EMBRIONE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PES UNIVERSITY

NATIONAL LEVEL HACKATHON

KODiKON -4.0

NETWORKING – HACKATHON – MENTORSHIP

RenderTest

Streamline Your UI Testing with Automated Component Rendering

DEV TOOLS

KODiKON -4.0

TEAM DETAILS

TEAM NAME - R3ND3R

Team Member 1 Name:Kshitij Koushik Kota Phone Number:8073000306	Stream (ECE, CSE etc):CSE Email:kshitijkota22@gmail.com	College Name:PESU Campus Name: Ring Road
Team Member 2 Name:Sampriti Saha Phone Number:9620073788	Stream (ECE, CSE etc): CSE Email:sampritisaha@gmail.com	College Name:PESU Campus Name:Ring Road
Team Member 3 Name:Pranav Hemanth Phone Number:7483100279	Stream (ECE, CSE etc): CSE Email:pranavh2004@gmail.com	College Name:PESU Campus Name:Ring Road
Team Member 4 Name: Nishant Holla Phone Number: 6364354505	Stream (ECE, CSE etc): CSE Email: nishantholla19@gmail.com	College Name: PESU Campus Name: Ring Road

PROBLEM STATEMENT

DEV TOOLS

(Automated Frontend Component Tester)

Define:

As web applications become more complex, ensuring consistent and error-free user interfaces (UIs) is challenging. Dynamically rendering UI components for accurate previews is essential and manually creating ,maintaining test cases for interactions like clicks and forms is time-consuming and difficult to scale.

Measure:

Manual UI testing can take up to **30%** of a developer's time and often misses regressions due to outdated tests, leading to undetected bugs, post-release fixes, and decreased user satisfaction.

Analyze:

The problem lies in the manual effort needed to render components and update tests with each UI change. This slows development and increases bugs. Automating rendering and testing ensures reliable outcomes, faster iterations, and fewer errors.

SOLUTION

Detailed Solution Description:

The **Automated Frontend Component Tester** streamlines the development process by automating the rendering and previewing of React components, eliminating the need to manually call components and set up live servers for previews. The tool dynamically scans the component tree, automatically rendering individual components and their variations (e.g., with different props or states) in an integrated **live preview dashboard**. This allows developers to **instantly visualize UI components** without needing to run the full application.

In addition to rendering, the tool generates **automated test cases** for common interactions such as **clicks, form validation, and modal visibility**. It continuously monitors the UI for changes—such as updated button labels, newly added form fields, or modified props—and updates the relevant tests in real-time to ensure that tests remain **in sync** with the UI.

By combining **automated component previews with smart test generation**, the tool enhances productivity, reduces manual work, and ensures higher code quality, enabling seamless collaboration between developers and designers.

SOLUTION

Why the Solution Works:

Automated Rendering & Preview:
The tool eliminates the need to manually invoke and preview React components by automatically rendering them in an integrated live dashboard. This reduces setup time and ensures developers can instantly visualize components with various props, states, and interactions.

Real-Time Testing & Sync:
In addition to rendering, the tool automatically generates and updates test cases based on component interactions. This ensures that every component, from buttons to complex forms, is covered without manual effort, keeping tests aligned with the latest UI changes.

Consistent Component Behavior:
Automated tests ensure uniformity and prevent missing critical user flows across all components. As UI components evolve, the tool continuously monitors for modifications and updates tests without developer intervention, reducing the chance of regression bugs.

Seamless Framework Integration:
With built-in support for multiple frameworks, the tool interacts directly with component props and states, making it highly adaptable and relevant to modern frontend development workflows.

SOLUTION

Target Audience:

1. **Frontend Developers:**
   Automates component rendering and previews, saving time and allowing focus on feature development. Reduces the effort of maintaining tests.
2. **QA Engineers:**
   Auto-generates test cases, ensuring full coverage without manual effort. Speeds up the QA process.
3. **Designers & UI/UX Teams:**
   Provides live previews to interact with components, making collaboration easy and enabling real-time feedback.
4. **Agile Teams:**
   Keeps tests in sync with UI changes and automates testing in CI/CD pipelines for quick, high-quality releases.
5. **Organizations with Large Libraries:**
   Helps manage large, evolving component libraries with automated previews and real-time regression detection.

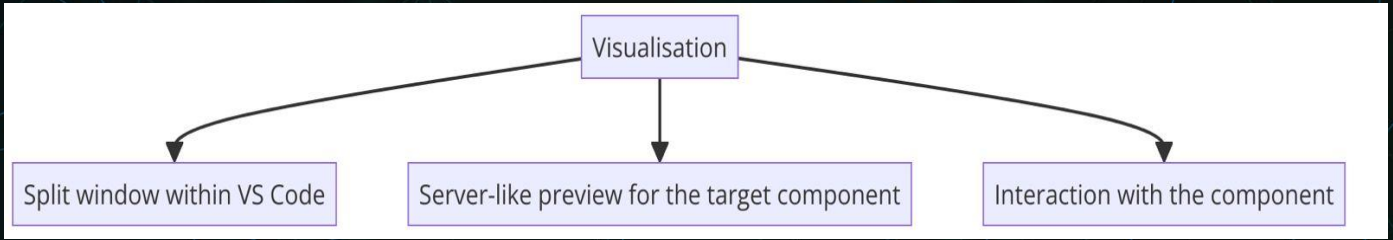
SOLUTION

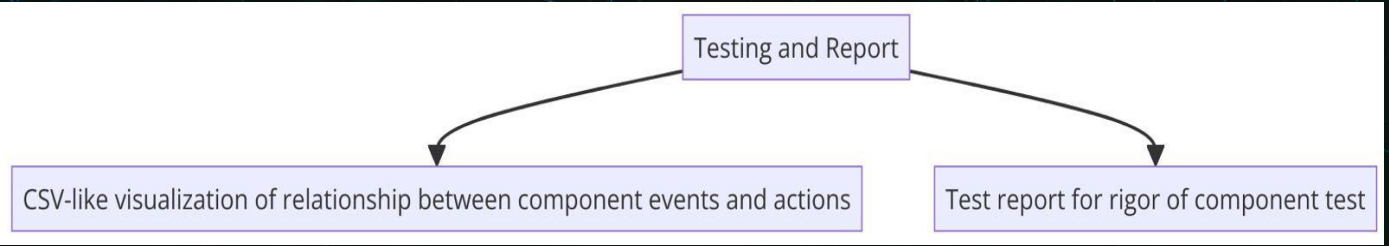
Expected Results:

1. **Increased Productivity:**
   Developers will spend less time writing and maintaining tests, leading to faster development cycles and more focus on building new features.
2. **Improved User Experience:**
   With more reliable and automated testing, applications are less likely to encounter UI-related bugs, resulting in a better user experience.
3. **Higher Test Coverage:**
   Automated generation of test cases ensures that all UI elements are tested, reducing the likelihood of undetected bugs.
4. **Reduced Manual Testing:**
   The tool minimizes reliance on manual UI testing, lowering the risk of human error and speeding up the QA process.

SOLUTION

Flow Chart:





TECH STACK

- **FRONTEND:**
  - **Webview API:** VS Code extensions use Webviews to display custom HTML content.
  - **React:** Supported front-end framework for visualization and testing.
  - **CSS:** Styling test bench and components.
- **BACKEND:**
  - **Node JS:** Backend to handle file system interactions and communication between the frontend and VS Code APIs.
  - **Webpack:** For dynamically compiling React components.
- **API's/SERVICES:**
  - **Express JS:** Run tests or visualizations in a separate environment, by exposing API to communicate between the extension and a local testing server.

Domains covered: Frontend Development, Backend Development, Tooling and IDE Integration, Software Testing and Quality Assurance, DevOps/Continuous Integration, API Services (RESTful APIs)

BUSINESS SCOPE

Value to Organizations:

- **Increased Efficiency:** Automates rendering and testing, speeding up development.
- **Enhanced Quality:** Ensures comprehensive test coverage, reducing production bugs.
- **Better Collaboration:** Live previews promote communication among developers, QA, and designers.
- **Adaptable Solutions:** Suitable for various project sizes across teams.
- **Cost Savings:** Lowers manual testing efforts, reallocating resources effectively.

Revenue Generation Potential:

- **SaaS Model:** Recurring revenue through tiered subscription plans.
- **Marketplace for Add-ons:** Income from premium plugins and features developed by third parties.
- **Freemium Model:** Basic features available for free, with advanced functionalities offered as paid upgrades.
- **Enterprise Licensing:** Custom solutions for larger teams with one-time or annual fees.