

* Turn-based gameplay logic supported by visuals to indicate whose move it currently is.
* Automating win condition and/or draw validation.
* Sound cues for when something important is happening.

## Detailed Enhancements

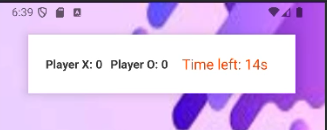
### Timer Integration

CountDownTimer class to countdown the time of a single player's turn. In every 5 seconds, it gives a warning sound, indicating time's up and automatically switches the turn.

**Code Implementation:**

|  |
| --- |
| private fun startTimer() {  timer?.cancel()  isLowTimeWarningPlayed = false  timer = object : CountDownTimer(TURN\_TIME\_MILLIS, 1000) {  override fun onTick(millisUntilFinished: Long) {  val secondsLeft = millisUntilFinished / 1000  timerTextView.text = "Time left: ${secondsLeft}s"  if (secondsLeft <= 5 && !isLowTimeWarningPlayed) {  soundManager.playTimerSound()  isLowTimeWarningPlayed = true  }  }  override fun onFinish() {  timerTextView.text = "Time's up!"  if (!checkGameStatus()) {  turn = if (turn == 'X') 'O' else 'X'  startTimer()  }  }  }.start()  } |

**Output :**



### Sound Effects

This SoundManager class holds the responsibility of all the audio functionalities. Different MediaPlayer objects are instantiated in order to play specific winning, drawing, and timer warnings sounds.

**Code Implementation :**

|  |
| --- |
| class SoundManager(private val context: Context) {  private var winSound: MediaPlayer? = MediaPlayer.create(context, R.raw.winning)  private var drawSound: MediaPlayer? = MediaPlayer.create(context, R.raw.nothing)  private var timerSound: MediaPlayer? = MediaPlayer.create(context, R.raw.timer2)  fun playWinSound() = winSound?.start()  fun playDrawSound() = drawSound?.start()  fun playTimerSound() = timerSound?.start()  fun release() {  winSound?.release()  drawSound?.release()  timerSound?.release()  }  } |

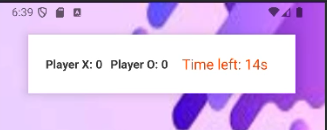
### Enhanced UI

It makes the UI a bit nicer using styles and a ConstraintLayout. The score section is shown in a LinearLayout and there are two buttons for resetting game or scores.

Code Implementation

|  |
| --- |
| <LinearLayout  android:id="@+id/score\_board"  android:layout\_width="337dp"  android:layout\_height="74dp"  android:orientation="horizontal"  android:gravity="center"  android:background="@color/white"  android:elevation="8dp">  <TextView android:id="@+id/playerXScore" android:text="Player X: 0" />  <TextView android:id="@+id/playerOScore" android:text="Player O: 0" />  </LinearLayout> |

**Output:**



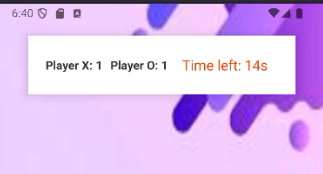
### Result Dialog

At the end of a game an AlertDialog is shown displaying result. This enables directly starting new game from a dialog.

**Code Implementation :**

|  |
| --- |
| private fun showGameResult(message: String) {  AlertDialog.Builder(this)  .setMessage(message)  .setPositiveButton(android.R.string.ok) { \_, \_ ->  startNewGame()  }  .show()  } |

**Output:**



# Testing Plan

## Objective

Ensure the application meets functional and usability standards by systematically testing core features and enhancements.

## Test Cases

1. Game Functionality

* Verify that players can take turns accordingly.
* Validate the winning and draw conditions are properly detected.

1. Timer

* Confirm that the timer starts and resets with each turn.
* Check that at the expiration of the timer, the turn would automatically change.

1. Sound Effects

* Ensure the appropriate sounds are played for win, draw, and timer alerts.

1. UI Interactions

* Verify reset and restart buttons work accordingly.
* Ensure the scores are reflected accordingly after each game.

1. Dialog Boxes

* Test result dialog for win and draw
* Ensure new game starts from the dialog.

## Testing Results

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Test Case | Expected Result | Status |
| Game Board | Cell interaction | Mark cell on click | Pass |
| Turn Change | Timer expiration | Change turn | Pass |
| Score Update | After winning | Increment score | Pass |
| Timer Alert | Timer reaches 5 seconds | Play warning sound | Pass |
| Reset Button | Click Reset | Clear game board | Pass |
| Restart Button | Click Restart | Reset scores | Pass |

# Conclusion

This Tic Tac Toe application development is quite an effective way of implementing a very interactive game using Android Studio. It develops on top of some base code structure with major enhancements added to it: tracking scores, adding a countdown timer for each turn, and finally sound effects for making it interactive. These enhancements seamlessly integrate into the original codebase in a way that demonstrates careful planning regarding new features added while maintaining readability and the structure of code.

Key Android development principles developed in this project include the use of TableLayouts, event listeners, and MediaPlayer, providing a lively user-friendly interface. Adding the timers meant managing the instances of the CountDownTimer and making the right balance in the mechanics-gameplay-resource management ratio. Furthermore, it included implementing the sound effects with the usage of the class SoundManager, adding to more interactivity with the application itself from the player.

This also involved a comprehensive test plan concerning the base application and added features; the testing assured that both behave appropriately according to the scenarios engaged. This gameplay application was tested mechanically, functionally-playing game functions, playing music, and different other user interactions: edge cases intended for timeout cases and tie conditions. That this was successfully tested means that such an application realized a sort of completeness needed in the set design goals that give it a complete user experience.

This project serves as a practical demonstration of Android app development skills and highlights the potential for future enhancements. For example, multiplayer functionality over a network or AI opponents could be further added to the game. The structured approach to enhancing the application can be extended to other projects, ensuring scalability, maintainability, and a user-centered design philosophy.

The entire project is thus an integration of software engineering principles with creative problem-solving, which resulted in a robust yet enjoyable application.

# Reference

Toe, T. (2024). *Tic Tac Toe Android Studio Kotlin Tutorial | Noughts & Crosses*. YouTube. Available at: https://youtu.be/POFvcoRo3Vw?si=V9osljVKwLvMHLmn [Accessed 10 Dec 2025].

‌