Task ##P/C – Spike: Core 1

Goals:

The goals for this assignment are:

- **App development**: develop a game which has one single activity allow the player to roll a dice, add or subtract the dice value from the score, reset the score and display the score in different colors based on specific conditions.
    - Implement button listeners to respond to user input.
    - Updating UI elements dynamically based on user actions.
    - Managing app state to retain data across configuration changes like screen rotation.
- **Layout Design**: Create layouts for portrait and landscape orientations, with at least one layout using ConstraintLayout. Ensure that the layouts differ visually and demonstrate the use of constraints rather than hardcoding.
    - Understanding and utilizing ConstraintLayout for flexible and responsive UI design
    - Create separate layouts for portrait and landscape orientations to provide a consistent user experience across different screen sizes and orientations.
- **State Management**: Implement functionality to save the score on rotation using SaveInstanceState to maintain state consistency.
    - Implementing savedInstanceState to persist and restore app state during configuration changes.
- **Localization**: Enable the app to be usable in a second language by externalizing string resources.
    - Externalizing string resources to separate files to facilitate localization.
    - Providing translations for app strings to support multiple languages and regions.
- **Logging**: Use of Log messages for testing and debugging purposes.
    - Using log statements to output debugging information and track the flow of execution within the application.
- **IDE Proficiency**: Demonstrate proficiency in working with the IDE by providing user steps or screenshot showing the development process, layout designs and running the application.

- Navigating and using the IDE tools effectively for app development, layout design, and debugging purposes.
- Understanding project structure, file organization, and build configurations within the IDE environment.

Tools and Resources Used

- Android Studio: used for developing Android applications with Kotlin.
- Kotlin: Programming language used for developing the Android application.
- ConstraintLayout: is used for creating flexible and responsive layouts.
- Github: is used for version control and storage.

Source code: https://github.com/SoftDevMobDev-2023-Classrooms/core1-MinMinis

Knowledge Gaps and Solutions

Gap 1: App development

In this knowledge gap, the focus is on developing an Android app with simple UI consisting of four buttons, one TextView to display the result, and one ImageView to display the image of a dice.

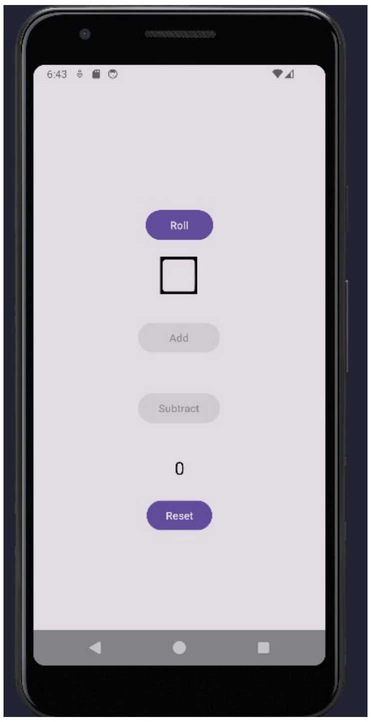


Figure 1: User Interface of Roll Dice Application

The app functionality includes changing the image view and result displayed on the TextView based on user interactions with the buttons. When user click the roll button, the image view of the dice will change randomly (Figure 2). After that, the roll button will be disabled.

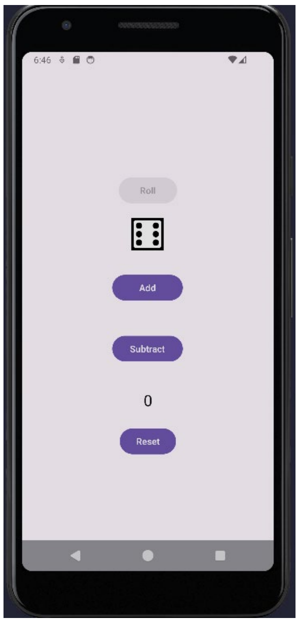


Figure 2: User click Roll button

Then, if the user decides to click Add button, the result will be added with the dice face, and both Add and Subtract number will be disabled (Figure 3), but the Roll button will be enabled again. However, if the user click subtract then the result will subtract with the dice face, and both Add and Subtract button will be disable. If the dice face is greater than the current result, the result can’t go below 0 if the user clicks Subtract button.

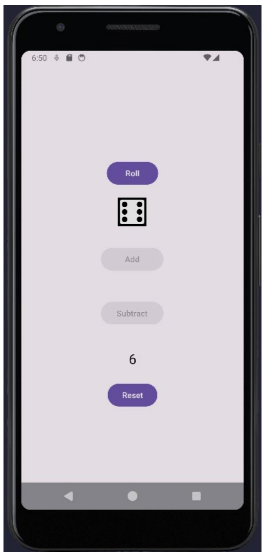


Figure 3: User click Add button

If the user wants to click the Reset button, it will set the result back to 0, both Add and Subtract buttons will be disabled and Roll button will be enabled.

If the result is greater than 20, then the result will be display in red (Figure 4).

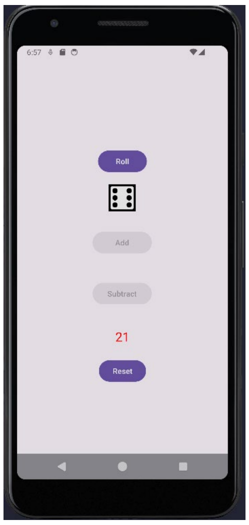


Figure 4: Result is in red color

If the result is smaller than 19, then the result will be set to black color. And if the result is exactly 20, then the result will be in green (Figure 5).

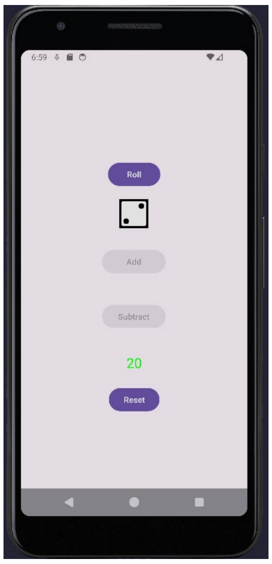


Figure 5: Result is in green color

Gap 2: Layout Design

In this knowledge gap, the focus is on designing layouts using ConstraintLayout to accommodate different screen sizes and orientations, specifically for portrait and landscape modes.

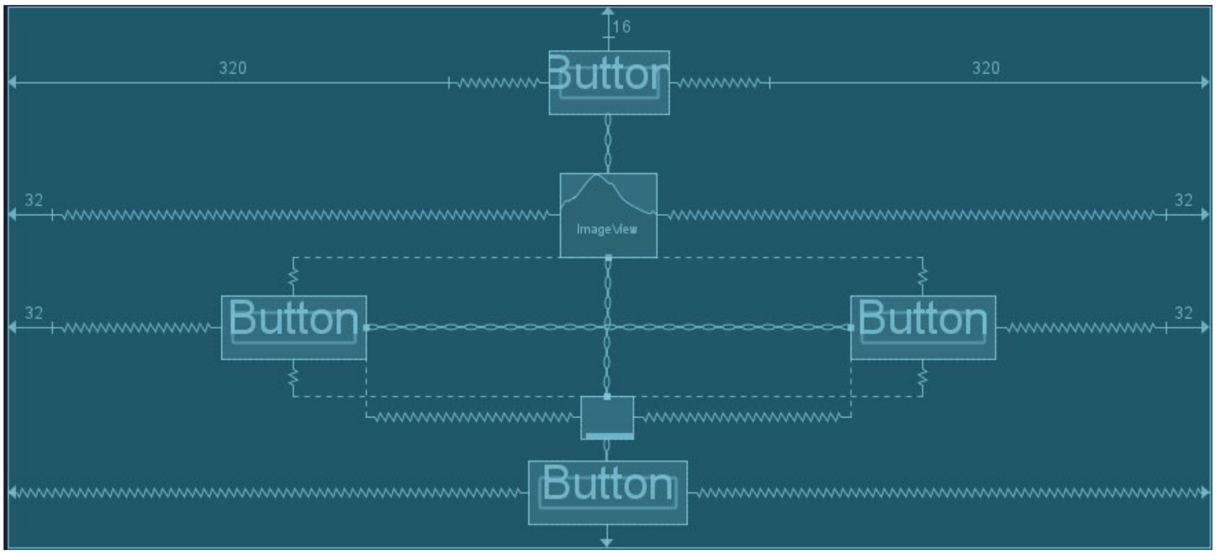


Figure 6: Constraint for landscape layout

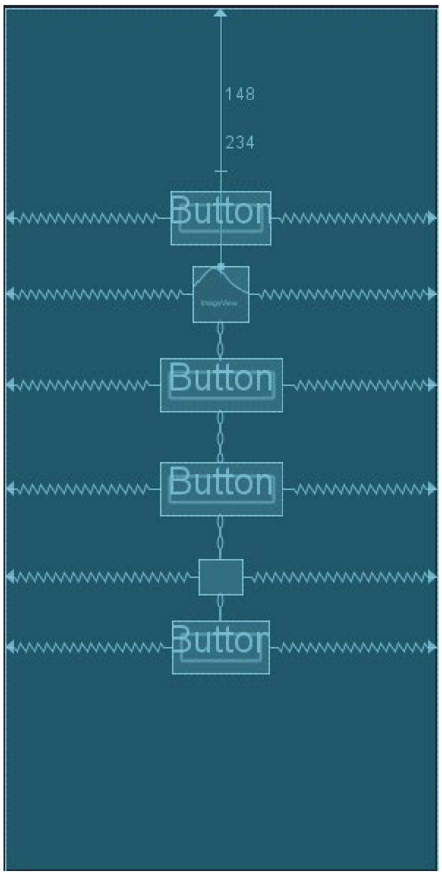


Figure 7: Constraint for portrait layout

Gap 3: State Management

This knowledge gap involves managing the state of key variables in an Android app, specifically the dice value, setvalue, and the result. State management ensures that the app retains its state across configuration like screen rotations. Utilizing technique onSaveInstanceState helps preserve and restore the state of variables (dice, setvalue, result) during lifecycle events. By implementing effective state management, the app can maintain its functionality and user interactions seamlessly even after configuration changes.

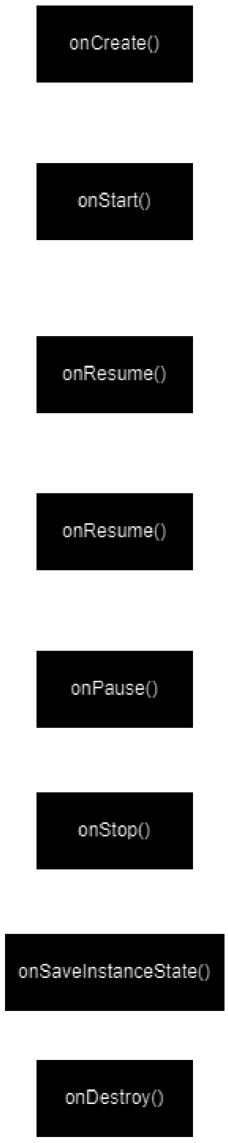
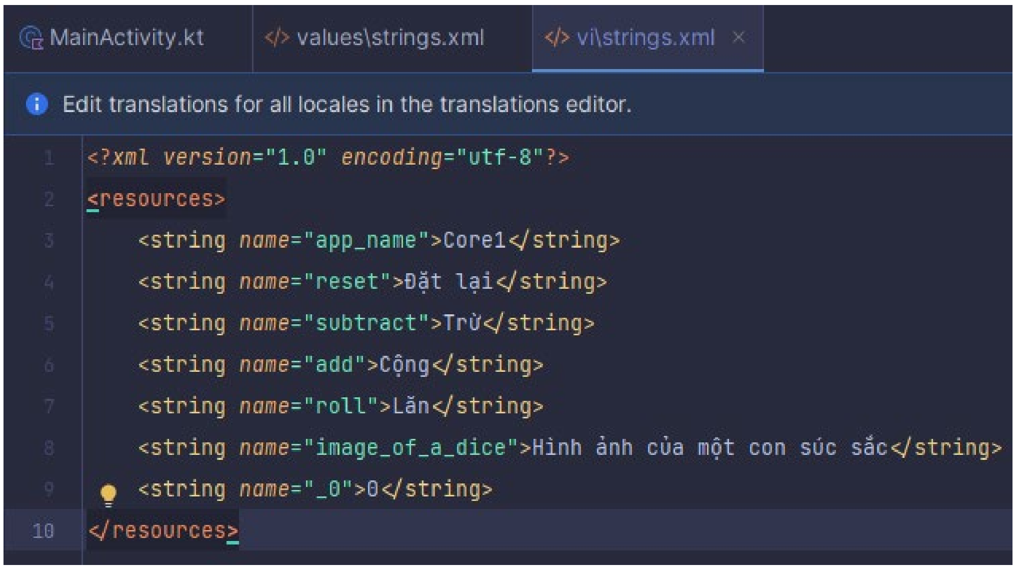


Figure 8: Lifecycle

Gap 4: Localization

Localization involves adapting an app to support multiple languages and regions, ensuring it can be used by a diverse audience. One essential concept in localization is string externalization, which involves storing all user-facing strings in external resource files rather than hardcoding them directly into the app code. This allows for easier translation and maintenance of different language versions of the app without modifying the source code.

To create a new resource for another language, here are steps to follow: add new Android Resource Directory to *res* folder. Add new locale and choose the language Vietnamese. Then, go to *values-vi* folder inside the *res* folder, and add the translations based on the name (Figure 9).


MainActivity.kt
values\strings.xml
vi\strings.xml
Edit translations for all locales in the translations editor.

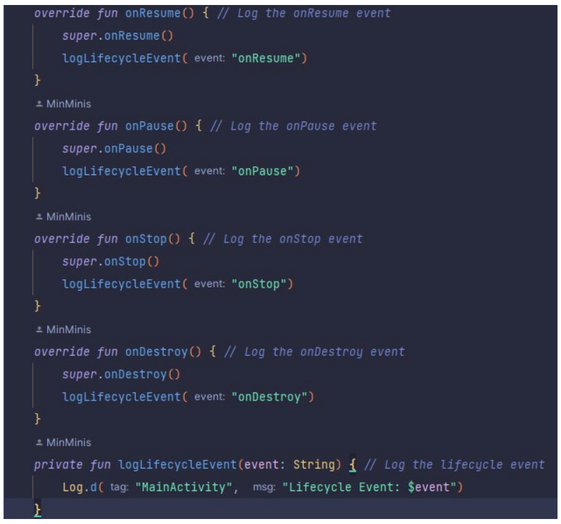
```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Core1</string>
    <string name="reset">Đặt lại</string>
    <string name="subtract">Trừ</string>
    <string name="add">Cộng</string>
    <string name="roll">Lăn</string>
    <string name="image_of_a_dice">Hình ảnh của một con súc sắc</string>
    <string name="_0">0</string>
</resources>
```

Figure 9: Vietnamese vocabulary

Gap 5: Logging

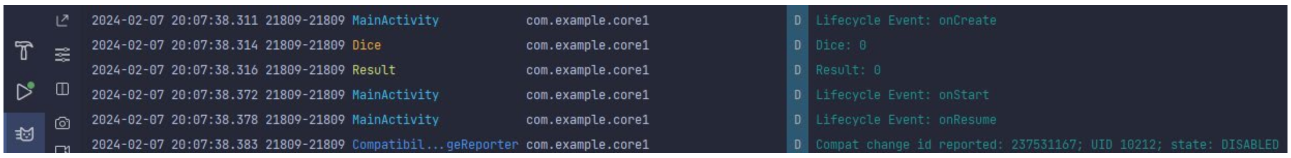
Logging is a crucial aspect of app development used for debugging, testing, and monitoring application behavior. In Android development, logs are commonly used to track the flow of execution, monitor variable values, and identify errors or unexpected behaviors during runtime.

The logLifecycleEvent() method is used to log various lifecycle events of the activity, including onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy(). These logs provide insights into the lifecycle of the activity and help track its state transitions. (Figure 10). All the logs can be viewed in the Logcat in Android Studio.



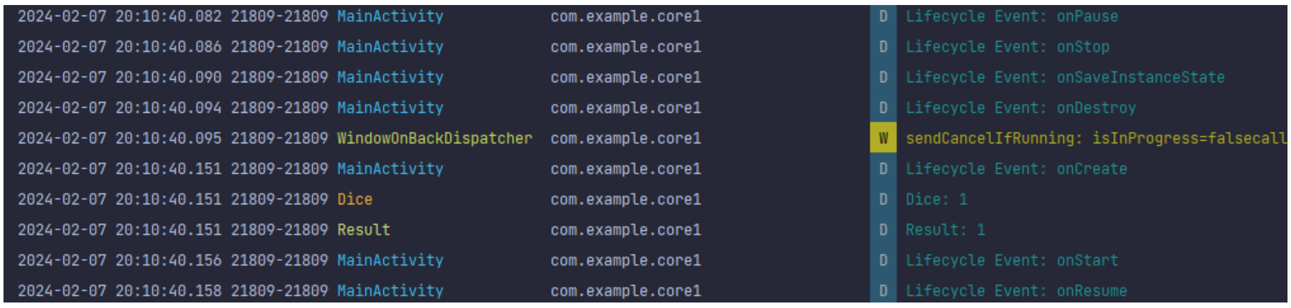
```
override fun onResume() { // Log the onResume event
    super.onResume()
    logLifecycleEvent( event: "onResume")
}
≗ MinMinis
override fun onPause() { // Log the onPause event
    super.onPause()
    logLifecycleEvent( event: "onPause")
}
≗ MinMinis
override fun onStop() { // Log the onStop event
    super.onStop()
    logLifecycleEvent( event: "onStop")
}
≗ MinMinis
override fun onDestroy() { // Log the onDestroy event
    super.onDestroy()
    logLifecycleEvent( event: "onDestroy")
}
≗ MinMinis
private fun logLifecycleEvent(event: String) { // Log the lifecycle event
    Log.d( tag: "MainActivity",  msg: "Lifecycle Event: $event")
}
```

Figure 10: Functions to log out lifecycle event



```
2024-02-07 20:07:38.311 21809-21809 MainActivity            com.example.core1                    D  Lifecycle Event: onCreate
2024-02-07 20:07:38.314 21809-21809 Dice                    com.example.core1                    D  Dice: 0
2024-02-07 20:07:38.316 21809-21809 Result                  com.example.core1                    D  Result: 0
2024-02-07 20:07:38.372 21809-21809 MainActivity            com.example.core1                    D  Lifecycle Event: onStart
2024-02-07 20:07:38.378 21809-21809 MainActivity            com.example.core1                    D  Lifecycle Event: onResume
2024-02-07 20:07:38.383 21809-21809 Compatibil...geReporter com.example.core1                    D  Compat change id reported: 237531167; UID 10212; state: DISABLED
```

Figure 11: Logs when starting the application.



```
2024-02-07 20:10:40.082 21809-21809 MainActivity             com.example.core1                D  Lifecycle Event: onPause
2024-02-07 20:10:40.086 21809-21809 MainActivity             com.example.core1                D  Lifecycle Event: onStop
2024-02-07 20:10:40.090 21809-21809 MainActivity             com.example.core1                D  Lifecycle Event: onSaveInstanceState
2024-02-07 20:10:40.094 21809-21809 MainActivity             com.example.core1                D  Lifecycle Event: onDestroy
2024-02-07 20:10:40.095 21809-21809 WindowOnBackDispatcher   com.example.core1                W  sendCancelIfRunning: isInProgress=falsecall
2024-02-07 20:10:40.151 21809-21809 MainActivity             com.example.core1                D  Lifecycle Event: onCreate
2024-02-07 20:10:40.151 21809-21809 Dice                     com.example.core1                D  Dice: 1
2024-02-07 20:10:40.151 21809-21809 Result                   com.example.core1                D  Result: 1
2024-02-07 20:10:40.156 21809-21809 MainActivity             com.example.core1                D  Lifecycle Event: onStart
2024-02-07 20:10:40.158 21809-21809 MainActivity             com.example.core1                D  Lifecycle Event: onResume
```

Figure 12: Logs when rotating the application.

Open Issues and Recommendations

Recommendations:

- User Interface Improvement: the current user interface design is too simple which can be visual appeal upgraded with more interactive elements and animations.
- Accessibility Considerations: Add more features such as text resizing, color contrast optimization for ensure inclusivity and usability for users with disabilities.