robotframework

Settings

Library SeleniumLibrary

Library RequestsLibrary

Library DatabaseLibrary

Library OperatingSystem

Variables

${BASE\_URL} http://your-bus-booking-app.com

${USERNAME} testuser

${PASSWORD} testpassword

${DATABASE\_URL} mysql://user:password@host:port/database

${API\_KEY} your-api-key

Keywords

Open Browser

Open Browser ${BASE\_URL} browser=Chrome

Close Browser

Close Browser

Login

[Arguments] ${username} ${password}

Input Text id=username ${username}

Input Text id=password ${password}

Click Button id=login-button

Search Bus

[Arguments] ${source} ${destination} ${date}

Input Text id=source ${source}

Input Text id=destination ${destination}

Input Text id=date ${date}

Click Button id=search-button

Select Bus

[Arguments] ${bus\_id}

Click Element xpath=//div[@data-bus-id='${bus\_id}']

Book Seats

[Arguments] ${seat\_numbers}

Click Button id=book-button

# Select seats using seat\_numbers

Confirm Booking

Click Button id=confirm-booking-button

Get Booking Details

[Arguments] ${booking\_id}

${response} = Get Request ${BASE\_URL}/bookings/${booking\_id}

# Extract booking details from response

Validate Booking Confirmation

[Arguments] ${booking\_id}

${booking\_details} = Get Booking Details ${booking\_id}

# Verify booking details match expected values

Make Payment

[Arguments] ${payment\_method}

# Simulate payment process using payment\_method

Validate Payment Confirmation

# Verify payment confirmation page or email

Get User Profile

${response} = Get Request ${BASE\_URL}/users/profile

# Extract user profile details from response

Update UserProfile

[Arguments] ${new\_name} ${new\_email}

# Update user profile with new\_name and new\_email

Validate UserProfileUpdate

# Verify updated user profile details

Get Bus Details

[Arguments] ${bus\_id}

${response} = Get Request ${BASE\_URL}/buses/${bus\_id}

# Extract bus details from response

Validate Bus Details

[Arguments] ${bus\_id}

${bus\_details} = Get Bus Details ${bus\_id}

# Verify bus details match expected values

Execute SQL Query

[Arguments] ${query}

${cursor} = Connect To Database ${DATABASE\_URL}

${result} = Execute SQL ${cursor} ${query}

Disconnect From Database ${cursor}

# Process result based on query

Validate Data Integrity

# Execute SQL queries to verify data consistency

Perform API Authentication

${response} = Get Request ${BASE\_URL}/api/protected-resource headers=Authorization: Bearer ${API\_KEY}

# Verify successful authentication

Encrypt Data

# Implement encryption logic using appropriate library

Decrypt Data

# Implement decryption logic using appropriate library

Validate Encryption

# Verify data encryption and decryption

Perform Load Test

# Use appropriate load testing tools to simulate user traffic

Validate Performance

# Analyze load test results to ensure performance meets requirements

Test Suites

Presentation Layer

[Tags] presentation

Test Case Validate UI Components

Open Browser

# Verify UI elements like buttons, text fields, etc.

Close Browser

Business Logic Layer

[Tags] business-logic

Test Case Validate Seat Allocation

Open Browser

Login ${USERNAME} ${PASSWORD}

Search Bus Bangalore Chennai 2024-03-15

Select Bus 1234

Book Seats 1A, 2B

Confirm Booking

Validate Booking Confirmation ${booking\_id}

Close Browser

Test Case Validate Booking Confirmation

Open Browser

Login ${USERNAME} ${PASSWORD}

Search Bus Bangalore Chennai 2024-03-15

Select Bus 1234

Book Seats 1A, 2B

Confirm Booking

Validate Booking Confirmation ${booking\_id}

Close Browser

Data Access Layer

[Tags] data-access

Test Case Validate Data Retrieval

Execute SQL Query SELECT FROM bookings

# Verify data retrieved from database

Test Case Validate Data Integrity

Execute SQL Query SELECT COUNT() FROM bookings WHERE status='confirmed'

# Verify data consistency based on business rules

Security Layer

[Tags] security

Test Case Validate API Authentication

Perform API Authentication

Test Case Validate Data Encryption

Encrypt Data

Decrypt Data

Validate Encryption

Deployment Layer

[Tags] deployment

Test Case Perform Load Test

Perform Load Test

Validate Performance

Explanation:

1. Test Suites: The code defines separate test suites for each layer: Presentation, Business Logic, Data Access, Security, and Deployment. This allows for organized testing and reporting.

2. Keywords: Reusable keywords are defined for common actions like opening the browser, logging in, searching for buses, booking seats, confirming bookings, executing SQL queries, performing API authentication, encrypting/decrypting data, and performing load tests.

3. Test Cases: Each test suite contains test cases that perform specific checks:

- Presentation Layer: Validates UI components and responsiveness.

- Business Logic Layer: Checks microservice operations like seat allocation, booking confirmation, and booking history.

- Data Access Layer: Executes SQL queries to validate data retrieval and integrity.

- Security Layer: Tests API authentication, data encryption, and decryption.

- Deployment Layer: Performs load testing and validates performance.

Notes:

- Replace placeholders like ${BASE\_URL}, ${USERNAME}, ${PASSWORD}, ${DATABASE\_URL}, and ${API\_KEY} with your actual values.

- You may need to adjust the keywords and test cases based on your specific application and requirements.

- This code provides a basic framework. You can further enhance it by adding more test cases, keywords, and reporting features.

- Consider using a test data management strategy to manage test data effectively.

- Integrate this code with your CI/CD pipeline for automated testing and deployment.