**TDTP**

**Documentation**

Academic year: 2023-2024, Spring Semester

Team Name: ECTM

Team members:

Țăpoi Marius-Ștefan (937),

Turcu Remus-Gabriel (937),

Stoicovici Ioan (937),

Iancu Ștefan-Cristian (731)

**Contents**

[1. Application Details. Investigated Features 3](#_heading=h.gjdgxs)

[2. AC. IOs 3](#_heading=h.30j0zll)

[3. Testing Mission 3](#_heading=h.1fob9te)

[4. Testing Strategy](#_heading=h.3znysh7) 4

[5. Selected Test Design Techniques](#_heading=h.2et92p0) 4

[6. Test Design. Test implementation. Test execution. Test Report 4](#_heading=h.tyjcwt)

[6.1. Test Design 4](#_heading=h.3dy6vkm)

[6.2. Test Implementation. Test Execution 4](#_heading=h.1t3h5sf)

[6.3. Test Report 5](#_heading=h.4d34og8)

[7. Issue Reporting 5](#_heading=h.2s8eyo1)

[8. Conclusions. Lessons Learned 5](#_heading=h.3rdcrjn)

# Application Details. Investigated Features

*TravelTracker is an application for creating a travel journal, called Bucket List, which aims to help users make the best decisions regarding the places they visit. TravelTracker inspires its users in the following ways: they can choose a public destination from those presented, which are available to everyone, or they can receive a suggestion from its chatbot called TravelBot. Additionally, for better spatial orientation, the application allows the visualisation of the Bucket List on the map, named "My Google Map" .*

*During testing, our primary focus will be on ensuring the quality and reliability of the latest sprint delivered to the customer in TravelTracker. Our testing scope encompasses both newly added features and existing functionalities. Specifically, we will thoroughly test the login functionality, destination voting, pin-pointing destinations from the bucket list on the map, and the pagination feature. Our goal is to verify that these key features perform as expected, delivering a seamless user experience while upholding the highest standards of quality and security.*

*Additionally, we will diligently assess the application for potential vulnerabilities, conducting rigorous testing on critical areas such as password registration, session management, API integrations, and resilience against brute force attacks. Our aim is to deliver a secure and dependable platform that not only inspires wanderlust but also prioritises the privacy and safety of our users' data.*

# AC. IOs

Application Context: You are part of a beginner testing team, which is very passionate and ambitious. The testing team is given an almost finished product to be tested

Part I: The project manager wants your team to test the latest sprint delivered to the costumer

Part II: As the project manager overseeing the security assessment of the application, the team needs to ensure the robustness and resilience of the application against potential security threats.

# Testing Mission

**Testing mission for Part I.**

First, the team will start by thoroughly understanding the features, changes, and updates delivered in the latest sprint. Then, review the requirements and acceptance criteria specified for the sprint to ensure alignment with testing objectives. There needs to be some assurance that the previously implemented features in past sprints are still working as expected.

**Testing mission for Part II.**

The team will start by testing any flaws in the authentication and authorization mechanisms that can lead to unauthorised access to sensitive data or functionalities. This includes vulnerabilities such as weak password policies, insufficient account lockout mechanisms, or improper session management. Furthermore, input validations, error handling and third-party integrations will be verified.

# Testing Strategy

**Testing Strategy for Part I:**

Focusing on testing the functionalities added in the latest sprint, while also ensuring that previously implemented features are functioning correctly, we employ a combination of model-based testing, methodical testing, and regression-based testing. Model-based testing helps us to create a conceptual model of the system’s behaviour, allowing us to verify the new features in a structured manner. Methodical testing, such as Equivalence Class Partitioning (ECP) and Boundary Value Analysis (BVA), enables us to systematically explore input ranges and edge cases, ensuring comprehensive coverage. Regression testing ensures that existing functionalities have not been adversely affected by the latest changes, maintaining the application's stability and reliability. By combining these strategies, we aim to comprehensively validate both the new and existing features of the application, mitigating the risk of regressions while also uncovering any potential defects or usability issues.

**Testing Strategy for Part II:**

We begin by employing methodical testing, which relies on predefined sets of tests or test conditions, to ensure thorough coverage of the application's functionality and critical pathways. This approach allows us to validate the product against predefined criteria and detect any deviations from expected behaviour. Additionally, we utilise process-oriented strategies and analytical approaches, integrating quick testing techniques such as ad-hoc testing and risk-based testing. Ad-hoc testing allows us to quickly explore specific areas of concern or high-risk functionalities, while risk-based testing helps us prioritise testing efforts based on the level of risk associated with different features. By combining these strategies, we aim to achieve comprehensive testing coverage while maximising the speed and effectiveness of our testing efforts, particularly in ensuring the security of the application through methods like brute force testing.

# Selected Test Design Techniques

For Part I, we employ several test design techniques to ensure comprehensive coverage and robust testing of the application's new and existing functionalities. Using Analytical, Methodical, Model-based, and Regression-based strategies, we apply Equivalence Class Partitioning (ECP) to divide input data into valid and invalid partitions, allowing us to test representative values from each partition to identify defects. Boundary Value Analysis (BVA) is used to test the edges of input ranges, as defects are often found at the boundaries of input domains. Function testing ensures that each function works correctly by verifying its output for given inputs, focusing particularly on user voting. Logical Expressions (LE) are applied to map destination pinpointing, ensuring that logical conditions governing feature behaviour are correctly implemented. For Part II, our focus shifts to security testing, employing Analytical, Process-oriented, and Methodical strategies. We integrate Quick Testing (QT) techniques such as ad-hoc testing to dynamically explore the application’s functionality without predefined test cases, enabling us to identify security weaknesses quickly. Additionally, we use Brute Force Testing to evaluate the robustness of login mechanisms against repeated attempts with different credentials, ensuring the application's resilience against unauthorised access. This approach is crucial for features such as password policy and session management, where security is paramount. By integrating these diverse techniques, we aim to thoroughly evaluate both functionality and security, ensuring a reliable and secure application for end users.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Part** | **Test Strategy** | **Test Design Technique** | **Dimension covered** | **Students and Features** |
| Part I | Analytical  Methodical  Model-based  Regression-based | Equivalence Class Partitioning(ECP) | Coverage | *Țăpoi Marius-Ștefan (Login)* |
| *LE  (Logical Expressions)* | Decision | *Turcu Remus-Gabriel*  *(Map destination pinpointing)* |
| *BVA (Boundary Value Analysis)* | Coverage | *Stoicovici Ioan*  *(Pagination)* |
| *Function testing* | Coverage | *Iancu Ștefan-Cristian*  *(User voting)* |
| Part II | Analytical  Process-oriented Strategy  Methodical | Quick Testing (QT) | Security | *Țăpoi Marius-Ștefan (Register - password policy)* |
| Quick Testing (QT) | Security | Turcu Remus-Gabriel  (Session management) |
| Quick Testing (QT) | Security | Stoicovici Ioan  (Secured API) |
| *Brute force testing* | Security | *Iancu Ștefan-Cristian (Login)* |

# Test Design. Test implementation. Test execution. Test Report

# Test Design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Information objective (Part I): The project manager wants your team to test the latest sprint delivered to the costumer* | | | | |
| **Student** | **Feature(s)** | **Test Design Technique** | **Details** | **Input, Expected output** |
| *Țăpoi Marius-Ștefan* | *Login* | Equivalence Class Partitioning(ECP) | *Login is tested using all the possible combinations of valid, invalid and empty username and password strings* | |  |  |  | | --- | --- | --- | | ***username*** | ***password*** | ***Expected Output*** | | *user1* | *user1* | *Passed* | | *user11* | *user1* | *Failed* | | *user1* | *abc* | *Failed* | | *abc* | *user1* | *Failed* | | *marius* | *tapoi* | *Failed* | |  | *user1* | *Failed* | | *user1* |  | *Failed* | |  |  | *Failed* | |
| *Turcu Remus-Gabriel* | *Map destination pinpointing* | *LE  (Logical Expressions)* | *The map pins are displayed based on the country and city of the destination. If the city is not valid, then the country is taken into consideration. If none of them are valid, the pin is not displayed.* | |  |  |  | | --- | --- | --- | | ***Country is valid*** | ***City is valid*** | ***Expected Outcome*** | | *T* | *T* | *Pin is on city’s centre* | | *T* | *F* | *Pin is on country’s centre* | | *F* | *T* | *Pin is on city centre* | | *F* | *F* | *Pin is not displayed* | |
| *Stoicovici Ioan* | *Pagination* | BVA  (Boundary Value Analysis) | The pagination is tested by trying all the possible combinations of the page number and size, and the result of the function call is compared with the expected list at that page. The mock data for the pages had a size of 100 entities. | |  |  |  | | --- | --- | --- | | **Page Number** | **Page Size** | **Expected Output** | | 0 | 100 | All the existing destinations | | 100 | 1 | Empty page | | 0 | 1 | The first destination in the list | | 1 | 100 | Empty page | | -1 | 10 | Error | | 1 | 0 | Error | | 1 | Integer.Max\_value | Error | | Integer.Max\_value | 10 | Error | |
| *Iancu Ștefan-Cristian* | *Destination voting* | *Function testing* | *The function testing for the user voting system involves verifying the correctness of the ‘UserVotesService’ methods: ‘addUserVotes’ and ‘existsUserVotes’ through a combination of valid and invalid input scenarios* | |  |  | | --- | --- | | *Input* | *Expected Output* | | *user id: 1 vote id: 1 Existing vote: no* | *Passed* | | *user id: 1 vote id: 1 Existing vote: yes* | *Failed* | | *user id: null vote id: 1* | *Passed* | | *user id: 1 vote id: null* | *Passed* | |

# 

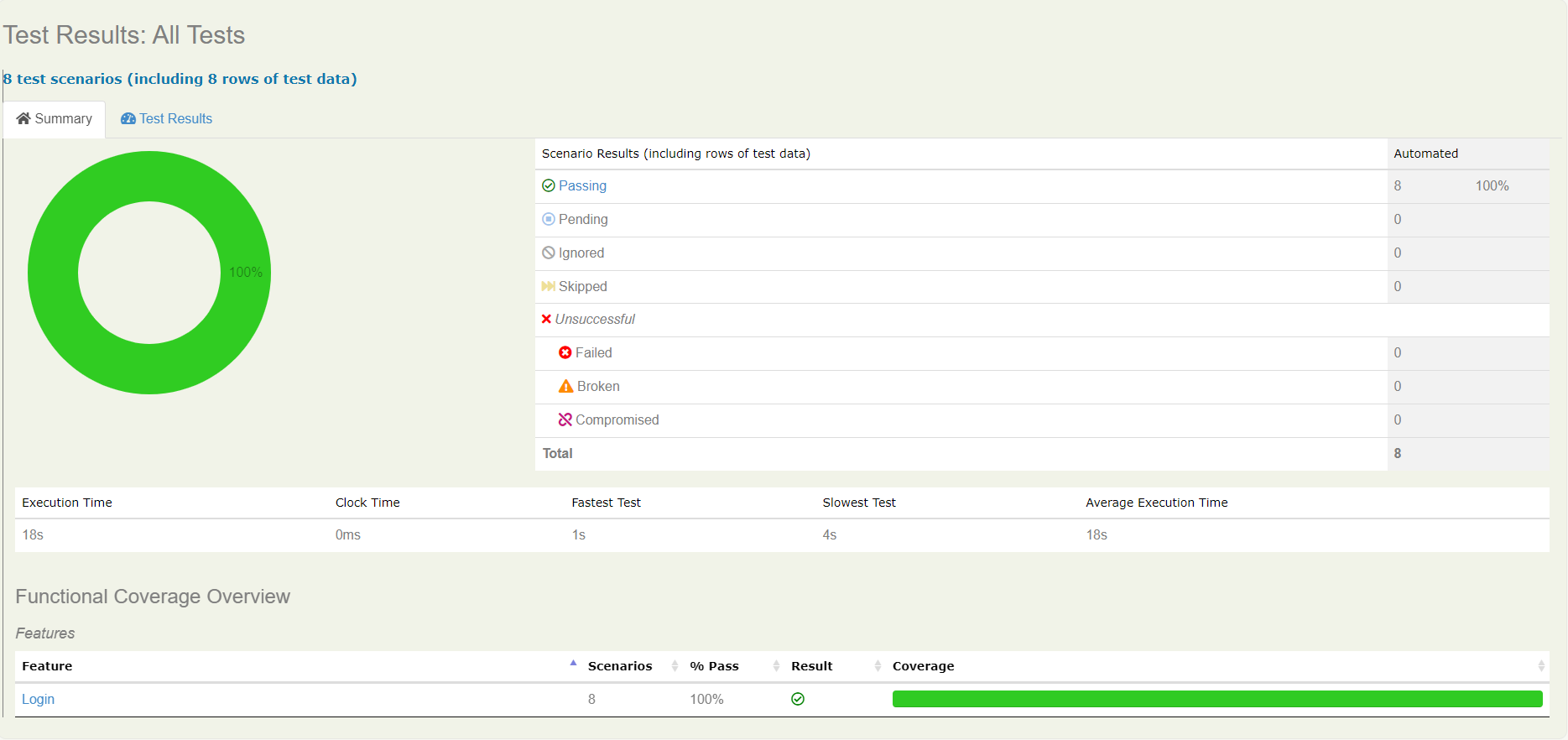
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Information objective (Part II): As the project manager overseeing the security assessment of the application, the team needs to ensure the robustness and resilience of the application against potential security threats.* | | | | |
| **Student** | **Feature(s)** | **Test Design Technique** | **Details** | **Input, Expected output** |
| *Țăpoi Marius-Ștefan* | *Register - password policy* | *Quick testing* | *The password policy from the register feature is tested. Registering should not be allowed if a password is not at least 6 characters long, exceeds 50 characters, does not contain at least one uppercase letter one lower case letter, a numeric character and a special character* | |  |  | | --- | --- | | ***Input*** | ***Expected Output*** | | *Pwd1!* | *Failed* | | *password123* | *Failed* | | *PASSWORD123* | *Failed* | | *Password* | *Failed* | | *Password123* | *Failed* | | *Password123!* | *Passed* | | *"<51 characters>"* | *Failed* | |
| *Turcu Remus-Gabriel* | *Session management* | *Quick Testing* | *Verify the access to pages after being logged out, in order to see how the session is managed inside the frontend app.* | |  |  |  | | --- | --- | --- | | ***Steps*** | ***Input*** | ***Expected output*** | | *Logout, go to the previous page and reload it* | *Valid user or admin account* | *Page should not load anymore* | | *Access the map page after being logged out* | *Valid user or admin account* | *Page should not load* | |
| *Stoicovici Ioan* | *Secured API* | Quick Testing | The goal of this test is to ensure that the privacy of all the users is not violated, that the data is truly private and can not be changed by another user. | |  |  | | --- | --- | | **Input Data** | **Expected output** | | Current user id, destination data | Successful | | Current user id, destination tip data | Successful | | Different User id, destination data | Error message | | Different User id, destination tip data | Error message | | No user Id, destination data | Error message | | No user Id, destination tip data | Error message | |
| *Iancu Ștefan-Cristian* | *Login* | *Brute force testing* | *The brute force login testing process involves trying out different usernames and passwords combinations to check the login functionality* | |  |  | | --- | --- | | *Input* | *Expected Output* | | usernames: user1-user10  passwords:  password1-password10 | Failed login | |

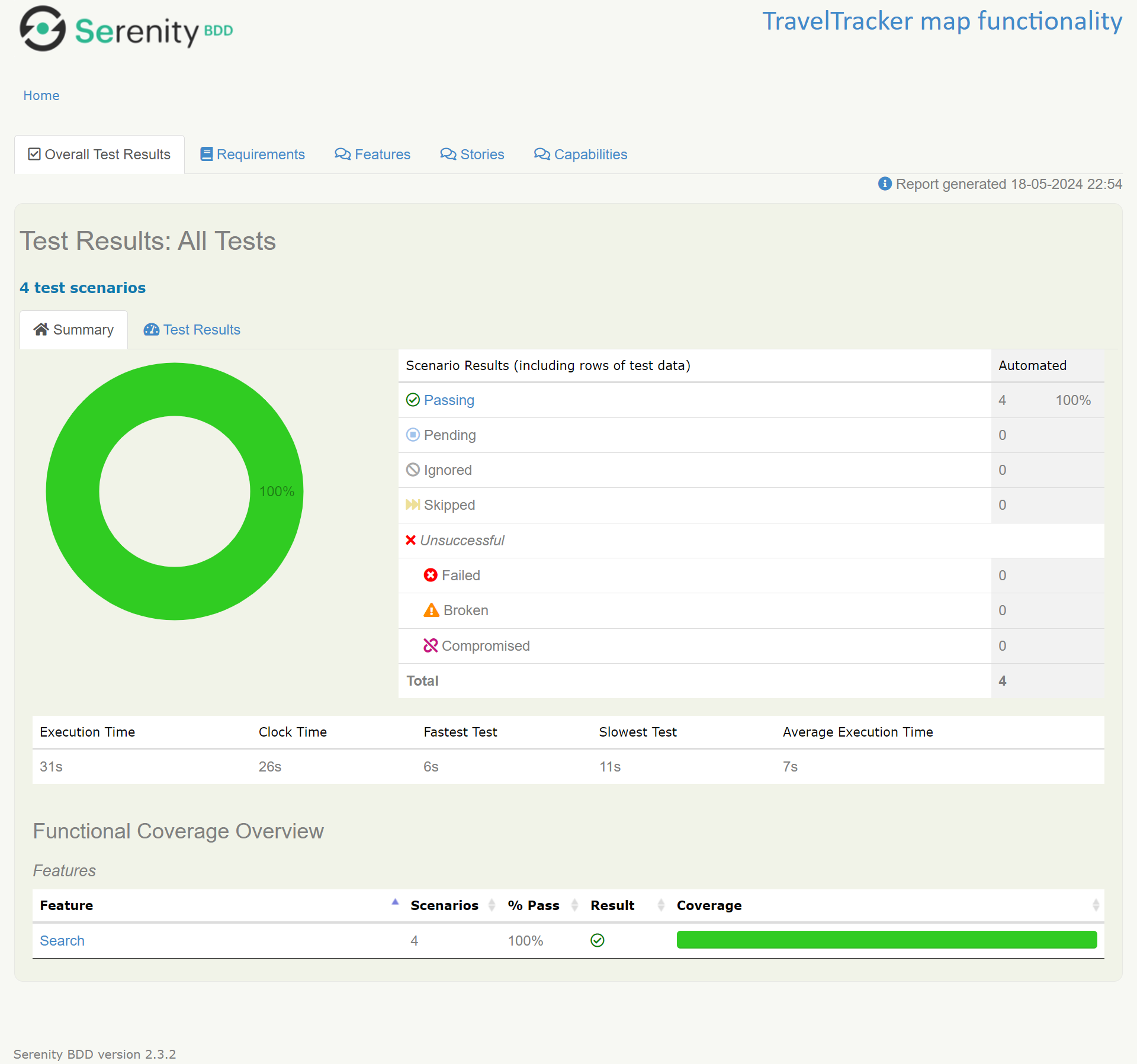
# Test Implementation. Test Execution

|  |  |  |  |
| --- | --- | --- | --- |
| **Part** | **Student** | **Feature(s)** | **Input, Expected Output, Actual Output** |
| *Part I* | *Țăpoi Marius-Ștefan* | *Login* | |  |  |  |  |  | | --- | --- | --- | --- | --- | | ***TCs*** | ***Input*** | | ***Expected Output*** | ***Actual Output*** | | ***username*** | ***password*** | | *TC01* | *user1* | *user1* | *Passed* | *Passed* | | *TC02* | *user11* | *user1* | *Failed* | *Failed* | | *TC03* | *user1* | *abc* | *Failed* | *Failed* | | *TC04* | *abc* | *user1* | *Failed* | *Failed* | | *TC05* | *marius* | *tapoi* | *Failed* | *Failed* | | *TC06* |  | *user1* | *Failed* | *Failed* | | *TC07* | *user1* |  | *Failed* | *Failed* | | *TC08* |  |  | *Failed* | *Failed* | |
| *Part I* | *Turcu Remus-Gabriel* | *Map destination pinpointing* | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | ***TCs*** | ***Input*** | | | | ***Expected Output*** | ***Actual Output*** | | ***Name*** | ***Country*** | ***City*** | ***Description*** | | ***TC01*** | ***Casa poporului*** | ***Romania*** | ***Bucharest*** | ***Bucharest is the capital and largest city of Romania. (...)*** | *Pin is on city’s centre* | *Pin is on city’s centre* | | ***TC02*** | ***Destination1*** | ***Romania*** | ***Inexistent*** | ***Descriere*** | *Pin is on country’s centre* | *Pin is on country’s centre* | | ***TC03*** | ***Destination1*** | ***Inexistent*** | ***Bistrita*** | ***Descriere*** | *Pin is on city centre* | *Pin is on city centre* | | ***TC04*** | ***Destination1*** | ***Inexistent*** | ***Inexistent*** | ***Descriere*** | *Pin is not displayed* | *Pin is not displayed* | |
| Part I | Stoicovici Ioan | *Pagination* | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **TCs** | **Input** | | **Expected Output** | **Actual Output** | | **Page Number** | **Page Size** | | **TC1** | 0 | 100 | All the existing destinations | All the existing destinations | | **TC2** | 100 | 1 | Empty Page | Empty page | | **TC3** | 0 | 1 | The first destination in the list | The first destination in the list | | **TC4** | 1 | 100 | Empty Page | Empty Page | | **TC5** | -1 | 10 | Error | Error | | **TC6** | 1 | 0 | Error | Error | | **TC7** | 1 | Integer.Max\_Value | Error | Error | | **TC8** | Integer.Max\_value | 10 | Error | Error | |
| Part II | Iancu Stefan-Cristian | Login | |  |  |  |  | | --- | --- | --- | --- | | Steps | Input | Expected Output | Actual Output | | Try to login with each username password combination | usernames: user1-user10  passwords:  password1-password10 | Failed login | Failed login | |

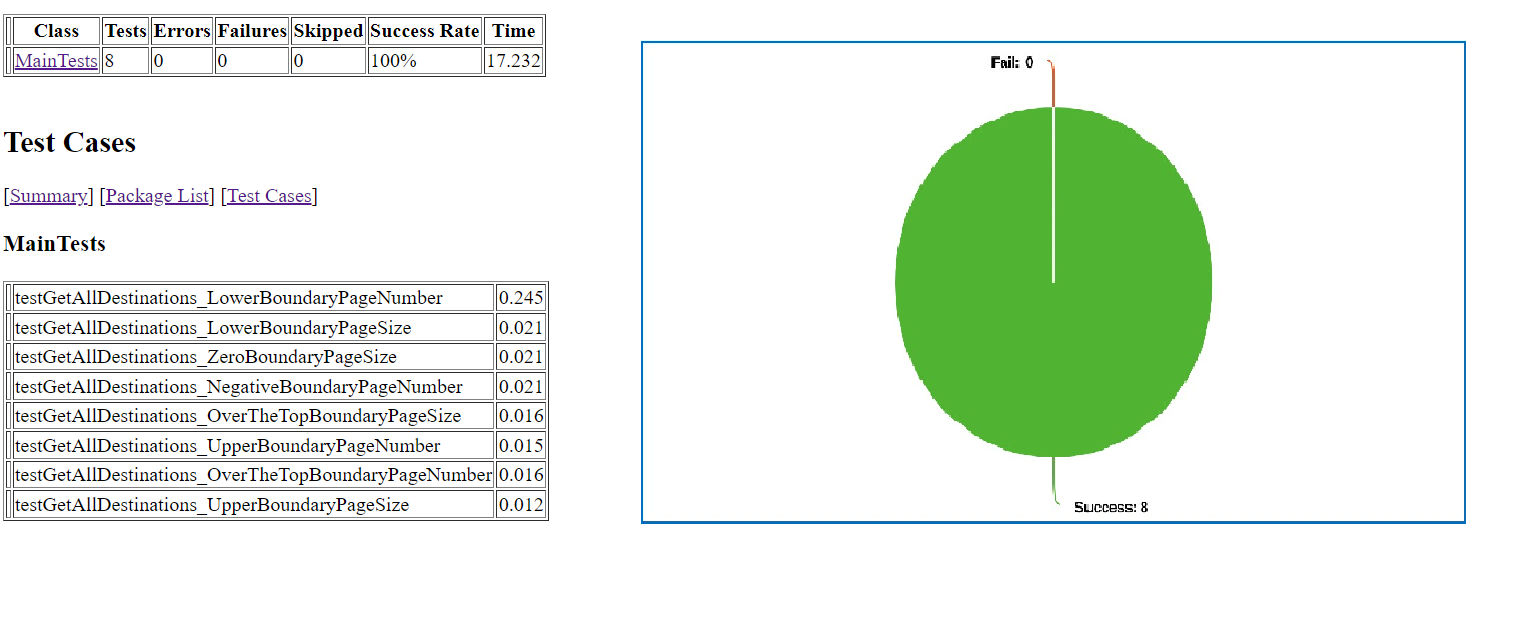
# Test Report

*Țăpoi Marius-Ștefan (Login)*

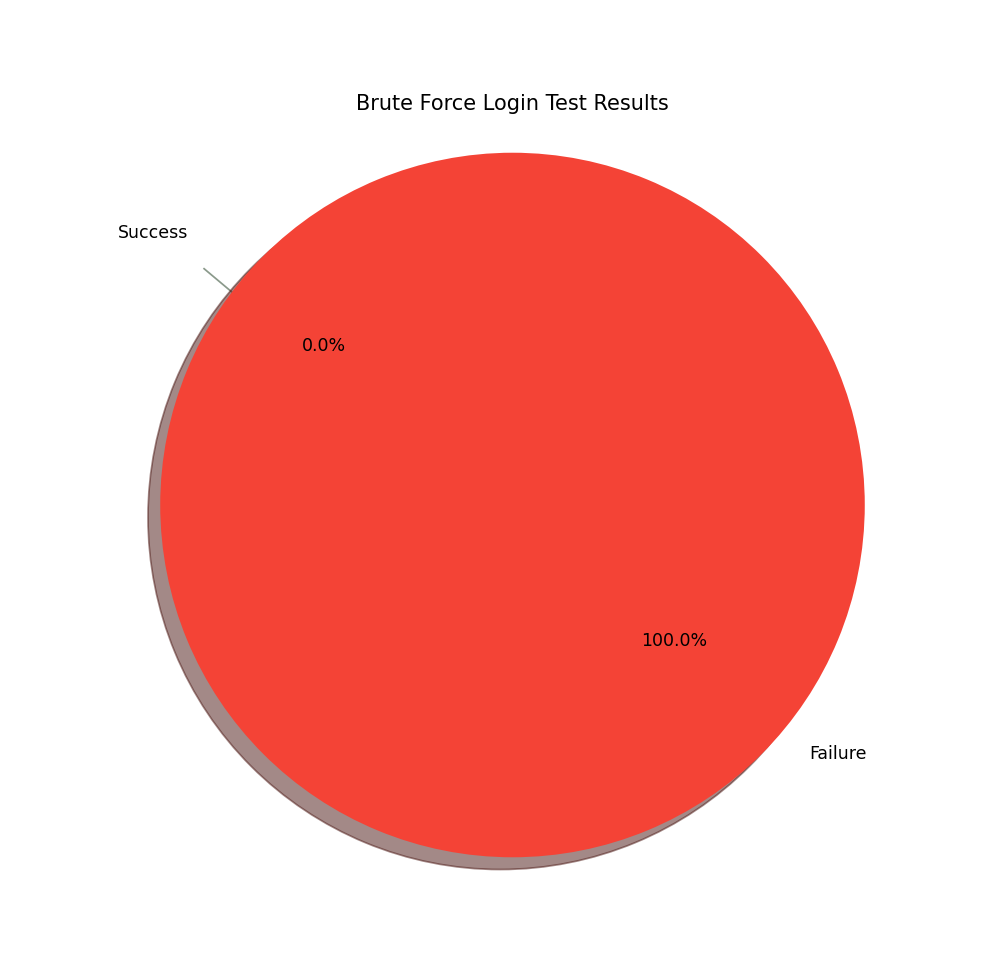


Turcu Remus-Gabriel(*Map destination pinpointing)*

Stoicovici Ioan(*Pagination*)

**

*Iancu Stefan-Cristian(Login Brute Force Testing)*

**

# Issue Reporting

One of the critical problems of this project is its security, so this subchapter will describe the issue using the RIMGEN strategy for writing a bug report. The elements of RIMGEN used in this report were, R (replicate), this report includes detailed steps to replicate the problem using Postman, ensuring that another developer can reproduce the problem. I (Isolate), the report explains the problem with two different scenarios and in the shortest steps, containing only the necessary information. The report also explains only one issue. And N (Neutral Tone), the report maintains a professional tone throughout, focussing on the issue and its implications rather than blaming someone and making subjective statements. Here is the report:

**Product:** Travel Tracker

**Report Date:** 18.05.2024

**Issue Summary:** Not handling authorization and authentication properly for certain scenarios. Allowing unauthorised access and actions by unauthenticated users.

**Version/ Found on:** 13.05.2024

**Hardware/Software**: Windows 10, 8GB RAM, AMD Ryzen 5, TravelTracker v1.0.0

**Importance/Severity**: Critical

**Reported by:** Team 5

**Issue type:** Defect

**Steps:**

1. Open Postman in web browser

2. For test scenario 1, when the user is logged in. Login to the Travel Tracker application using valid user credentials. Obtain the authentication token from the login response.

3. Send a POST request to the Travel Tracker API endpoint for adding a destination. Include destination details in the request body. Observe and record the response.

4. Send a POST request to the Travel Tracker API endpoint for adding a destination tip. Include tip details in the request body. Record the response.

5. Send a POST request to the Travel Tracker API endpoint for adding a tip with another user's ID. Include the tip details and another user's ID in the request body. Record the response.

6. Send a POST request to the Travel Tracker API endpoint for adding a destination with another user's ID. Include the destination data and another user's ID in the request body. Record the response.

7. Go to scenario 2, Restart the application for the user to logout.

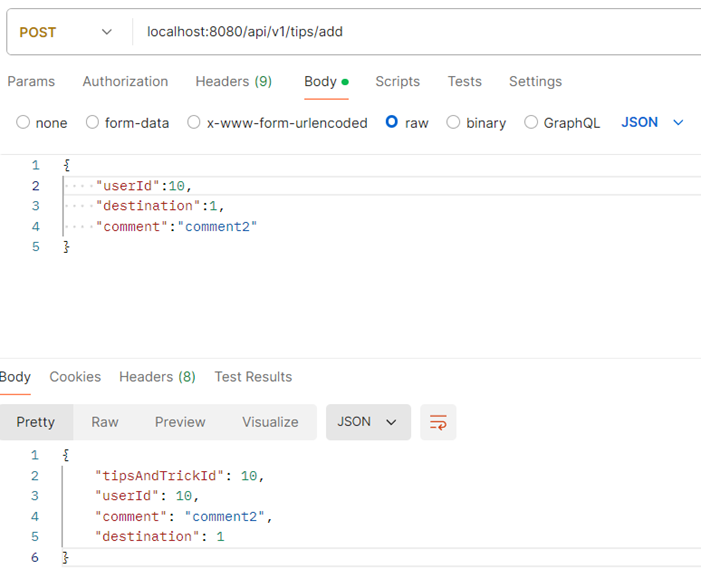
8. Send a POST request to the Travel Tracker API endpoint for adding a tip with a random but valid user ID. Include the tip details and the user's ID in the request body. Record the response.

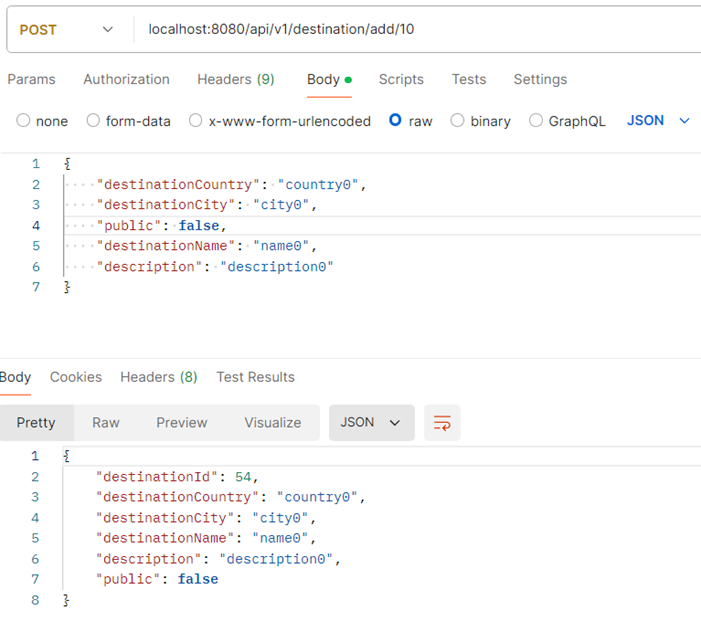
9. Send a POST request to the Travel Tracker API endpoint for adding a destination with a random but valid user's ID. Include the destination details and the user's ID in the request body. Record the response.

**Expected output:** The addition of destination and tip under the current logged in user to succeed, and the attempts to add a destination or tip if the user is not logged in, or the user tries to change the private data of another user to fail.

**Actual Output:** The logged in user can indeed change its private data, making this test succeed, but the current user can also change private data of another user, making the tests fail.

**Screenshots:**





**Other details/Remarks:** The screenshots above are an example of a logged in user whose id is 8, can add a destination and a tip under another user's id, in this case 10. The application seems to do absolutely no check on the user that is currently logged in, and the user's accessibility seems to allow him to do anything, it feels like every account is an admin account. The data seems to be kept all together, for all the users, and the front end sends a destination and a user id, but in the eventuality that the frontend makes a mistake and sends a wrong user id, the privacy of all users will be violated.

# Conclusions. Lessons Learned

*Testing the key features of TravelTracker, a travel application designed to help users maintain a list of desired travel destinations, proved to be an exciting and informative experience. Our team evaluated the application from both functional and security perspectives. The main conclusion from our project is straightforward: while the application's functionalities perform as expected, it lacks critical security protections, such as defences against brute force attacks and secure API configurations. Throughout the project, we learned the critical importance of selecting an appropriate testing strategy and employing correct test design techniques to cover various testing dimensions effectively. This approach ensured comprehensive coverage of functional and security aspects, enhancing the overall quality and reliability of the application. By meticulously planning our testing efforts, we were able to identify and address potential issues more efficiently.*

*Through effective collaboration and regular progress reviews, we delivered a comprehensive and high-quality test suite. Each team member was dedicated to their tasks and successfully met the project deadlines.Our team members shared a similar academic background, which facilitated a cohesive working environment. However, our varied professional experiences significantly enriched the project. Some members had backgrounds as QA testers, others as software developers, and some had specialised further in testing methodologies. This diversity allowed each of us to leverage our preferred tools and methods while also learning from and teaching our peers. Initially, our team's collective knowledge level was average, but by the end of the project, everyone had substantially expanded their expertise.*

*The tools we utilised were integral to our success. We employed Serenity, Selenium, and SpringBootTest for automation, which streamlined our testing processes and enhanced our efficiency. Postman was invaluable for testing API calls, allowing us to verify the application's backend interactions thoroughly. GitHub was our version control system of choice, ensuring that code changes were managed effectively and collaboratively. These tools, combined with our collaborative efforts, enabled us to achieve a comprehensive understanding of TravelTracker’s strengths and weaknesses, ultimately leading to a more robust and secure application.*