International Journal of Research Publication and Reviews, Vol 6, Issue 5, pp 16876-16879 May 2025

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Interactive Node-Based Application

Tanya[1], *Ashwini Gowda*[2]

[1]B.E Student, Bangalore

[2]H.B, Assistant Professor, Dept. of Elec & Comm Engg., DSCE, Dept. of Elec & Comm Engg., DSCE, Bangalore

[1]spk2tanya2210@gmail.com , [2]ashwini-ece@dayanandasagar.edu

Abstract-

This project focuses on the design and development of an innovative and interactive Android application that leverages gesture- based user input to enable users to create and manipulate graphical nodes and their interconnections in real-time. The application is developed using the Kotlin programming language within the Android Studio IDE, incorporating modern software engineering principles and best practices.

At the heart of the application is a custom View component that interprets and processes multi- touch gestures, such as taps, drags, and swipes, to dynamically generate nodes (represented visually as circles or points) on the screen. When users perform gestures using multiple fingers, the system detects the touch points and creates nodes at those positions. Further gestures allow for the drawing of lines (edges) between these nodes, mimicking the structure of interactive graphs or diagrams. This real-time interaction is rendered on a Canvas, providing users with immediate visual feedback, thus fostering an engaging and responsive user experience. The user interface (UI) is built to be intuitive and fluid, with responsiveness optimized for both newer and mid-range Android devices. The app supports dynamic user inputs without lag or interruption, even with multiple simultaneous touch points.

The application has broad potential applications, particularly in areas such as:

Educational tools (e.g., teaching graph theory, logic circuits, flowcharts), Prototyping interfaces for UI/UX designers, Design and simulation environments for developers or engineers. Development was guided by the Agile methodology, ensuring iterative enhancement, modular code structure, regular testing, and validation on real Android devices to maintain performance and compatibility. The modular architecture makes it easy to extend or adapt the app for different gesture types or rendering styles. In summary, the app successfully combines gesture detection, real-time rendering, and user-centric design principles to provide a powerful, flexible, and user-friendly platform for visual interaction. It lays a strong foundation for future advancements in gesture- driven mobile applications and interactive learning platforms.

Keywords: Android App, Gesture Detection, Multi-touch, Kotlin, Interactive UI

I. INTRODUCTION

In recent years, touchscreen interfaces have become a fundamental aspect of mobile computing, revolutionizing the way users interact with digital content. With the widespread adoption of smartphones and tablets, gesture-based input has evolved into a natural and intuitive means of controlling applications. The rise of multi-touch technology has particularly enabled richer, more complex forms of interaction beyond simple taps and swipes, allowing users to manipulate on- screen elements with multiple fingers simultaneously. This project introduces an interactive node-based Android application that harnesses the power of multi-touch gestures to enable users to create, connect, and manipulate visual elements—referred to as nodes—on the device screen. These nodes serve as the basic building blocks of a visual structure, and users can interconnect them with lines to form diagrams, graphs, or conceptual networks in a seamless and responsive environment. Unlike traditional drag-and-drop interfaces that may feel rigid or slow, this application emphasizes fluid and dynamic interaction, achieved through real-time gesture recognition and rendering using custom View components and Canvas drawing APIs in Android. The application interprets user inputs to:

*Dynamically generate nodes at touch points.
*Allow dragging and repositioning of existing nodes.
*Connect nodes via swipe gestures to form lines, representing relationships or pathways.
The main objective is to provide an interactive, user-friendly platform for visual prototyping and learning. The interface is designed to be minimalistic yet powerful, offering a hands-on experience for tasks such as:
*Drawing flowcharts, logic circuits, or concept maps.
*Prototyping UI layouts or system architectures.
*Simulating graph-based algorithms for edu. or development purposes.

Developed using Kotlin and Android Studio, the project follows the Agile development methodology, promoting modular design, regular testing, and iterative refinement based on user feedback. This ensures that the application remains scalable, maintainable, and compatible with a wide range of Android devices.

By focusing on real-time interaction and intuitive usability, the application demonstrates the potential of mobile platforms to support creative, educational, and design-oriented tasks that traditionally required desktop-based tools. It also lays the foundation for future enhancements such as gesture customization, node labeling, real-time collaboration, or even exportable diagram formats.

II. OBJECTIVE OF INTERNSHIP

2.1 To gain hands-on experience in Android app development using Kotlin and Android Studio by designing and implementing a gesture-based node interaction system with real-time rendering.

2.2 To apply software engineering principles, including modular coding, Agile methodology, and iterative testing, in the development of a responsive and user-friendly mobile application.

2.3 To enhance technical and problem-solving skills through the integration of custom views, multi-touch gesture recognition, and dynamic canvas rendering for educational and design- focused use cases.

III. METHODOLOGY

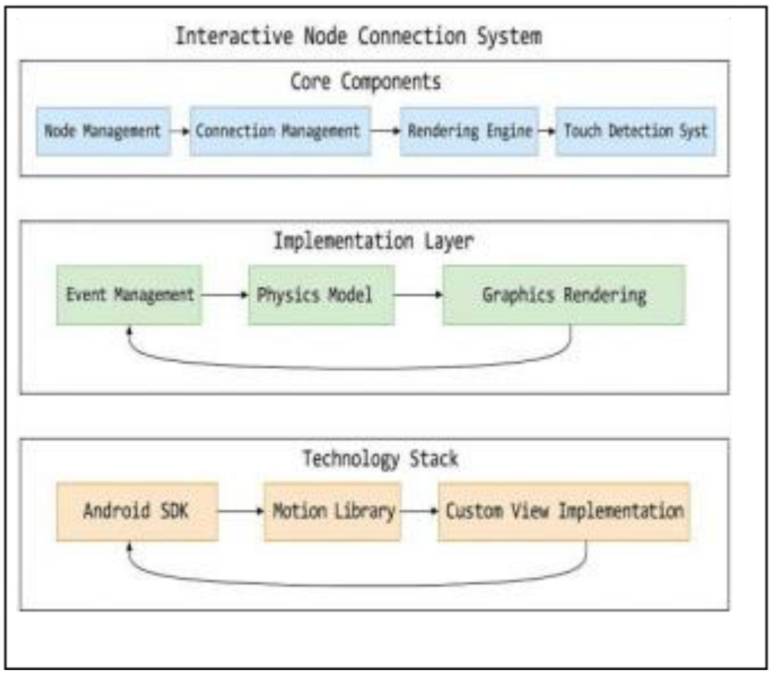

Figure 2.1 Methodology chart

3.1 Core Components Layer

This layer forms the foundational logic of the application and defines the main building blocks required for node interaction.

Node Management: Handles creation, positioning, and identification of each node placed on the canvas through user touch inputs.

Connection Management: Manages logical and visual relationships between nodes by establishing lines/links based on gesture-driven actions.

Rendering Engine: Responsible for dynamically drawing nodes and their connections in real time using Android's Canvas API.

Touch Detection System: Captures and processes multi-touch gestures, identifying actions like tap, drag, and swipe to initiate node creation or connection logic.

3.2 Implementation Layer

This middle layer bridges the interaction between core logic and visual output, encapsulating behavioural and visual dynamics.

Event Management: Collects and interprets user interactions passed from the Touch Detection System, triggering actions like node creation or movement.

Physics Model: Simulates interactions such as node spacing, collision handling, and dynamic feedback (e.g., springy lines, drag effects) to provide a more natural user experience.

Graphics Rendering: Translates computed node and connection states into graphical output on the screen using canvas-based drawing logic. This ensures a smooth and intuitive visual response.

3.3 Technology Stack

This base layer consists of the tools and libraries used for implementing the system and delivering its functionalities.

Android SDK: The foundational framework used to build, run, and deploy the app on Android devices.

Motion Library: Handles gesture tracking, animation transitions, and motion dynamics needed for fluid interaction and responsive UI behaviour.

Custom View Implementation: Encapsulates all rendering and gesture logic into reusable, extendable custom view classes that define how nodes and lines are drawn and updated.

IV. KEY LEARNINGS

4.1 Real-Time Gesture Handling and Multi- Touch Processing

Gained in-depth understanding of how Android captures and processes multi-touch inputs using MotionEvent, enabling accurate gesture-based node creation and connection.

4.2 Custom View and Canvas Drawing in Android

Learned how to design and implement custom View components and use the Canvas API to render dynamic shapes (nodes, lines) in real-time, enhancing responsiveness and user interactivity.

4.3 Modular App Architecture and Agile Development

Applied Agile principles to iteratively build and test the app, while following modular design patterns to separate core logic (like Node Management and Physics Model) from UI rendering and gesture detection.

4.4 Integration of Physics-Based Interactions and Motion Library

Explored how physics models and motion libraries can be used to simulate natural behaviour (e.g., smooth dragging, spring animations), improving the fluidity and intuitiveness of the user interface.

V. OUTCOMES AND IMPACT

A fully functional prototype was developed using Kotlin and Android Studio, capable of capturing multi-touch gestures to create and interconnect nodes dynamically on the screen. The application features a responsive and intuitive user interface, powered by custom view rendering and Canvas-based drawing for real-time feedback. Thorough testing on physical Android devices ensured smooth performance and broad compatibility. Designed with a modular architecture, the app is easily extendable to support additional features such as node labelling, zooming, undo/redo actions, or collaborative editing. It demonstrates the viability of gesture-based interaction as an efficient input method for design- oriented mobile applications. The hands-on development process significantly enhanced the team's understanding of real-time rendering, gesture detection, and Android architecture, contributing to industry-ready technical skills.

VI. CHALLENGES AND SOLUTION

Challenge1:Real-Time Rendering Performance
Problem: Rendering multiple nodes and lines dynamically on the canvas caused lag or flickering, especially with frequent updates.
Solution: Optimized the custom View class by minimizing redraw calls (invalidate()), caching node states, and batching canvas operations to maintain smooth frame rates.
Challenge 2: Node Overlap and Collision Handling
Problem: When nodes were created too close to one another, visual overlap reduced clarity and usability.
Solution: Integrated a basic physics model to apply spacing logic and repel nearby nodes slightly upon creation, ensuring a more readable layout.
Challenge 3: Device Compatibility and Touch Sensitivity
Problem: Gestures behaved inconsistently across devices with different screen sizes and touch sensitivity.
Solution: Calibrated gesture thresholds based on screen DPI and used device-independent pixels (dp) for consistent scaling. Extensively tested on multiple devices to validate uniform behaviour.

VII. CONCLUSION

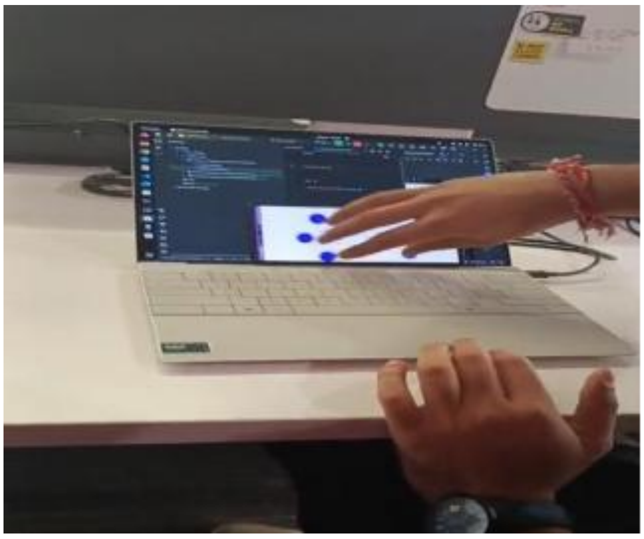

Figure 7.1 Output simulation

The development of the interactive node-based Android application has successfully demonstrated how multi-touch gesture recognition and custom canvas rendering can be combined to create an intuitive and dynamic user interface. By allowing users to create and connect nodes in real-time through natural gestures, the app serves as a powerful tool for visual learning, rapid prototyping, and interactive diagramming.

The use of Kotlin, Android SDK, and custom view components enabled fine-grained control over gesture inputs and graphical outputs, resulting in a seamless and engaging user experience. Through modular architecture and Agile development practices, the project remains flexible for future enhancements such as node labeling, export functionality, or collaborative features.

This work not only met its technical goals but also provided valuable hands-on experience in Android development, gesture handling, and UI/UX design—skills that are directly applicable in modern software engineering and educational tool development.

REFERENCES

[1] Google, "Android Developers Documentation," Google Developers. [Online]. Available: https://developer.android.com. [Accessed: May 24, 2025].

[2] JetBrains, "Kotlin Language Documentation," Kotlin Programming Language. [Online]. Available: https://kotlinlang.org/docs/home.html. [Accessed: May 24, 2025].

[3] Stack Overflow and GitHub, "Stack Overflow & GitHub Discussions - Community-driven insights on custom views, touch handling, and canvas rendering," Stack Overflow and GitHub Community Forums. [Online]. Available: https://stackoverflow.com and https://github. com. [Accessed: May 24, 2025].

[4] R. C. Martin, *Agile Software Development: Principles, Patterns, and Practices*. Upper Saddle River, NJ: Prentice Hall, 2003.

[5] JetBrains and Google, "Android Studio User Guide - Comprehensive documentation on IDE usage, layout design, and performance optimization," JetBrains and Google Developer Documentation. [Online]. Available: https://developer.android.com/ studio. [Accessed: May 24, 2025].