WEB-DRIVEN CROSS-PLATFORM SIMULATION OF WINDOWS 11 DESKTOP

Somaprakash Rath 4th Year, Department of CSE, Gandhi Institute for Technology, BPUT, India
somaprakash2021@gift.edu.in
Animesh Khatua 4th Year, Department of CSE, Gandhi Institute for Technology, BPUT, India
animesh2021@gift.edu.in
Mr. Jagannath Ray Assistant Professor, Department of CSE, Gandhi Institute for Technology, BPUT, India

Abstract:
This project, titled "Web-Driven Cross-Platform Simulation of Windows 11 Desktop" , explores the implementation of the Windows 11 desktop environment using modern web technologies like React, CSS (SCSS), and JavaScript. The objective is to provide users with a seamless, interactive, and cross-platform Windows 11-like experience directly in their web browsers, regardless of the operating system they are using. By leveraging the power of React, a popular JavaScript library for building user interfaces, this project mimics core elements of the Windows 11 desktop, such as the taskbar, start menu, window management, and desktop icons. SCSS is used for advanced styling, ensuring that the UI maintains a high degree of flexibility and responsiveness across different screen sizes. The entire application is designed to run in any modern web browser,
providing users with the ability to experience the familiar functionality of Windows 11 without requiring native software installation.
This simulation allows for efficient cross-platform compatibility, enabling Windows 11's design and usability to be accessible on macOS, Linux, and other operating systems. Through this project, users gain insights into how front-end web development tools can be utilized to replicate a desktop environment traditionally tied to a specific operating system, pushing the boundaries of web-based UI/UX design.

Keywords:
HTML, SCSS, JAVASCRIPT , React.js , React Draggable / RND , Redux , Extension packs

CHAPTER 1 : INTRODUCTION

The "Cross-Platform Windows 11 Desktop Simulation with Web Technologies" project aims to recreate a familiar and
immersive Windows 11 desktop experience using modern web technologies. By leveraging React, CSS (SCSS), and JavaScript, this project bridges the gap between traditional desktop environments and the power of web-based solutions, enabling users to interact with a simulated Windows 11 interface directly in their browsers. The goal is to provide a seamless, responsive, and cross-platform solution that mimics the core functionalities of Windows 11, such as the taskbar, start menu, window
management, and desktop icons, while ensuring compatibility across a range of operating systems, including macOS, Linux, and Windows.
Win11React is designed to run in any modern web browser, providing users with the ability to experience the familiar functionality of Windows 11 without requiring native software installation. This cross-platform compatibility ensures that users on macOS, Linux, and other operating systems can access and interact with the simulation seamlessly.

CHAPTER 2 : OVERVIEW

The "Cross-Platform Windows 11 Desktop Simulation with Web Technologies" project focuses on replicating the Windows 11 desktop environment using modern web technologies, such as React, SCSS, and JavaScript. The project aims to provide users with an interactive, cross-platform desktop experience that mimics the functionality of the Windows 11 operating system, all within the web browser. This simulation eliminates the need for native software installation and enables users from various operating systems, including Windows, macOS, and Linux, to interact with a familiar desktop

ISSN: 2278-4632
Vol-15, Issue-07, July: 2025

interface.

At its core, the simulation features key elements of the Windows 11 desktop, such as the taskbar, start menu, window management, and desktop icons. These elements are built with React, ensuring smooth transitions and real-time interactivity.

React's component-based structure allows for efficient management of the interface, enhancing the scalability and responsiveness of the design. The use of SCSS allows for a more dynamic and maintainable styling approach, making it easier to ensure that the interface adapts seamlessly to various screen sizes and devices.

CHAPTER 3 : TECHNOLOGIES USED

The "Cross-Platform Windows 11 Desktop Simulation with Web Technologies" project leverages a range of modern web development tools and technologies to recreate a functional and interactive Windows 11-like experience. Below is an overview of the key technologies used in this project:

1. React

React is a popular JavaScript library used for building dynamic and interactive user interfaces. In this project, React plays a critical role in managing the components and structure of the simulated desktop environment. By utilizing React's component- based architecture, we can efficiently manage the UI elements such as the taskbar, start menu, windows, and icons. React ensures smooth state management and real-time updates to the interface, providing a seamless user experience similar to that of a native desktop operating system.

2. SCSS (Sassy CSS)

SCSS is a powerful CSS preprocessor that enhances the capabilities of traditional CSS. SCSS offers features like variables, nesting, mixins, and partials, making it a more flexible and efficient way to style the application. For this project, SCSS is used to create a responsive and scalable layout that adjusts seamlessly across different screen sizes, providing users with an optimal experience whether they are using a mobile device, tablet, or desktop.

3. JavaScript

JavaScript is the backbone of the interactivity and dynamic features of the simulation. In this project, JavaScript is used for various tasks such as handling user events (e.g., opening and closing windows, interacting with icons), managing application state, and implementing functionalities like dragging and resizing windows. JavaScript allows us to create the behavior and logic that makes the simulation feel like a native desktop environment.

4. HTML5

HTML5 provides the structure for the application, ensuring that the simulation's layout is correctly represented in the browser. It is used to define the basic structure of the desktop environment, such as the taskbar, window containers, start menu, and icons, which are then styled and made interactive with CSS and JavaScript. HTML5 ensures compatibility across different browsers and devices, making it a key technology in this project's cross-platform approach.

5. CSS Grid & Flexbox

In this project, CSS Grid is primarily used to manage the layout of the desktop environment, including the arrangement of icons and taskbar components. Flexbox is employed for tasks requiring alignment and distribution of space, such as positioning windows and resizing them.

CHAPTER 4 : CHALLENGES AND SOLUTION

Throughout the development of the "Cross-Platform Windows 11 Desktop Simulation with Web Technologies" project, several challenges were encountered. Each challenge was addressed with a thoughtful solution to ensure the application was functional, responsive, and efficient across various platforms. Below are the key challenges faced and the solutions

ISSN: 2278-4632
Vol-15, Issue-07, July: 2025

implemented to overcome them:

Challenge: Achieving Cross-Browser Compatibility

1. One of the primary challenges was ensuring that the desktop simulation worked seamlessly across different browsers (Chrome, Firefox, Safari, Edge, etc.), each of which may handle web standards slightly differently. Ensuring that the simulated Windows 11 environment looked and functioned consistently across these platforms required attention to detail in the code.
2. Challenge: Maintaining Responsive Design

Given the nature of the simulation, ensuring that the interface would look and function properly on different screen sizes, ranging from mobile devices to large desktop monitors, presented a significant challenge. The desktop layout, taskbar, and windows needed to adapt to a wide range of device resolutions without sacrificing usability or appearance.

3. Challenge: Simulating Desktop-Like Interactivity in a Web Environment

Replicating the desktop-like interactions of opening, closing, dragging, and resizing windows in a web environment was another challenge. Unlike traditional desktop applications, which are built specifically for interacting with the OS, web browsers have limitations regarding handling native-like interactions.

CHAPTER 5 : USER EXPERIENCE AND INTERFACE DESIGN

The "Cross-Platform Windows 11 Desktop Simulation with Web Technologies" project places a strong emphasis on creating an intuitive, engaging, and user-friendly experience. By replicating the familiar elements of Windows 11, the goal was to provide users with a seamless and interactive desktop environment within the browser, ensuring a smooth and responsive experience. Below are the key considerations and principles guiding the UX and interface design of the simulation:

1. Familiar Desktop Experience

One of the core goals of the simulation is to recreate a user experience similar to Windows 11, which users are already familiar with. The interface design mimics key features such as the taskbar, start menu, system tray, and window management

(dragging, resizing, minimizing, etc.), offering a seamless transition from the native operating system to a web environment.

The goal was to ensure that users could immediately feel comfortable with the interface, making the transition between native Windows 11 and the web-based simulation as smooth as possible.

2. Responsive and Adaptable Design

Given the diverse range of devices and screen sizes users may access the simulation from, responsive design was a key focus. The simulation adjusts fluidly to varying screen resolutions, from mobile phones to large desktop monitors, ensuring that the interface remains functional and visually appealing regardless of the device. This is achieved through the use of CSS Grid,

Flexbox, and media queries, enabling the UI to adapt based on screen size and resolution.

3. Consistency and Visual Appeal

The design maintains consistency with the Windows 11 aesthetic, including rounded corners, subtle animations, and smooth transitions. The visual design was crafted to feel modern and familiar, reflecting the latest design trends while staying true to the Windows 11 interface. SCSS was used to create a flexible and maintainable design system, ensuring that the interface remains visually appealing and responsive across devices.

CHAPTER 6 : CROSS-PLATFORM COMPATIBILITY

Ensuring that the Windows 11 desktop simulation functions seamlessly across different platforms is a critical aspect of the project. This guarantees that all users, regardless of their operating system or device, can access and enjoy the simulation

without any issues. To achieve this, several strategies and testing methods are used to ensure the cross-platform compatibility of the simulation.

Since web browsers can render the same webpage differently, it is important to ensure that the simulation works properly across all major web browsers. The simulation is tested on popular browsers

such as Google Chrome, Mozilla Firefox, Apple
Safari, and Microsoft Edge to ensure that it is fully compatible with different rendering engines used by these browsers. Testing is automated with tools like BrowserStack and Sauce Labs, which allow the simulation to be tested on multiple browsers and versions simultaneously. These tools make it easier to identify any issues with how the simulation appears or behaves on
different browsers, ensuring that all users receive a consistent experience.
Different devices use different input methods—touch gestures for mobile devices and mouse interactions for desktops. The
simulation is designed to handle both types of input effectively. For example, users on mobile devices can swipe, tap, or pinch to zoom and interact with the simulation, while desktop users can use the mouse to drag windows, click on buttons, and
interact with elements. Special attention is given to ensure that the transition between these two input methods is smooth and that there are no issues with usability, no matter the device type. This guarantees that the simulation is intuitive and easy to use, whether accessed on a touchscreen or through a traditional mouse and keyboard setup.

CHAPTER 7 : SECURITY AND DATA PRIVACY

Ensuring the security and privacy of users is a critical aspect of the cross-platform Windows 11 desktop simulation. As the
simulation operates in a web environment, it is essential to take a proactive approach in protecting user data and ensuring that the system is secure from any potential threats. Below are the key security and privacy considerations implemented to
maintain a secure and private environment for all users:
access to the system's underlying resources and preventing malicious activities from spreading. This sandboxing mechanism ensures that any potentially harmful scripts or actions remain contained within the web environment, reducing the risk of unauthorized system access. This adds an additional layer of protection by isolating the simulation from the rest of the
operating system and minimizing potential security vulnerabilities.
All communication between the client (user's browser) and the server is encrypted using HTTPS (Hypertext Transfer Protocol Secure). This encryption ensures that any data transmitted between the client and the server remains secure and cannot be
intercepted by third parties during transmission. By utilizing HTTPS, the integrity and confidentiality of the data are maintained, which is crucial for protecting sensitive user information, even if the network is compromised.
User sessions are managed securely to prevent unauthorized access to the simulation. Sessions are encrypted using industry- standard protocols, and user login states are stored temporarily and securely. Session tokens are used to maintain user states without exposing any sensitive data. Additionally, session timeouts are implemented to automatically log users out after a
period of inactivity, reducing the risk of unauthorized access in case a user forgets to log out.
To maintain the security of the simulation over time, regular security audits are performed to identify any potential vulnerabilities or weaknesses. The system is regularly updated to address emerging security threats, ensuring that the
simulation remains resilient against cyberattacks. Security patches are promptly applied, and new technologies or methods are integrated as necessary to further strengthen the security framework.

CHAPTER 8 : PERFORMANCE OPTIMIZATION

To ensure that the cross-platform Windows 11 desktop simulation runs efficiently, several strategies for performance optimization have been employed:
Resource Management in Simulation

- Efficient Asset Loading: Large assets, such as images and icons, are optimized for quick loading to minimize delays during the initial setup. Lazy loading techniques are used to load assets only when needed, reducing the initial load time.

- Memory Management: Memory usage is carefully managed by disposing of unused resources and implementing caching mechanisms to avoid memory leaks.

Benchmarking and Testing Performance

- Performance Benchmarks: The simulation undergoes regular benchmarking using tools like Lighthouse and browser developer tools to identify performance bottlenecks. Metrics such as Time to Interactive (TTI) and First Contentful Paint (FCP) are monitored.
- Cross-Platform Performance Testing: The simulation is tested on various devices, operating systems, and browsers to ensure that it performs optimally across all supported platforms, with specific focus on load times and responsiveness.

CHAPTER 9 : Backend Services and Tools

A robust backend can power user accounts, cloud sync, and analytics for the simulator. Possible services include:

The extension follows a typical MV3 flow: the React-based UI (in a popup or browser tab) sends messages to the background service worker, which handles all data fetching and logic. The service worker uses Chrome's extension APIs (e.g. chrome.storage, message passing) and then calls out to external cloud services (auth, database, storage, analytics) over HTTPS. This separation aligns with Chrome's MV3 model of UI vs. background threadscss- tricks.comcodimite.ai. The diagram below illustrates this flow:

- Ofline Support: MV3 extensions can use the service worker's Cache API or chrome.storage to cache assets and data. Although extensions aren't PWAs, you can preload the UI shell into the service worker so that the desktop UI loads quickly on startup. Service workers in MV3 can precache static files for limited ofline functionalitycodimite.aicodimite.ai. For example, cache the main HTML/CSS/JS bundle so the UI opens even with no network, while dynamic data can sync when online.

CHAPTER 10 : LIMITATIONS

- Imperfect Simulation: Native Windows 11 features, such as system-level notifications and advanced window management, can't be fully replicated in a web environment.
- Rendering Issues: Complex graphics or high-fidelity effects may not perform as well in a web-based environment.
- Inconsistent Experience: Minor differences may appear across devices and browsers, affecting user experience. Simulation Accuracy
- Interface Limitations: The simulation may not achieve pixel-perfect accuracy in replicating the Windows 11 UI or system behaviors.
- Complex Interactions: Features requiring deep OS integration, such as virtual desktops, cannot be fully simulated. Security and Privacy
- Restricted Data Access: Web-based simulations cannot access secure data or system settings.
- Privacy Risks: Web-based systems might expose user data, requiring careful management of privacy concerns. Performance and Scalability

CHAPTER 11 : CONCLUSION

The "Cross-Platform Windows 11 Desktop Simulation with Web Technologies" project successfully demonstrates the potential of modern web technologies to replicate a familiar, interactive desktop environment directly within a web browser. By leveraging tools like React, SCSS, and JavaScript, this project brings the key elements of the Windows 11 desktop experience— such as the taskbar, start menu, and window management—into a cross-platform, responsive simulation accessible on any device.

This project not only showcases the capabilities of front-end development but also highlights the possibilities for creating rich, interactive desktop-like experiences using web technologies. It emphasizes the potential for cross-platform accessibility, where users can enjoy a seamless experience

without the need for native software installation, making it available to individuals on different operating systems, including Windows, macOS, and Linux.
Overall, this project is a significant step forward in demonstrating how web technologies can create complex, immersive user interfaces that extend beyond traditional websites or applications. It sets a strong foundation for future developments in web- based simulations and applications, paving the way for even more advanced, interactive, and cross-platform experiences in the web development space.

CHAPTER 12 : FUTURE ENHANCEMENTS

While the "Cross-Platform Windows 11 Desktop Simulation with Web Technologies" project successfully replicates the core functionality and design of a Windows 11 desktop environment, there is always room for improvement and expansion. Future enhancements can further improve user experience, add more features, and make the simulation even more versatile. Below are some potential areas for future development:

1. Integration of Native Desktop Features

Currently, the simulation provides a basic level of interaction with desktop elements like windows, taskbars, and icons. Future enhancements could involve adding more native desktop-like features, such as:

•File System Simulation: Introducing a file management system where users can create, open, and save files within the simulation. Users could simulate file navigation, moving files between folders, and opening documents or images.

•Drag-and-Drop Functionality: Enabling users to drag and drop files or windows between desktop areas, mimicking the real desktop environment more closely.

2. Advanced Window Management

Although the current implementation allows for basic window management (dragging, resizing, minimizing), there is potential for more advanced features:

•Multiple Virtual Desktops: Implementing the ability for users to create and manage multiple virtual desktops within the simulation, allowing for a more organized workspace, similar to what is available in native Windows 11.

•App Store Simulation: Creating a basic app store within the simulation where users can "download" and open apps that are web-based, allowing for an enriched desktop-like experience.

3. Enhanced Performance and Optimization

As the simulation grows in complexity with more features, optimizing performance will be essential to maintain smooth functionality, especially on devices with limited resources:

•Lazy Loading of Resources: Further optimizing resource loading by only loading essential components when needed, especially when dealing with multiple windows and applications.

•Improved Memory Management: Handling memory more efficiently to prevent performance degradation when opening multiple windows or apps, ensuring that the simulation remains responsive and fluid.

CHAPTER 13 : REFERENCES

React Documentation https://reactjs.org/ CSS Grid Layout
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout SCSS Documentation
https://sass-lang.com/ JavaScript: The Good Parts
Crockford, D. (2008). JavaScript: The Good Parts. O'Reilly Media. Building Cross-Platform Web Applications with React
Gusev, V. (2018). Building Cross-Platform Web Applications with React. Packt Publishing.
Windows 11 Design Guidelines
https://developer.microsoft.com/en-us/windows/apps/design/ Responsive Web Design
https://www.w3.org/TR/css3-mediaqueries/ Windows 11 Design Guidelines
https://developer.microsoft.com/en-us/windows/apps/design/ Stack Overflow
https://stackoverflow.com/questions

