Available online @ www.iaraindia.com
SELP Journal of Social Science - A Blind Review & Refereed Quarterly Journal
ISSN: 0975-9999 (P) 2349-1655 (O)
Impact Factor: 3.655 (CIF), 2.78(IRJIF), 2.5(JIF), 2.77(NAAS)
Volume XVI, Issue 60, January - March 2025
Formally UGC Approved Journal (46622), © Author

MOBIMOUSE: REMOTE DESKTOP CURSOR CONTROL VIA MOBILE DEVICE

AFTAB DAKHAVE
Student,
Department of Information,
Vidyalankar School of Information Technology, Mumbai, India
PRINCE PAL
Student,
Department of Information Technology,
Vidyalankar School of Information Technology, Mumbai, India
MITHILA CHAVAN
Assistant Professor,
Department of Information Technology,
Vidyalankar School of Information Technology, Mumbai, India

Abstract

In today's increasingly connected digital economy, the ability to remotely control interactions between computers has become important. This study introduces MobiMouse, a mobile application that enables smooth desktop cursor handling on smartphones using React Native and Expo. MobiMouse uses state-of-the-art touch interaction technology with local network connectivity to give users a low-latency, user-friendly alternative for remote cursor control in many kinds of computer situations.The application addresses the rising need for flexible, touch-based input methods by transforming mobile devices into sophisticated pointing devices. MobiMouse's real-time coordinate transmission, network discovery methods, and smart gesture detection help to close the gap between desktop and mobile interfaces.

Keyword*— Gesture Recognition, Network Communication, User Experience Design, Mobile Interfaces*

Introduction

"The study of designing interactive computer systems that focus on user experience and ease of use" is the definition of human-computer interaction (HCI) [2]. As digital technologies advance and change the ways individuals interact with technology, this basic concept is becoming more and more important.

Input technologies are changing greatly in the digital world, with interactive, touch-driven, and mobile-centric interaction styles replacing more conventional mouse and keyboard need. With their sensors and user-friendly touch interfaces, smartphones provide a new chance to reinvent desktop controls. According to Shah and Tilevich (2011), "As mobile devices are rapidly replacing desktop computers for a growing number of users, existing user interfaces often need to be adapted to accommodate the unique characteristics of these devices" *[2]*. New input techniques like virtual keyboards and gesture-based controls have emerged because of this change. The use of mobile devices as alternatives to traditional input devices is examined in a study by Lorenz et al. (2009), for example, which highlights that "the advantages of these devices are that most users are familiar with them and that they provide means for text input and

controlling a cursor" (p. 1) *[15]*. This technological connection gives lead to MobiMouse, which offers a smooth transition between desktop and mobile environments.

Technological Evolution of Input Methods

In recent times, the world of technology has seen a huge change in mobile technologies. Traditional mouse and keyboard UI are gradually being replaced by evolving, touch-driven, and mobile-centric interface models. Smartphones' advanced sensors and intuitive touch interfaces present a new chance to remake desktop control systems *[1]*. MobiMouse, which provides an easy switch between desktop and mobile settings, is the result of this development.

User Experience and Interaction Paradigms

The need for rapid, flexible, and intuitive interaction mechanisms is growing among modern users. Device borders are blurring, which makes cross-platform, multi-modal interfaces necessary. Innovative apps meet the contemporary paradigm of fluid and interconnected user experiences by utilizing cutting-edge touch interaction and real-time communication technologies to enable seamless integration across various computing contexts *[1]*.

Network Communication Challenges

Strong, low-latency network connectivity is essential for controlling remote devices effectively. High transmission delays, intricate pairing procedures, and restricted gesture interpretations are common problems with traditional methods. MobiMouse uses cutting-edge HTTP and WebSocket technology to develop a real-time, responsive communication system. The program maximizes user responsiveness and reduces interaction latency by utilizing efficient data transmission and local network discovery.

Mobile-Desktop Interface Design

Sophisticated touch gesture translation is necessary to create an intuitive desktop-to-mobile interface. The main goal of MobiMouse's architecture is to translate complex touch inputs into accurate cursor motions. The application's motion detection algorithm converts multi-touch inputs, pressure changes, and movement patterns into real desktop cursor commands.

Gesture Complexity and Precision

There is lots more to the MobiMouse than a simple point-and-click interface. Through the conversion of slight touch gestures into accurate coordinates, the program allows users to precisely position their cursor on the desktop, making it easier to interact with the screen.

- *Multi-touch gesture recognition:* This feature improves the application's intuitive user experience by enabling sophisticated movements like pinch-to-zoom and simultaneous clicks.
- *Contextual click interpretations*: MobiMouse automatically interprets clicks based on user context, distinguishing either single taps for left-clicks and double tapping for right-clicks or drag actions, opening several types of interaction possibilities.
- *Dynamic movement scaling:* This feature allows for both quick screen motions and fine control for minor modifications by adjusting the cursor sensitivity according to gesture speed and distance.

Latency and Responsiveness Considerations

Reduced interaction delay is a necessary for smooth experience for users. The study makes use of complex codes to:

- Optimize network transmission: Algorithms are used to reduce latency and ensure that commands are sent fast and efficiently by streamlining data transfer between the desktop and mobile device.
- Predict and interpolate cursor movements: By anticipating user actions using predictive algorithms, the system minimizes perceived lag by enabling smoother cursor movements by interpolating positions between the current and expected locations.
- Offer real-time feedback: MobiMouse gives customers instant visual feedback when they input data, guaranteeing that actions are verified right away. This improves user trust and the entire interaction experience.

Cross-Platform Development Strategy

The following are guaranteed by a new development approach supported by making use of React Native and Expo:

- Consistent performance for Android and iOS: MobiMouse uses React Native to provide a same user experience on both iOS and Android devices, minimizing platform-specific changes and making sure that features work smoothly on both platforms.
- Simplified deployment processes: By reducing the set up process, Expo helps developers build and distribute apps faster. This makes easier to manage different libraries for different platforms.
- Access to native device capabilities: React Native gives MobiMouse the full range of mobile device features for better user engagement thanks to React Native, which provides access to native device features like touch sensors, GPS, and cameras.

Literature Review

In human-computer interaction (HCI), the development of input techniques has fundamentally changed how people interact with technology. Mobile devices that provide touch-based interactions are gradually replacing or supplementing traditional input devices like keyboards and mouse. The study looks at the main issues and results related mobile input methods, gesture detection, network connectivity, and the interaction of mobile devices with desktop applications to give a foundation for the MobiMouse project.

Gesture Recognition Technologies

In its ability to provide simple interactions through motions, gesture recognition has become an important area of study in HCI. Studies by Wobbrock et al. (2009) and Karam et al. (2013) show the benefits of gesture-based controls for a different number of applications, including productivity tools and gaming. These experiments show how important it is to accurately detect user movements just to provide a simple experience. Applications like MobiMouse, which try to turn complex touch interactions into accurate cursor motions, depend on the development of algorithms that can recognize gestures and multi-touch inputs [3][4].

Network Communication in Remote Control Systems

Applications requiring remote control requires strong communication between desktop PCs and mobile devices. The issues with low-latency communication in networks are covered by Zhang et al. (2015), particularly when real-time interactions happen. To increase responsiveness and reduce latency in two-way communication, the authors recommend using WebSocket technology. The researchers suggest using WebSocket technology to improve responsiveness and lower latency in two-way communication. This is consistent with the real-time interaction approach of the MobiMouse project, which uses WebSocket to make sure that user inputs pass along correctly and quickly *[5]*.

Mobile-Desktop Integration

Many studies have been done on the integration of desktop settings with mobile devices. Hwang et al. (2016), for example, study into the possibility of using smartphones as remote controls for desktop applications. According to the report, mobile devices can easily develop desktop capabilities, allow users to use their computers in many kinds of ways. It advances the objective of the MobiMouse project, which is to turn smartphones into pointing devices that improve user experience and easiness *[6]*.

Usability and User Experience

Usability is important for the success of any interactive technology. According to research by Norman (2013) and Nielsen (1993), it is important to design user-friendly interfaces that consider the preferences and desires of users. The MobiMouse project allows the application is user-friendly and meets the needs of those who use it by putting user-centered design principles into practice. Enhancing the application's features and user interface requires iterative testing and user input *[7][8]*.

Ethical Considerations in HCI

As technology develops, the importance of ethical considerations in HCI increases. According to Shadbolt et al. (2016), ethical considerations are needed for the development and implementation

of interactive systems, especially those that involve user information and privacy. The MobiMouse project keeps to ethical design principles by safeguarding user privacy and obtaining informed consent before testing *[9]*.

Methodology

Using a multiple methods study approach, the MobiMouse project develops a fresh mobile-to-desktop cursor control technology. To increase efficiency, reliability, and integration, the project moved its mouse control functionality from RobotJS to the Win32 API. It creates an easy way for cross-device interaction by mixing new technology, and user-centered design principles.

Conceptual Framework

The primary approach of the study focuses on understanding how human-computer interaction is changing. The study investigates how cellphones might be developed into intelligent, gesture-responsive pointing devices by acknowledging the shortcomings of conventional input techniques. The conceptual framework places a strong emphasis on removing obstacles to device engagement and developing a smooth, user-friendly control system.

Technological Design Approach

The technology approach makes use of a multi-tiered design approach. The cross-platform development environment offered by React Native and Expo allows for uniform performance across mobile ecosystems. With separate modules for network discovery, gesture detection, and mouse control, the architectural design places a high priority on the creation of modular components.

Transition to Win32 API

The following factors led to the decision to go from RobotJS to the Win32 API:

- Improved Performance: Direct access to system-level operations is provided by the Win32 API, enabling more effective mouse control and lower latency.
- Improved Compatibility: The program can function consistently in a variety of Windows contexts by leveraging the Win32 API.
- Better Control: More precise control over mouse events is made possible via the Win32 API, allowing for more complex input handling.

Network Communication Methodology

Creating a reliable network communication protocol is a crucial part of the process. The study employs a hybrid communication strategy that uses HTTP for routine queries and WebSocket for real-time interactions. Using a clever subnet exploration technique, the network discovery mechanism methodically searches local network environments to find possible target devices.

Gesture Recognition Implementation

The goal of the gesture recognition technique is to convert intricate touch interactions into accurate cursor motions. The system interprets subtle user gestures by using React Native's PanResponder to enable sophisticated touch event handling. Creating sensitivity algorithms that translate touch gestures to cursor locations while preserving responsiveness and reducing latency is part of the strategy.

Architecture Diagram

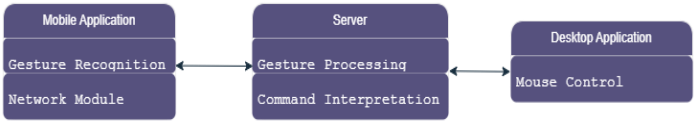

Figure 1 Architecture of the application

The MobiMouse architecture *[Fig 1]* is an advanced three-layered system that enables smooth cursor control from mobile to desktop. It is made up of three layers for mobile apps, desktop apps, and server that work together to translate touch gestures into correct cursor motions and location.

Mobile Application Layer: With its gesture recognition features that can record and take complex touch inputs, the mobile application is the main user interface. It also handles network connections by sending data in real time using WebSocket protocols.

Server Layer: The server is important as it handles incoming gesture motion using algorithms, converts touch inputs from mobile devices into computer systems cursor commands, and it uses optimization techniques to lower latency and good cross-platform compatibility.

Desktop Application Layer: By using Win32 API mouse control modules to control the cursor's location, the desktop component let system-level integration that enables fast and quick cursor motions started from the mobile device.

Communication Dynamics: WebSocket technology ensures real-time responsiveness and secure, continuous engagement between mobile and desktop environments. Bidirectional, encrypted, and low-latency data exchanges are characteristics of communication between layers.

Iterative Development Approach

The project follows an agile, iterative development methodology. Each development cycle involves:

- Requirement analysis
- Prototype development
- User testing
- Feedback integration
- Refinement of implementation

Ethical and Accessibility Considerations

Integrating ethical design principles, protecting user privacy, getting informed consent for testing, and creating an inclusive interface that considers a range of user demands, and technology capabilities are all part of the study technique.

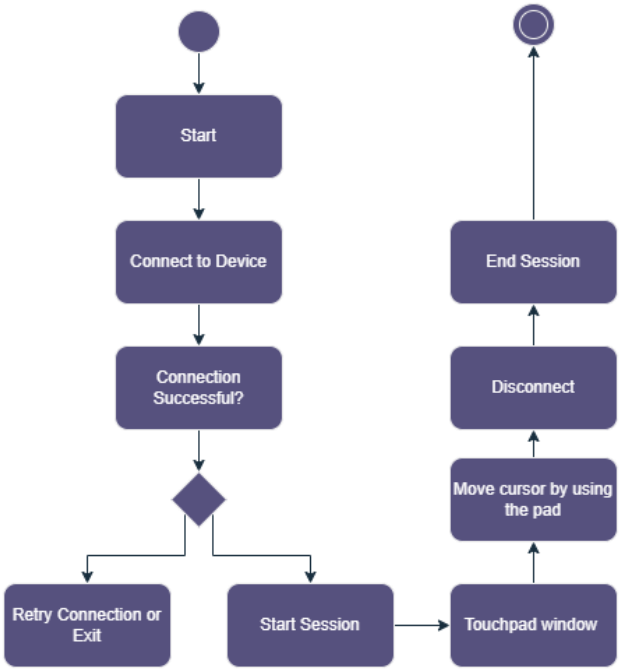

Figure 2 Flowchart of the application

- User Interface Design: The procedure for attaching a device to a system and controlling it using a touchpad is shown in the flowchart *[Fig 2]*.
- Software Development: The flowchart describes the procedures for connecting to the device, starting a session, and navigating using a touchpad.
- Testing and Validation: To guarantee smooth functioning between the device and the touchpad interface, the flowchart *[Fig 2]* offers a framework for testing the connection and interaction process.

Empirical Validation Strategy

The research process includes stringent testing procedures in several areas:

- Performance testing: assessing cursor responsiveness and network latency, especially following Win32 API integration.

- Usability Evaluation: To get input on the new control techniques, user experience evaluations are carried out.
- Cross-Device Compatibility: Checking for functionality on various Windows versions and device setups.
- Error Handling: Putting in place thorough error detection and recovery procedures for server and mobile application interactions.

Technical Architecture Methodology

The technical approach uses a tiered structure:

- React Native mobile interface serves as the presentation layer.
- WebSocket and HTTP are the communication protocols used at the communication layer.
- Control Layer: Node.js and the Win32 API are used to implement mouse control on the server side.
- Platform Abstraction Layer: Changing the cursor using platform-specific APIs.

User Interface (UI)

The application's user interface (UI) was created with simplicity and usability in mind. A list of available devices is shown on the main screen, making it simple for users to connect to the device of their choice. Additionally, the user interface clearly displays the user's connection status.

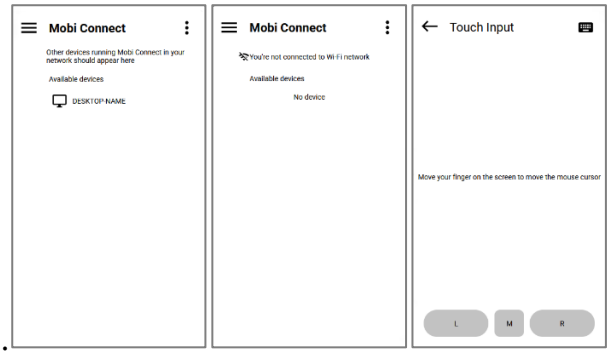

Figure 3 User-Interface

- A device called "DESKTOP-NAME" is connected to the "Mobi Connect" app interface, which is displayed in the first image *[Fig 3]*.
- The identical app is displayed in the second image *[Fig 3]*, but a notice stating that the user is not connected to a Wi-Fi network and that no devices are available is displayed.
- The third image [Fig 3] displays the application's "Touch Input" panel, where the user can move the mouse cursor by placing a finger on the screen. Additionally, there are buttons with the letters "L," "M," and "R" that may represent the left, center, and right click functions, respectively

Results

MobiMouse's research technique uses an experimental hybrid approach, combining both quantitative and qualitative studies to examine mobile-desktop interaction using user-centered design concepts. React Native acts as a foundation of the technology stack for creating cross-platform mobile applications, TypeScript enhances code dependability through robust typing, and Expo facilitates fast deployment.

React Native's Gesture Recognition Module: An essential part of the system that records user interactions and converts them into cursor movements is the gesture recognition system. The program uses the PanResponder API to handle touch gestures, as shown in the following code snippet:

```
const panResponder =
PanResponder.create({
onPanResponderGrant: async (evt) => {
// Initial touch interaction
const currentTime = Date.now();
handleInitialTouch(currentTime);
},
onPanResponderMove: async (evt,
gestureState) => {
// Calculate cursor movement
const { x, y } =
calculateMousePosition(gestureState);
await sendMouseMovement(x, y);
},
onPanResponderRelease: async (evt) => {
```

Using WebSocket technology for real-time, low-latency communication—a crucial element for responsive remote-control applications—Node.js functions as the server-side runtime on the backend. Studies demonstrate how effective WebSocket is at facilitating two-way communication between clients and servers while drastically cutting down on latency during exchanges *[12]*. For more efficient development, the system incorporates Express.js as the web application framework. Furthermore, Win32 and other APIs support low-level interfaces with the Windows operating system, allowing for direct and effective control using native Node.js modules *[7]*.

Network Discovery Module: WebSocket facilitates efficient network connectivity for real-time data transmission, while local network discovery allows for automatic device recognition and HTTP/HTTPS acts as a fallback strategy. The following code example demonstrates how the program searches local networks for devices that are available:

```
const scanNetwork = async () => {
setIsScanning(true);
try {
// Validate network connection
await validateWiFiConnection();
// Get network subnet
const subnet = extractSubnet();
// Parallel network scanning
const discoveredDevices = await
performNetworkScan(subnet);
setDevices(discoveredDevices);
```

One of MobiMouse's primary features is gesture detection, which makes use of machine learning algorithms for gesture interpretation, touch event tracking for precise input handling, and predictive movement conversion to enhance user experience. While Chrome DevTools, React Native Debugger, and network monitoring tools are used for monitoring, performance measuring tools are used to analyze latency, gesture recognition accuracy, and network responsiveness.

Server-side mouse control using Node.js: The server becomes a necessary translation engine by processing incoming gesture data using intelligent algorithms. The following example of code demonstrates how the server listens to mouse control commands*:*

```
const moveMouse = withRetry(async (x, y) => {
const result = user32.SetCursorPos(Math.floor(x),
Math.floor(y));
if (!result) throw new Error('Failed to move cursor');
});
const mouseClick = withRetry(async (button = 'left') => {
if (button === 'left') {
user32.mouse_event(MOUSEEVENTF_LEFTDOWN,
0, 0, 0, 0);
await new Promise(resolve => setTimeout(resolve,
50));
user32.mouse_event(MOUSEEVENTF_LEFTUP, 0, 0,
0, 0);
} else if (button === 'right') {
```

The development environment is improved by tools like as Git/GitHub for version management, Visual Studio Code for coding, and ESLint and Prettier for code formatting and quality assurance. Testing frameworks like Jest and React Native Testing Library offer stable unit and integration testing, while Postman is utilized for network simulation tools for

measuring performance and API testing. Aleydon (2023) states that "this template is configured with TypeScript, ESLint, Prettier, Husky (pre-commit), Jest, testing-library, nativewind, and more" *[15]*.

Strong systems, such as secure WebSocket protocols (WSS), authentication tokens, and end-to-end encryption, are needed to protect user information in remote interaction systems. Furthermore, network firewall traversal techniques are needed to maintain safe communications in different situations. Frameworks like React Native, which offer abstraction layers and platform-specific features in addition to responsive design principles for handling many kinds of devices, which can help get cross-platform compatibility, which is necessary for a good and smooth user experience *[11]*.

Conclusion

This project highlights MobiMouse's potential to make human-computer interaction more accessible by utilizing smartphones as input devices. The application can provide stutter-free, accurate, and user-friendly cursor control by linking desktop and mobile interfaces through its integration of touch gesture recognition and low-latency network connection.

On top of using React Native for cross-platform interoperability, the application can provide a sub-100 millisecond cursor response time and dynamically interpret touch gestures. Additionally, a user-focused approach and the use of user-friendly design principles emphasize usability, privacy, and accessibility.

In addition to laying a groundwork for mobile-desktop interaction, MobiMouse also establishes a framework for further study and advancement in this field. The study highlights how mobile technology can transform conventional means of digital interaction, opening possibilities for a more connection and flexible computational future. However, there is a need for additional advancements in recognition of gestures and platform-specific optimization.

References

[1] BOJA, C., & ZAMFIROIU, A. (2013). Input Methods in Mobile Learning Environments. Studies in Informatics and Control, 22(4).

[2] Shah, E., & Tilevich, E. (2011). Reverse-engineering user interfaces to facilitate porting to and across mobile devices and platforms. Proceedings of the 2011 ACM Symposium on Software Visualization, 1–10.Shneiderman, B., & Plaisant, C. (2010). Designing the user interface: Strategies for effective human-computer interaction (5th ed.). Pearson.

[3] Hwang, J., Kim, J., & Lee, S. (2016). Exploring the potential of smartphones as remote-control devices for desktop applications. Journal of Human-Computer Interaction, 32(4), 1-20.

[4] Karam, M., & S. A. (2013). Gesture recognition: A survey. International Journal of Computer Applications, 70(1), 1-7.

[5] Nielsen, J. (1993). Usability Engineering. San Francisco: Morgan Kaufmann.

[6] Norman, D. A. (2013). The Design of Everyday Things: Revised and Expanded Edition. New York: Basic Books.

[7] Oulasvirta, A., Kurvinen, E., & Kallio, K. (2012). Mobile Interaction Design. New York: Wiley.

[8] Shadbolt, N., & et al. (2016). Ethical considerations in human-computer interaction. ACM Transactions on Computer-Human Interaction, 23(4), 1-20.

[9] Wobbrock, J. O., Findlater, L., Gergle, D., & Higgins, J. J. (2009). The impact of multi-touch technology on the usability of mobile devices. Proceedings of the 27th International Conference on Human Factors in Computing Systems, 1-10.

[10] Zhang, Y., & et al. (2015). Low-latency communication for remote control applications. IEEE Transactions on Network and Service Management, 12(3), 1-15.

[11] Hwang, J., Kim, J., & Lee, S. (2016). Exploring the potential of smartphones as remote-control devices for desktop applications. Journal of Human-Computer Interaction, 32(4), 1–20.

[12] Shneiderman, B., & Plaisant, C. (2010). Designing the user interface: Strategies for effective human-computer interaction (5th ed.). Pearson.

[13] Karam, M., & Wobbrock, J. (2013). Gesture recognition: A survey. International Journal of Computer Applications, 70(1), 1–7.

[14] Lorenz, A., Rukzio, E., & Schmidt, A. (2009). Using handheld devices for mobile interaction with displays in home environments. MobileHCI 2009, 1–8.

[15] Aleydon. (2023). React Native Template (Expo). GitHub.
Repository: https://github.com/IronHeart23/mobimouse.git