MODEL BASED TESTING APPROACH FOR TOURISM APPLICATIONS : USE CASE SCENARIOS FOR BOOKING ACCOMMODATIONS

Efe Batur Giritli, Yekta Said Can[1], Alper Sen and Fatih Alagöz[2]
[1]R&D Center, Tatilsepeti, Istanbul, TURKEY

[2]Computer Engineering Dept., Bogazici University, Istanbul, Turkey

ABSTRACT

In the rapidly evolving landscape of the global tourism industry, efficient and reliable online accommodation booking systems are paramount. The primary objective of this paper is to explore the application of Model-Based Testing (MBT) in enhancing the efficiency and reliability of online accommodation booking systems within the global tourism industry. Our research developed and implemented a model-based testing framework that utilizes the GraphWalker tool to create detailed behavior models of tourism applications, specifically focusing on the booking accommodation process. Through the integration of GraphWalker with Selenium WebDriver, we successfully automated the testing process, ensuring comprehensive coverage of user scenarios from login to payment completion. Our results demonstrate the effectiveness of the MBT approach in identifying system vulnerabilities, improving test coverage, and providing a seamless testing experience, thereby contributing to the development of robust and user-friendly online booking platforms.

KEYWORDS

model based testing, graphwalker, selenium web driver, model based booking

1. INTRODUCTION

The Model-Based Testing (MBT) technique relies on utilizing models to generate test cases based on specific requirements [1]. In simpler terms, it involves comparing the software's behavior with the behaviors outlined in the model. The general semantics of MBT, as illustrated in Figure 1, reveal that the model serves as an abstraction of the real-world system, jointly created by software developers and testers. It functions as a blueprint for preparing abstract tests, which can then be derived by a software developer, a testing expert, or an automated tool system. Subsequently, these abstract tests can be executed to become executable tests. These executable tests are pivotal in carrying out dynamic executions and verifying the system's correctness or identifying software defects.

In certain instances, it is feasible to directly execute tests from the model, offering specific advantages [2]-[6]. The presence of abstract tests provides distinct benefits. For instance, ReCDroid [7] utilizes abstract test steps sourced from software bug reports to create tests. ReCDroid automates the process of transforming these abstract tests into executable ones. Another test generation tool, FARLEAD-Android [8], derives its executable tests from user-friendly test cases written in the Gherkin language which is a domain specific language [9]. The use of abstract tests in ReCDroid and FARLEAD Android enables users, even those without in-depth knowledge of source code or coding skills, to understand the scope of what is being tested.

David C. Wyld et al. (Eds): NLMLT, SEAU– 2024
pp. 57-64, 2024. - CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.142006

Furthermore, abstract testing empowers software developers and testers to extract environment-specific details from test cases, facilitating the concealment of tests on various devices and adaptability to different operating system configurations. Notably, abstract tests are more resourceefficient and demonstrate greater resilience to minor code modifications, making them a sustainable choice for testing endeavors.

Typically, software developers don't commonly construct models for Graphical User Interface (GUI) applications. Consequently, it might not be feasible to derive tests from the system model. This challenge is particularly pronounced in the realm of Android GUI applications, given the intense competition in the market. This competitiveness often discourages the allocation of additional time to GUI modeling. As a solution, automated test generation tools frequently acquire knowledge of the GUI model through dynamic execution.

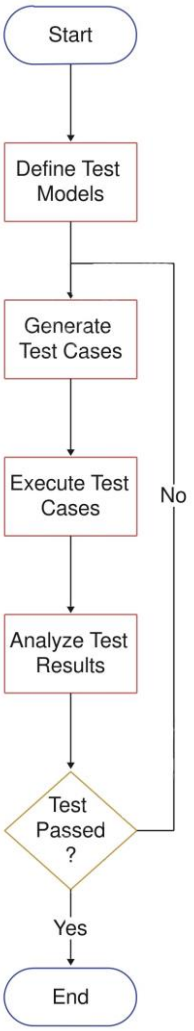

Fig. 1. Model Based Testing Flowchart.

At this point, Graphwalker serves as an invaluable modeling tool. It provides a visual representation of the application's behavior. This visual representation likely comes in the form of a graph or a diagram, making it easier for stakeholders, developers, and testers to understand how different components of the application interact and behave. By offering a visual representation of the application's behavior, Graphwalker makes it easier to identify potential

weaknesses and vulnerabilities. This means that stakeholders can visually inspect the model to pinpoint areas where the application may not be performing optimally or where security vulnerabilities might exist. It also plays a crucial role in the development and testing processes by providing insights that are difficult to obtain through other means.

Tourism is increasingly embracing technology and artificial intelligence (AI) with each passing day [10]. As the industry evolves, we're witnessing a remarkable shift towards the integration of cutting-edge technology. In line with this trend, there's a growing momentum in the development of modelbased booking systems designed to enhance the overall tourism experience as in the other sectors [11]. In this study, we also tested this system by using model based testing approach with GraphWalker and provided the results.

2. BACKGROUND: GRAPHWALKER AND SELENIUM

The integration of GraphWalker and Selenium tools, driven by model-based testing, is a pivotal aspect of the testing process [12].

GraphWalker, an open-source model-based testing tool, offers three key features that significantly enhance the testing workflow. It encompasses an intuitive editor for creating and editing models. The tool facilitates model testing by generating test paths, making the models verifiable for users to assess their accuracy. GraphWalker is adaptable for online use via command line tools and enables offline test trace generation.

A GraphWalker model comprises two fundamental elements: edges and nodes. Edges represent actions and transitions, encompassing actions like API calls, button clicks, and time timeouts. The validity of your test depends on these actions, ensuring they lead to the desired state without constraints. Circular paths can be employed for validation, represented as assertions that check the correctness of values returned by API calls, confirming button clicks trigger dialog boxes, and verifying timeouts.

Test templates generated from the abstract models using GraphWalker can be seamlessly incorporated into the Selenium library. Selenium WebDriver, a cross-platform tool for configuring and controlling web browsers, acts as a versatile test framework. It provides a programming interface to perform actions on web elements, supporting multiple programming languages such as Java, C, PHP, and Python. Selenium WebDriver also offers built-in capabilities for generating test results. For comprehensive test reporting, users can employ third-party tools like GraphWalker, TestNG, TestNG for test management, and integrate it with frameworks like JUnit [13].

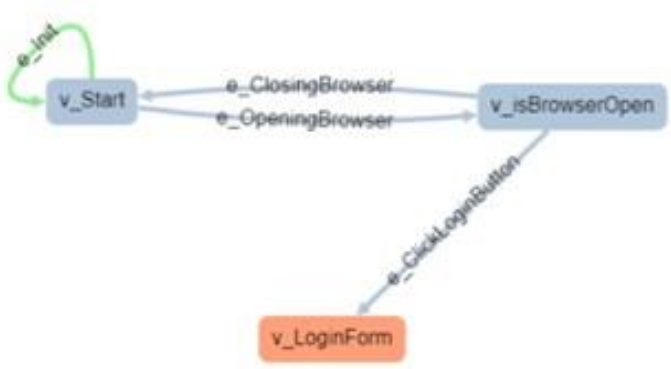

Fig. 2. Automated test case scenario for user login.

3. METHODOLOGY

We created a scenario for testing the model-based approach. In this scenario, the user is first tries to login. After that, the user lists hotels in Antalya (a touristic city in Turkey) and selects a hotel, provide credit card information and reserves the hotel.

3.1. Login

Objective: Ensure that users can successfully log in to the tourism hotel search website (see Figure 2).

Steps:

1. Launch the website and navigate to the login page.
2. Enter a valid username and password.
3. Click the “Login” or “Submit” button.
4. Verify that the login is successful by checking for a successful login message or a user dashboard page.

Expected Outcome: The test should confirm that a user can log in without any errors.

3.2. Hotel Search

Objective: Verify that users can search for hotels in Antalya, Turkey, after a successful login (see Figure 3).

Steps:

1. Once logged in, navigate to the hotel search page.
2. Select the destination “Antalya, Turkey.”
3. Configure search criteria such as check-in and check-outdates, number of guests, room type, etc.
4. Initiate the search by clicking the “Search” or “Find Hotels” button.

Expected Outcome: The test should confirm that the hotel search results are displayed without errors, and they are specific to Antalya, Turkey.

Fig. 3. Automated Test Scenario for hotel search in Antalya, Turkey.

3.3. Reservation

Objective: Test the process of selecting a hotel and initiating a reservation (see Figure 4). Steps: button.

1. Review the search results and select a hotel of your choice.
2. Navigate to the hotel's detail page.
3. On the hotel detail page, click the "Reserve" or "Book Now" button.
4. Enter guest information and any other required details for the reservation.

Expected Outcome: The test should confirm that a reservation can be initiated for the selected hotel without any issues.

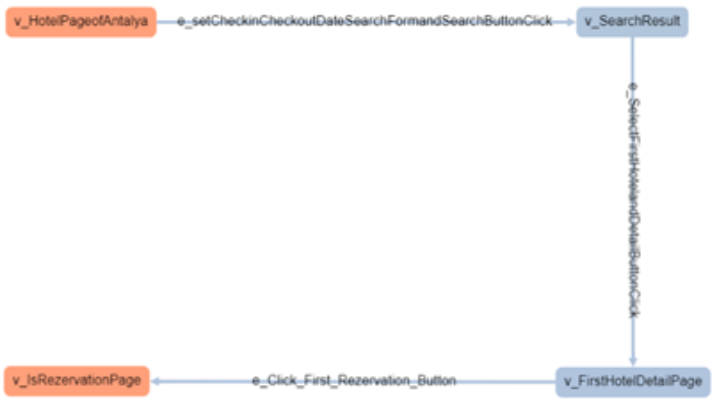

Fig. 4. Automated test scenario for selecting a hotel and initiating a reservation.

3.4. Payment

Objective: Ensure that the payment process can be completed successfully (see Figure 5).

Steps:

1. After initiating the reservation, the website should redirect you to the payment page.
2. On the payment page, enter test credit card information, including card number, \ expiration date, CVV, and billing address.
3. Complete the payment by clicking the "Pay" or "Confirm Reservation" button.
4. Verify that the payment is successful, and a confirmation message or page is displayed.

Expected Outcome: The test should confirm that the payment process works as expected, and the reservation is successfully completed.

Overall Expected Outcome: The entire automated test scenario should execute without any errors or issues. It should cover the entire flow from logging in to making a hotel reservation with a test credit card.

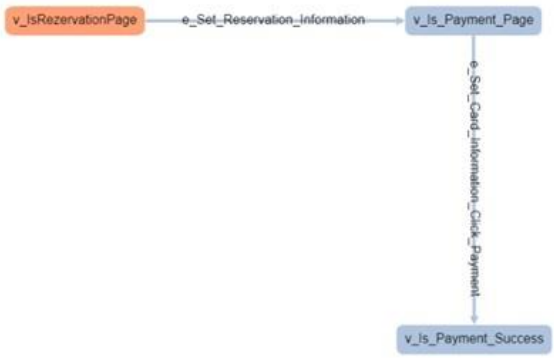

Fig. 5. Automated test scenario for the payment process.

4. DISCUSSION

Models for the whole hotel reservation process are prepared. Graphs with edges, vertex and nodes are created. After the models are prepared, the contents of the test scripts are filled in using the Selenium library. We wanted to measure the coverage of the tests [14]. Graphwalker then generates test traces from the models according to the desired coverage rate, e.g. random 100% button or edge coverage. These tests are also run with the filled scripts. Figure 6 shows the result of a test run on the generated models. The test results were obtained using random 100% vertex coverage (random (vertex _coverage(100)) and edge coverage (random(edge coverage(100)). The test was run from start to the end, and it automatically perform the tasks and finished the reservation and payment processes successfully.

```
Model and element to be called: HotelReservationFormSharedState.v_Is_Payment_Success
Running: v_Is_Payment_Success
{
  "totalFailedNumberOfModels": 0,
  "totalNotExecutedNumberOfModels": 0,
  "totalNumberOfUnvisitedVertices": 0,
  "verticesNotVisited": [],
  "totalNumberOfModels": 4,
  "totalCompletedNumberOfModels": 4,
  "totalNumberOfVisitedEdges": 10,
  "totalIncompleteNumberOfModels": 0,
  "edgesNotVisited": [
    {
      "modelName": "LoginSharedState",
      "edgeId": "bc237c60-4df2-11ed-a3f7-07114f577d8f",
      "edgeName": "e_ClosingBrowser"
    }
  ],
  "result": "ok",
  "vertexCoverage": 100,
  "totalNumberOfEdges": 11,
  "totalNumberOfVisitedVertices": 13,
  "edgeCoverage": 90,
  "totalNumberOfVertices": 13,
  "totalNumberOfUnvisitedEdges": 1
}
```

Fig. 6. The result of a test run on the generated models.

5. CONCLUSION

In the dynamic online accommodation booking, ensuring the robustness and functionality of booking platforms is of paramount importance. This paper has delved into the application of model-based test approaches, leveraging the power of Graphwalker alongside Selenium for the systematic evaluation of online accommodation booking systems.

Our methodology has showcased its efficacy in methodically designing, executing, and validating test scenarios. By developing graph models that diverse user journeys within the booking system, we help testers to comprehensively cover a wide spectrum of use cases and potential interactions. Graphwalker serves as a modeling tool, offering a visual representation of the application's behavior, thereby facilitating the identification of possible weaknesses and vulnerabilities.
Selenium, a well-established automation library, plays a pivotal role in the execution of the generated test scripts. This dual synergy of Graphwalker and Selenium not only simulates user interactions but also automates the entire process, minimizing the likelihood of human errors and vastly amplifying the efficiency of testing procedures.

The real-world application of this approach, particularly within the context of booking accommodations online, underscores its practicality and relevance. From the initial login to hotel search, reservation, and payment processes, this approach delivers a comprehensive framework for the exhaustive testing of the entire system. It ensures that users can effortlessly and without encountering errors, complete their booking transactions.

Moreover, the model-based approach significantly enhances test coverage by accommodating various pathways and user behaviors. This approach allows for the agile creation and modification of test scenarios, rendering it adaptable to the landscape of the application and helping continual enhancement of test coverage.

Furthermore, this methodology empowers the early detection of potential issues in the software development lifecycle. Testers, armed with test scripts derived from the model, are well-equipped to pinpoint areas of concern, leading to swift bug resolution and continuous improvements.
In summation, the model-based test approach, where Graphwalker and Selenium take center stage, furnishes a systematic and effective mechanism for ensuring the dependability and functionality of online accommodation booking systems. It improves test coverage, streamlines the testing process, and adds to the overall quality of the software.

By embracing this methodology, organizations can increase confidence in their users by providing a seamless and errorfree online booking experience. This, in turn, contributes to heightened customer satisfaction and cements the success of their online accommodation booking platforms. However, this work is not without limitations. The number of scenarios is limited. There are a lot more scenarios in booking accommodation process. In future works, we plan to add more scenarios and extend the test coverage in terms of scenarios for the booking accommodation process.

REFERENCES

[1] garousi, v., keles¸, a. b., balaman, y., guler, z.¨ o., arcuri, a. (2021).¨ model-based testing in practice: an experience report from the web applications domain. journal of systems and software, 180, 111032.
[2] d. amalfitano, a. r. fasolino, p. tramontana, b. d. ta, and a. m. memon, "mobiguitar: automated model-based testing of mobile apps," ieee software, 32(5):pp.53–59, 2015.
[3] k. moran, m. l. v´asquez, c. bernal-c´ardenas, c. vendome, and d. poshyvanyk, "automatically discovering, reporting and reproducing android application crashes," in ieee international conference on software testing, verification and validation (icst), pp. 33–44, 2016.
[4] k. mao, m. harman, and y. jia, "sapienz: multi-objective automated testing for android applications," in 25th international symposium on software testing and analysis (issta), pp. 94–105, 2016.
[5] y. koroglu, a. sen, o. muslu, y. mete, c. ulker, t. tanriverdi, and y. donmez, "qbe: qlearning-based exploration of android applications," in ieee international conference on software testing, verification and validation (icst), 2018.

[6] koroglu, yavuz, and alper sen. ”tcm: test case mutation to improve crash detection in android.” international conference on fundamental approaches to software engineering. cham: springer international publishing, 2018.

[7] y. zhao, t. yu, t. su, y. liu, w. zheng, j. zhang, and w. g. j. halfond, “recdroid: automatically reproducing android application crashes from bug report,“ in proceedings of the 41st international conference on software engineering, icse, pp.128–139. ieee press, 2019.

[8] y. koroglu and a. sen, “reinforcement learning-driven test generation for android gui applications using formal specifications,“ arxiv preprint arxiv:1911.05403, 2019.

[9] dos santos, e. c., & vilain, p. (2018). automated acceptance tests as software requirements: an experiment to compare the applicability of fit tables and gherkin language. in agile processes in software engineering and extreme programming: 19th international conference, xp 2018, porto, portugal, may 21–25, 2018, proceedings 19 (pp. 104119). springer international publishing.

[10] chan, leong, liliya hogaboam, and renzhi cao. ”artificial intelligence in tourism and hospitality.” applied artificial intelligence in business: concepts and cases. cham: springer international publishing, 2022. 213229

[11] . a. sapan, b. oztekin, e¨ unsal and a. s¸en, ”testing openapi bank-¨ ing payment system with model based test approach,” 2020 turkish national software engineering symposium (uyms), istanbul, turkey, 2020, pp. 1-4, doi: 10.1109/uyms50627.2020.9247010.

[12] “model with ease,” graphwalker. [online]. available: https://graphwalker.github.io/.

[13] “webdriver,” webdriver: documentation for selenium. [online]. available: https://www.selenium.dev/documentation/en/webdriver/. https://github.com/graphwalker/graphwalker-project/wiki.

[14] a. sen and m. s. abadir, “coverage metrics for verification of concurrent systemc designs using mutation testing,” 2010 ieee international high level design validation and test workshop (hldvt), anaheim, fl, usa, 2010, pp. 75-81.