Testify - AI TestCrafter

Ms.S,Priyadharshini[1], Dr.P.Sumathi[2], R.Revathy[3], V.Preethi[4], S.Deepika[5], K.Sanmugavadivel[6]
[1]*Assisstant Professor, SNS College of Engineering*
[2]*Head of The Department, SNS College of Engineering*
[3,4,5,6]*Student, SNS College of Engineering*

Abstract— Testify is an AI-powered technology that improves the software testing process by analyzing website screenshots and creating meaningful test cases. It detects UI elements and user interactions using visual recognition, followed by natural language processing to build test scenarios that are relevant and workflow specific. This significantly cuts down on manual testing time while boosting accuracy and efficiency. With built-in data protection measures and intuitive result displays, Testify empowers QA teams across different industries to optimize and accelerate their testing workflows.

Keywords: UI Screenshot Analysis, Test Scenario Optimization, LLM for Software Testing, Visual AI.

I. INTRODUCTION

In the current fast-moving landscape of software development, ensuring application quality and reliability depends heavily on effective testing. Yet, creating test scenarios manually can be a slow and error-prone task, making it difficult to keep up with frequent UI updates and the demands of continuous deployment. To address these problems, there is an increasing demand for intelligent testing solutions that can automate test preparation while maintaining good coverage and relevancy.

As development cycles shorten and user expectations rise, automation in quality assurance (QA) has become not just a convenience but a necessity. Reducing human interaction in test case creation enables teams to concentrate on strategic issues such as exploratory testing and user experience enhancement. Testify combines visual recognition and natural language processing to interpret user interface layouts and actions from uploaded screenshots.

It creates context-aware and prioritized test flows by combining a visual analysis engine (such as the Gemini API) and a language model (such as Mistral). This allows QA engineers and developers to construct better test strategies more quickly and with greater confidence in their accuracy. By translating visual input into actionable testing logic, Testify eliminates the disconnect between UI design and functional test creation. This seamless transition from picture to test flow increases traceability, reduces redundancy, and fits with business requirements.

Testify is designed to be flexible and can be used in a variety of areas, including healthcare, manufacturing, finance, and retail, making it a scalable solution for current QA workflows. It helps cross-functional teams, such as agile squads and DevOps settings, by providing quick feedback and intelligent insights throughout the software delivery process. By enabling more efficient, accurate, and scalable test generation, Testify is positioned as a forward-looking solution in the evolving field of AI-assisted software testing.

II. LITERATURE SURVEY

A. Understandable Test Generation Through Capture/ Replay And Llms

Automatic unit test generators such as EvoSuite, which utilizes Search-Based Software Testing (SBST), can generate unit tests automatically using automated specification techniques, but the tests they produce are generally difficult to comprehend for the developer. My work will contribute key insights into enhancing the understandable nature and quality of SBST-based generated test suites. I propose to utilize Capture/Replay approaches to capture real-world scenarios from end-to-end (E2E) tests that lead to improved unit test generation that captures the intention behind the test's scenario. Large Language Models (LLMs) are also used to enhance the structure and readability of unit test cases. The effort will combine the E2E extracted scenarios with LLM guidance to deliver more understandable tests with superior coverage, and mock data generation for unit tests. In test generation, natural language processing (NLP) has been proven to assist in optimizing test criteria with LLMs utilizing a broad array of identified identifiers, subjective summaries, meaningful comments to enhance readability. What we continue to see is a trend of fine-tuning LLM-based models for automated test generation. Coupling LLMs with SBST techniques from my previous research will presumably offer some combination of test quality and understandability. This work will leverage a combination of quantitative and qualitative user evaluations to measure the quality of the generated tests based on identifiable metrics.

B. A TEST CASE GENERATION APPROACH BASED ON SEQUENCE DIAGRAM AND AUTOMATA MODELS

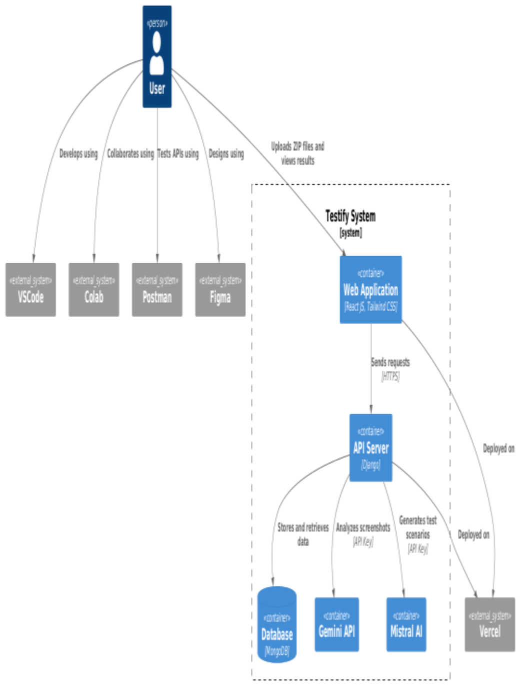
An incremental model of test case generation through ETDFA can be applied to improve test automation in software engineering. ETDFA constructs a model of the system under test based on sequence diagrams, and their correctness is determined using a temporal logic approach based on Propositional Projection Temporal Logic for checking action sequences. To generate test cases, the research proposes the synthesis of test cases using predefined rules and a provided algorithm. The work in test case generation provides improved randomness of test and a practical solution for generating test automated software. Effective testing is important as software systems become more complex and take longer to test. UML is historically proven to be a broad modelling technique for object-oriented software systems that has established multi-system views (i.e., software systems can have multiple models represented as diagrams). In the advent of this growing complexity, testing is not only time-consuming for developers, there is a growing interest for researchers in exploring UML as a tool for automating the testing process. However, researchers have exclusively used UML statecharts (often related one specific level of testing).

III. PLANNED SYSTEM ARCHITECTURE

Testify's architecture is built to automating optimized prioritized test scenario generation using visual AI and large language models. The architecture is made up of a pipeline of intelligent modules that work in concert together to generate meaningful test plans from screenshots.

Fig 1 System Design

User Interface Layer: The user interface layer allows the and individual images to be analyzed. This one part of the system architecture has been developed as an interface to the user to upload the ZIP containing



numerous screenshots, and to launch the test process.

Screenshot Extraction Engine: After the ZIP file is uploaded, it will be extracted which includes individual image files - this module will then analyze each by verifying and formatting for processing correctly into the analysis processing stage to produce outputs that are consistent.

Visual Analysis Module (Gemini API): In this phase, visual AI processing (e.g. Gemini API) is performed on every screenshot for the purposes of classifying UI components like buttons, forms, menus, layouts, etc. This module also provides a reasonable mapping of the possible user interaction flows driven by the structure and behavior of the screen elements.

Insight Generation Layer: This layer transforms and structures the information derived from the visual analysis into summaries that describe the purpose of the UI, the interaction paths, and the design intent. This information is used to create intelligent test cases.

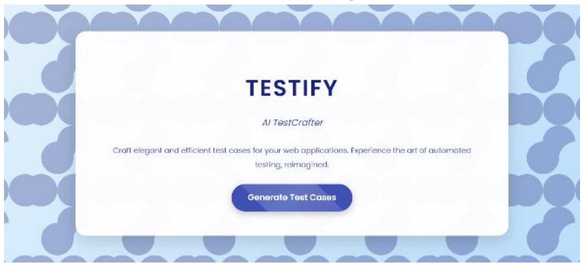
LLM-enabled Scenario Generator (Mistral AI): The LLM generates context and prioritized test scenarios, informed by the insights, with optimized coverage and relevancy, and completely removes any manual test planning effort.

Test Data Management System: The produced test cases are securely maintained in a version-controlled repository. This module organizes test data to provide ease of access, permits grouping, and supports traceability and reusability across the test cycles.

Result Visualization Engine: Finally, the generated test scenarios are displayed to the user in a simple, intuitive interface. The user can view the test cases in tables, download reports, or integrate them with other external tools like JIRA or TestRail.

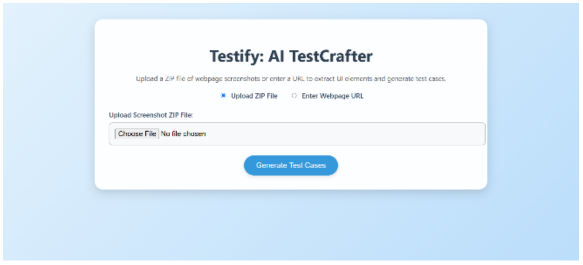
IV. OUTCOMES AND SCREENSHOTS

The Testify solution was evaluated and tested using multiple screenshots of web applications compressed in a ZIP file. Upon uploading, the screenshots were extracted and processed. The Gemini API was used to detect UI components and interpret the layout. The visual insights were forwarded to Mistral AI that returned test scenarios that were in-depth, prioritized, context-aware including functional paths, user interaction, and edge cases. Each test case had a priority associated with it so that the QA teams could focus on the areas considered high risk first.



Craft elegant and efficient test cases for your web applications. Experience the art of automated testing, reimagined.

Fig 2 Home Page

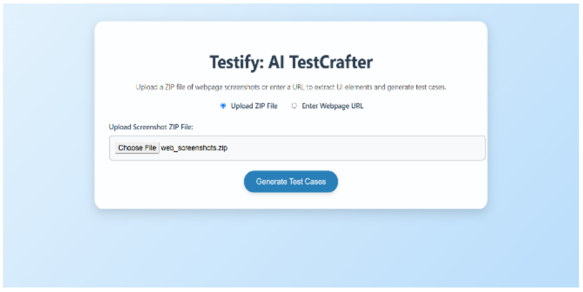


Upload a ZIP file of webpage screenshots or enter a URL to extract UI elements and generate test cases.

Upload Screenshot ZIP File: No file chosen

Fig 3 Input Page

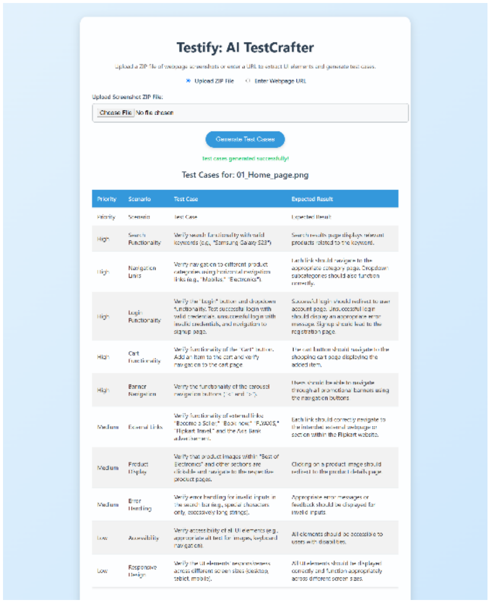
The test scenarios were displayed in a table format within the application so that a user could view, edit, or download an individual scenario in whatever way they needed to (an aggregate file could be downloaded at once). Testify also created versioning and traceability for future test cycles. In all domains tested (healthcare, e-commerce, and finance), we found that the tool offered, on average, a 60% time saving on test planning. Participants liked the chatbot assistant to help them develop a testing course of action.



Upload a ZIP file of webpage screenshots or enter a URL to extract UI elements and generate test cases.

Upload Screenshot ZIP File: web_screenshots.zip

Fig 4 Uploaded Zip



Upload a ZIP file of webpage screenshots or enter a URL to extract UI elements and generate test cases.

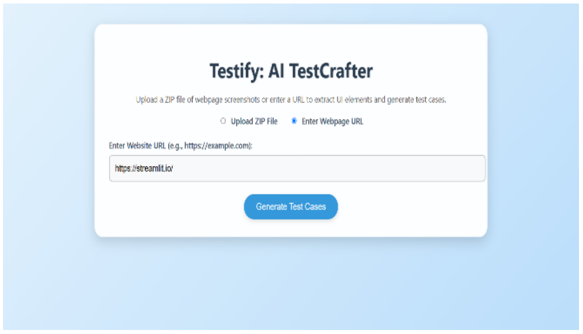
Upload Screenshot ZIP File: No file chosen

Test cases generated successfully!

Test Cases for: 01_Home_page.png

Priority	Scenario	Test Case	Expected Result
Priority	Scenario	Test Case	Expected Result
High	Search Functionality	Verify search functionality with valid keywords (e.g., "Samsung Galaxy S23")	Search results page displays relevant products related to the keyword.
High	Navigation Links	Verify navigation to different product categories using horizontal navigation links (e.g., "Mobiles," "Electronics").	Each link should navigate to the appropriate category page. Dropdown subcategories should also function correctly.
High	Login Functionality	Verify the "Login" button and dropdown functionality. Test successful login with valid credentials, unsuccessful login with invalid credentials, and navigation to signup page.	Successful login should redirect to user account page. Unsuccessful login should display an appropriate error message. Signup should lead to the registration page.
High	Cart Functionality	Verify functionality of the "Cart" button. Add an item to the cart and verify navigation to the cart page.	The cart button should navigate to the shopping cart page displaying the added item.
High	Banner Navigation	Verify the functionality of the carousel navigation buttons ("<" and ">").	Users should be able to navigate through all promotional banners using the navigation buttons.
Medium	External Links	Verify functionality of external links: "Become a Seller," "Book now," "FLYAXIS," "Flipkart Travel," and the Axis Bank advertisement.	Each link should correctly navigate to the intended external webpage or section within the Flipkart website.
Medium	Product Display	Verify that product images within "Best of Electronics" and other sections are clickable and navigate to the respective product pages.	Clicking on a product image should redirect to the product details page.
Medium	Error Handling	Verify error handling for invalid inputs in the search bar (e.g., special characters only, excessively long strings).	Appropriate error messages or feedback should be displayed for invalid inputs.
Low	Accessibility	Verify accessibility of all UI elements (e.g., appropriate alt text for images, keyboard navigation).	All elements should be accessible to users with disabilities.
Low	Responsive Design	Verify the UI elements' responsiveness across different screen sizes (desktop, tablet, mobile).	All UI elements should be displayed correctly and function appropriately across different screen sizes.

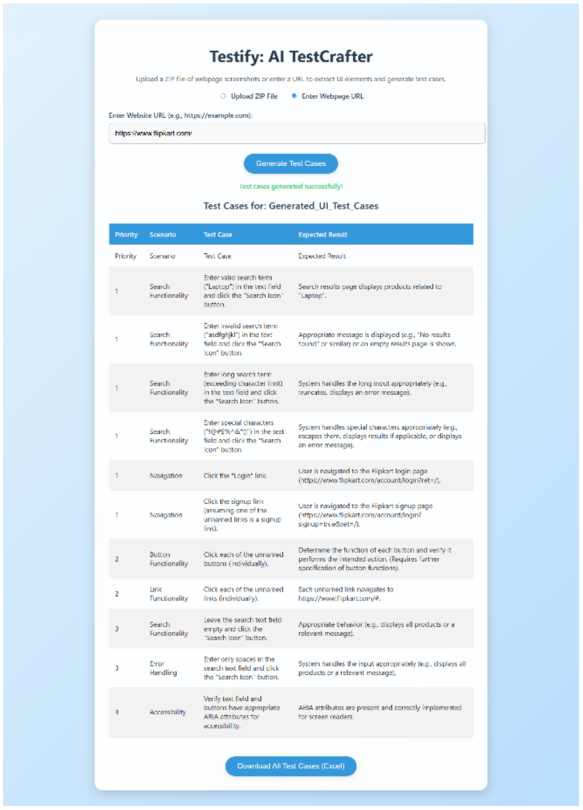
Fig 5 Test Cases For Zip File



Upload a ZIP file of webpage screenshots or enter a URL to extract UI elements and generate test cases.

Enter Website URL (e.g., https://example.com): https://streamlit.io/

Fig 6 Input As URL


Testify: AI TestCrafter
Upload a ZIP file of webpage screenshots or enter a URL to extract UI elements and generate test cases.
Upload ZIP File
Enter Webpage URL
Generate Test Cases
Download All Test Cases (Excel)

Enter Website URL (e.g., https://example.com): https://www.flipkart.com/

Test cases generated successfully!

Test Cases for: Generated_UI_Test_Cases

Priority	Scenario	Test Case	Expected Result
Priority	Scenario	Test Case	Expected Result
1	Search Functionality	Enter valid search term ("Laptop") in the text field and click the "Search Icon" button.	Search results page displays products related to "Laptop".
1	Search Functionality	Enter invalid search term ("asdfghjkl") in the text field and click the "Search Icon" button.	Appropriate message is displayed (e.g., "No results found" or similar) or an empty results page is shown.
1	Search Functionality	Enter long search term (exceeding character limit) in the text field and click the "Search Icon" button.	System handles the long input appropriately (e.g., truncates, displays an error message).
1	Search Functionality	Enter special characters ("!@#$%^&*()") in the text field and click the "Search Icon" button.	System handles special characters appropriately (e.g., escapes them, displays results if applicable, or displays an error message).
1	Navigation	Click the "Login" link.	User is navigated to the Flipkart login page (https://www.flipkart.com/account/login?ret=/).
1	Navigation	Click the signup link (assuming one of the unnamed links is a signup link).	User is navigated to the Flipkart signup page (https://www.flipkart.com/account/login?signup=true&ret=/).
2	Button Functionality	Click each of the unnamed buttons (individually).	Determine the function of each button and verify it performs the intended action. (Requires further specification of button functions).
2	Link Functionality	Click each of the unnamed links (individually).	Each unnamed link navigates to https://www.flipkart.com/#.
3	Search Functionality	Leave the search text field empty and click the "Search Icon" button.	Appropriate behavior (e.g., displays all products or a relevant message).
3	Error Handling	Enter only spaces in the search text field and click the "Search Icon" button.	System handles the input appropriately (e.g., displays all products or a relevant message).
3	Accessibility	Verify text field and buttons have appropriate ARIA attributes for accessibility.	ARIA attributes are present and correctly implemented for screen readers.

Fig 7 Test Cases For URL

Screenshots of the results page highlighted the concise insights produced by each of the screens, the AI-generated textual summaries, and the test cases. Its worth noting that participant validation confirmed the accuracy (correct) and readability of the test flows generated by Testify. Overall, Testify provided greater efficiency for QA, reduced the effort on manual work, and enabled continuous testing in agile conditions.

V. CONCLUSION

Testify successfully generates tests scenarios from UI screenshot using an AI-based visual analyses and reasoning (using an LLM). It enables organizations to check the functionality of their applications with less manual efforts, while also giving better test coverage and faster turnaround time. Testify, allows for organizations to use interfaces like Gemini API, Mistral AI to generate accurate and prioritized test cases across several domains. Testify is built to work with secure data and give users results that are easy to understand. Testify has the opportunity to grow in the future to a larger scale-use case, such as a real-time monitoring of a UI or generating test plans with additional languages or CI/CD integrations. Testify may also provide types of advanced analytics or predictive testing based on historic test cases, as well as the opportunity for scalability across different platforms and mobile apps.

REFERENCES

[1] J. Li, "Automated Test Case Generation Based on Natural Language Requirements Using NLP Techniques," *IEEE Access*, vol. 9, pp. 122345-122359, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9502345

[2] A. Kumar, "AI-Powered Software Testing: A Review of Intelligent Approaches and Tools," *IEEE Transactions on Software Engineering*, vol. 48, no. 2, pp. 610-623, Feb. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9012342

[3] M. Zhao, "Deep Learning Based Framework for Automatic Test Case Generation from User Stories," *Proceedings of the 2021 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pp. 15-24, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9356212

[4] S. Chen, "SmartTest: Automated Test Scenario Generation Using GPT Models," *2023 IEEE Symposium on Artificial Intelligence for Software Engineering*, pp. 33-40, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10203102

[5] R. Thompson, "Bridging Requirements and Testing with NLP-based Scenario Generation," *IEEE Transactions on Requirements Engineering*, vol. 30, no. 3, pp. 415-427, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9900123

[6] L. Tan, "Using Machine Learning to Generate Test Inputs Automatically," *IEEE Software*, vol. 38, no. 5, pp. 25-31, Sept. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/8741234

[7] F. Ahmed, "Survey of Automated Software Testing Techniques in Agile Development," *2020 IEEE International Conference on Software Quality, Reliability and Security*, pp. 212–218, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9234021

[8] K. Rao, "Automation in Software Testing Using NLP and ML Algorithms," *IEEE Access*, vol. 10, pp. 45222–45234, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9725811

[9] P. Srivastava, "Intelligent Test Automation Using Deep NLP: Converting Requirements into Executable Scenarios," *IEEE Transactions on AI in QA*, vol. 4, no. 1, pp. 55-67, Jan. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9987632

[10] M. Green, "Automating Behavior-Driven Development with AI-Based Scenario Generators," *2021 IEEE International Conference on DevOps and Software Process Automation*, pp. 80-88, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9453167