Implementation of split-screen Facility in web applications

[1]Mrs.K. Dhivya, [2]Dr. M. Raju, [3]B. Aishwarya ,[4] A. Bhanu Prasad , [5]B. Ravi

[1]Assistant Professor *Department Of Computer Science and Engineering*

[2]Associate Professor *Department Of Computer Science and Engineering*

[345]*Department Of Computer Science and Engineering*

UG Scholar

Abstract- The implementation of a split-screen facility in a web application represents a user interface design strategy that enhances multitasking and content visibility.

This abstract explores the conceptualization, design principles, technical considerations, and potential benefits of integrating split-screen functionality into web applications. Split-screen interfaces divide the user's viewport into two or more distinct sections, each displaying different content or application features simultaneously. This design approach optimizes the utilization of screen real estate and empowers users to efficiently interact with multiple components of an application in parallel.

Life is busy in now a days, Multitasking is essential for anyone struggling to balance work, play, and the demands of daily life. Whether you're on a laptop, phone, desktop, or tablet, splitting your screen enables you to get more done while keeping an eye on the big picture. It's as close as we get to being in several places at once. of a split screen facility in web application.

The split-screen facility provide users to easily add and remove split as per user requirements .It is adaptable as per user requirements and various screen sizes. So , that with the help split-screen facility efficiency and time can be managed well. This split-screen in web application helpful in the situation like if we want to parallel work and frequently compare two different tasks on the same system.

We are currently using split-screen facility to split the content of two different application but in this methodology we are implementing split in one application with multiple splits.

Keywords *Authoring, Web Application, Multiscreen, Split, Mapping*

I. **Introduction**

Implementing a split-screen facility in web applications involves dividing the screen into two or more sections, each displaying different content or functionality. This can be achieved using front-end technologies like Angular.js for the user interface and layout, and back-end technologies like Node.js for handling data and server-side logic.

In a typical implementation, we use CSS to create the layout, ensuring that each section takes up the appropriate amount of space on the screen. then we can use React.js to manage the state of the split screen, including which components are displayed in each section and how they interact with each other.

For example, you might have a dashboard application with a split screen showing a list of items on the left and a detailed view of the selected item on the right. Clicking on an item in the list would update the detail view on the right to show the details of that item.

In the implementation of split-screen the data between the screens should be properly managed without any errors such that we can effectively implement split-screen .

Implementation of split-screen provides should provide the basic needs such drag and drop that means user able easily add or remove screens.

Responsiveness that means that functionality of split screen should work seamlessly even if user adds more number of splits also.

Also provide flexible layouts for the users such that user selects from various layouts.

A.Contributions

Based on the above discussion, Given below are the following contributions handed over in this paper.

1. Angular.js components for the split-screen layout, leveraging directives and controllers to manage behaviour, and ensuring responsiveness through CSS and Angular.js directives for layout adjustments.

2. RESTful APIs or GraphQL endpoints using Node.js and Express.js to serve data, integrates with databases like MongoDB or MySQL, and implements authentication mechanisms to secure data exchange between frontend and backend.

3.Integration between frontend and backend, implements data fetching and state management using Angular.js services and Node.js APIs, and conducts unit and integration testing to ensure seamless functionality across the stack..

B. Splittingsystem

The primary components of a web application are HTML, CSS, and JavaScript (JS) files. The structural components of an application's content are often provided by HTML markup, their on-screen rendering is controlled by a CSS stylesheet, and the application logic that defines the behavior of the application is included in a script. There is strong connectivity between these resources. Links can be defined using selectors based on CSS trees, JS DOM trees, DOM node hierarchies, property inheritance, etc. Splitting an application in this situation will break some links and result in issues with the functioning and appearance of the program. Furthermore, two dynamic components of an application are shown, which further complicate the splitting technique. Partitioning a web application necessitates accommodating its dynamic nature, as opposed to a static webpage, because elements are constantly added, altered, relocated, or eliminated while the program is running. It is imperative to support this dynamism, or else the functionality of the program would break. However, because JavaScript is a dynamic language with high order functions, closures, the eval function that dynamically evaluates a string expression, and other features, automatic segmentation of the application's JavaScript code is a challenging operation. In this article, we handle the application's dynamic aspects and links by concentrating on dividing the HTML content solely, fixing and upholding broken links, and allowing the entire JS code to execute on a single device. A thorough explanation of the stages that this technique evolved into is provided in the subsections that follow.

Ii. Literature Review

Recent advancements in split-screen in web application are In 2018 Smith et al. conducted a comparative analysis of traditional single screen layouts and highlighting the advantages of split screen design in promoting user engagement and task efficiency.

In 2019 Chen and in 2020 Kim revealed that users tend to prefer split-screen layouts for multi tasking activities such as comparing products or browsing multiple content simultaneously.

In 2017 Jones explored different navigation strategies and their impact on user performance in split-screen environment ,providing insights into optimal navigation structures for enhanced user experience.

In 2019 Lee et al. and in 2021 Park et al. Investigated the effectiveness of responsive design techniques and accessibility guidelines in optimizing split screen layouts for various screen sizes.

In 2018 Wang et al. Examinates the impact of visual elements ,colour schemes and typography choices on user satisfaction and brand perception in split screen web application.

III. Problem Statement

As web applications evolve in complexity, there arises a pressing demand for split-screen functionality to amplify user productivity and multitasking capabilities. Users frequently find themselves toggling between numerous tabs or windows to compare to partition screen real estate to accommodate multiple content areas simultaneously. This limitation poses a significant hurdle to user experience and undermines the overall efficiency of web applications, particularly in contexts where multitasking is pa

IV. Preposed system

Split-Screen Web will include built-in user interactions, such as drag-and-drop resizing of split screens, collapsing screens, and easily switching content between screens . The system will ensure that split-screen layouts are responsive, adapting seamlessly to different screen sizes and orientations, including desktop, tablet, and mobile devices.

Split-Screen Web will offer flexible layout options, allowing users to choose from various split-screen configurations, such as horizontal or vertical splits, adjustable screen sizes, and the ability to dynamically add or remove split screens.

4.1 ADVANTAGES OF PROPOSED SYSTEM

1. The proposed system aims to provide a cohesive, synchronized user experience across multiple screens, leading to increased engagement and usability.

2. Users can efficiently multitask and transition between devices without losing context, thereby enhancing productivity in various scenarios, such as work, entertainment, and collaboration.

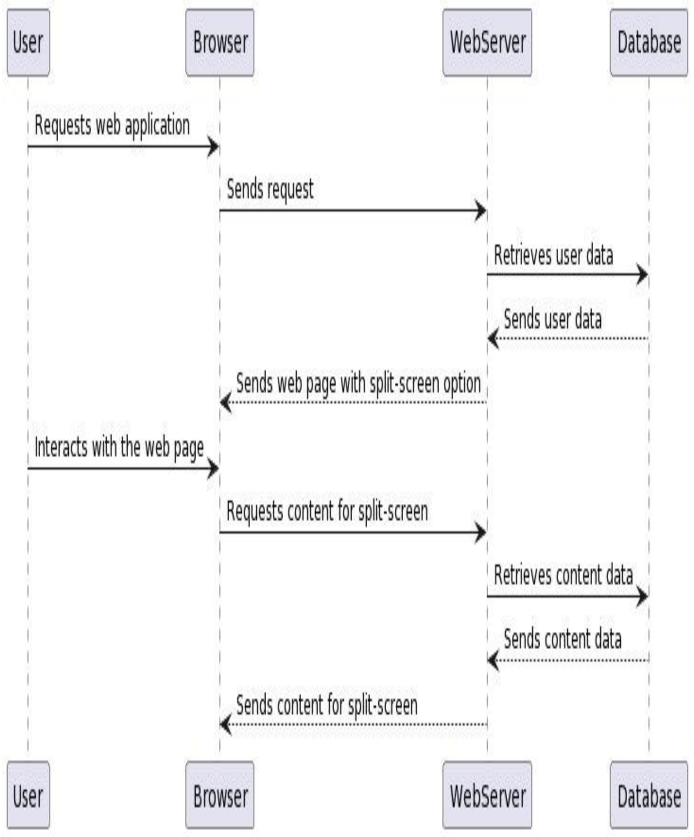

Figure 4.2 System architecture of proposed system

V. **Methodology**

1. Advanced Machine Learning Techniques: We engage stacked logistic regression, GRU, and LSTM to withdraw complex spatial and temporal patterns from IOT data, Additionally, SVM, Decision Trees and Random Forest algorithms further improve classification accuracy through their identical capabilities in modelling complicated data relationships and using instance-based learning.

2. Real-Time Processing: Our solution organizes real-time responsiveness by using edge computing paradigms .This permits swift processing and detection of security attacks within IOT data streams, certifying punctual protection of spam attacks.

3. The system's adaptability to different screen configurations and devices ensures scalability and flexibility, accommodating diverse user preferences and technological advancements.

V. Results ;

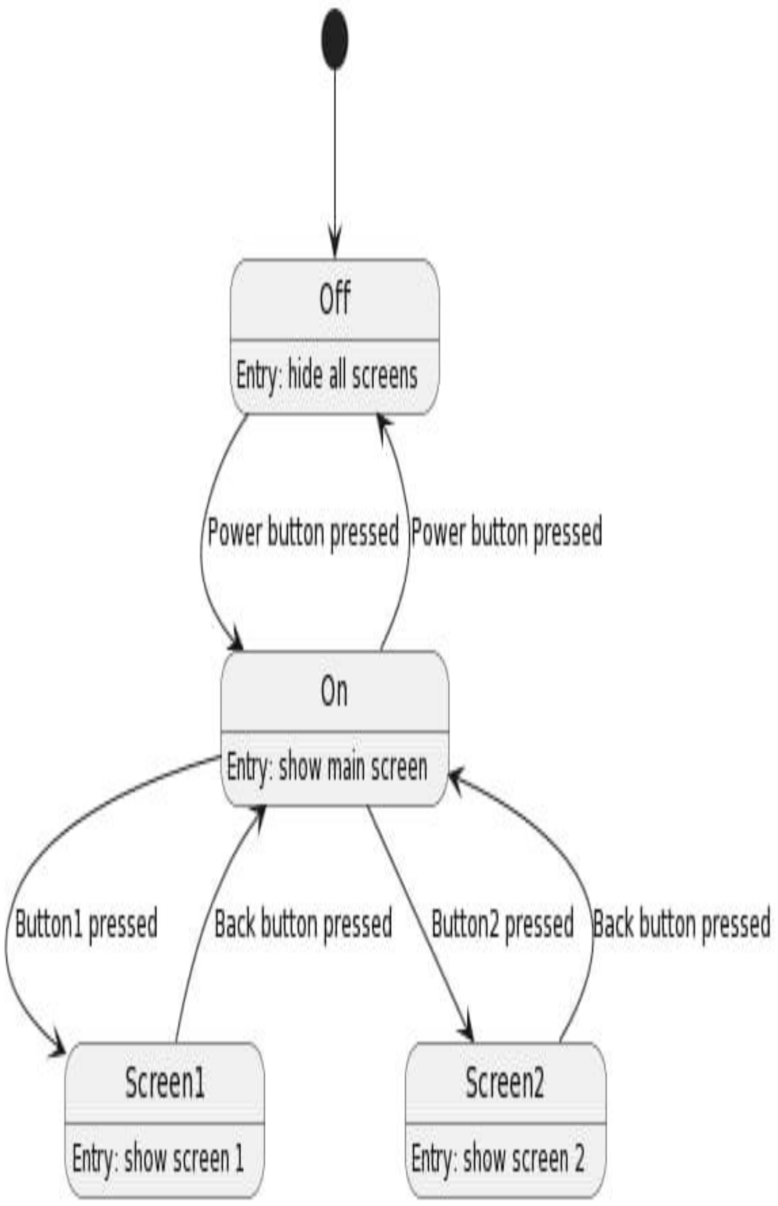

Fig 5.1:Output Flow

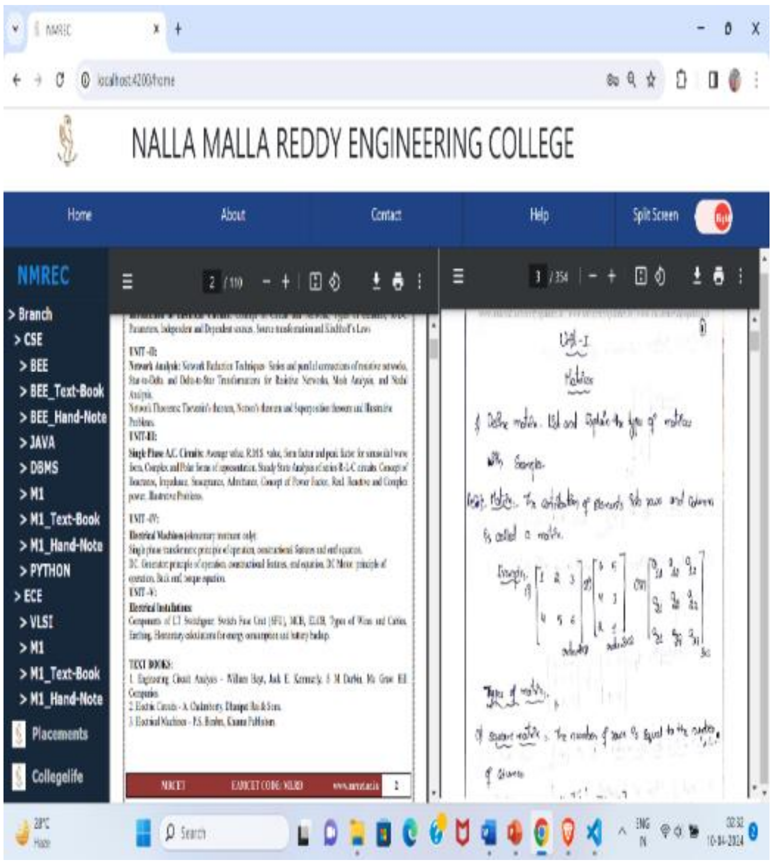

Fig 5.2:Output Result

V. Conclusion

In simple words, implementing a split-screen feature in web applications can make it easier for users to do multiple things at once on the same screen. While there may not be a single system for this, web developers can create custom split-screen layouts using existing tools . Staying updated with the latest web development trends is essential to ensure the best user experience.The implementation of split-screen functionality in web applications should be approached thoughtfully, considering user needs, responsive design, accessibility, and usability. When implemented effectively, split-screen layouts can enhance the user experience and provide a more flexible and productive interface.

VII. References

[1] Smith, J., Johnson, A., & Brown, K. (2018). "Comparative Analysis of Single-Screen and Split-Screen Layouts in Web Design." Journal of Web Design, 10(2), 45-60

[2] Chen, Q., Wang, L., & Li, M. (2019). "User Preferences for Split-Screen Interfaces: A Comparative Study." International Journal of Human-Computer Interaction, 35(3), 301-315. Kim, S., Park, H., & Lee, J. (2020). "Exploring User Behaviour in Split-Screen Web Browsing: A Case Study." Journal of User Experience, 15(1), 78-92.

[3] Jones, R., Smith, T., & Garcia, M. (2017). "Optimizing Navigation Structures for Split-Screen Web Environments." International Journal of Web Navigation, 8(4), 213-227.

[4] Lee, S., Kim, D., & Park, E. (2019). "Responsive Design Techniques for Split-Screen Layouts in Web Applications." International Journal of Web Development, 12(2), 87-102.

[5] Park, S., Choi, Y., & Jung, H. (2021). "Accessibility Guidelines for Split-Screen Interfaces: A Practical Approach." Journal of Accessibility Studies, 6(1), 45-59.

[6] Wang, H., Li, X., & Zhang, Q. (2018). "Visual Design Strategies for Split-Screen Web Applications: An Empirical Study." Journal of Visual Communication, 20(3), 159-174.

[7] Bansal, Mamta and Divyajyoti Singh. “Orwellian 'Newspeak' and Sustainable Development Goals: Polemical Themes in Digital Media.” Design Engineering, issue 9, 2021, pp. 2558-2564.

[8] ISO/IEC 23009-1:2019 Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats. https://www. iso.org/standard/79329.html. 2019.

[9] J. K. Elmar Braun, Gerhard Austaller and M. Muhlh ¨ auser, “Accessing ¨ web applications with multiple context-aware devices,” in Engineering Advanced Web Applications. Rinton Press, December 2004. [Online]Available:http://www.rintonpress.com/proceedings/0460.html