Automated GUI-Based Testing of Blekingetrafiken.se

Assignment 2

PA2552 VT26

Software Testing

Author: Md Asif Iqbal Ahmed
Submission Date: February 16, 2026

Blekinge Institute of Technology

Karlskrona, Sweden

1 Introduction

This report describes the automated GUI test suite built for **Blekingetrafiken.se**, the public transport website for Blekinge, Sweden. The test suite covers 10 feature requirements using Selenium WebDriver with C# and the NUnit test framework.

Technology Stack:

- **Language:** C# (.NET 8.0)
- **Test Framework:** NUnit 3
- **Browser Automation:** Selenium WebDriver 4.40
- **Browser:** Google Chrome

2 Feature Requirements

2.1 US1: Journey Planning

User Story: As a traveler, I want to search for a journey between two locations with time options, so that I can plan my trip using public transport.

Acceptance Criteria:

- The home page displays a journey planner with "Från" (From) and "Till" (To) input fields
- A time selection section ("När vill du åka?") is visible

2.2 US2: Extended Journey Search

User Story: As a traveler, I want to access the journey search from the travel information section, so that I can search for trips beyond the home page.

Acceptance Criteria:

- The journey search page provides "Från" and "Till" input fields along with a "Sök" button
- An extended search page is accessible at `/reseinformation/sok-resa/` and loads successfully

2.3 US3: Traffic Information

User Story: As a traveler, I want to check current service disruptions and delays, so that I can update my travel plans.

Acceptance Criteria:

- The traffic information page shows keywords about service disruptions (e.g., "trafikläge", "störning", "försening")
- The page has an external link to Trafikverket for live rail traffic updates

2.4 US4: Stop/Station Search

User Story: As a traveler, I want to search for bus and train stops by name, so that I can find station information for my location.

Acceptance Criteria:

- The stations page lists at least 3 stations
- Known stations such as Karlskrona and Ronneby are displayed
- Transport type information (tåg/buss) is shown for each station

2.5 US5: Timetables

User Story: As a traveler, I want to view timetables for specific bus and train lines, so that I can plan recurring trips.

Acceptance Criteria:

- The timetables page displays the heading "Tidtabeller"
- Clickable links are available to access specific line timetables

2.6 US6: Ticket Information

User Story: As a traveler, I want to view available ticket types and their details, so that I can choose the best ticket for my needs.

Acceptance Criteria:

- The tickets page displays all five ticket types: Enkelbiljett, Flexbiljett, 24-timmarsbiljett, 30-dagarsbiljett, and 365-dagarsbiljett
- Each ticket type has a link to detailed information

2.7 US7: Zone Information

User Story: As a traveler, I want to view zone information, so that I can understand the fare zones for my journey.

Acceptance Criteria:

- The zones page has sub-section headings
- Zone maps in PDF or image format are available for download

2.8 US8: Customer Service

User Story: As a user, I want to access customer service resources, so that I can get help when needed.

Acceptance Criteria:

- The customer service page displays a heading

- Links to all main service sections are present: FAQ (Vanliga frågor), delay compensation (Förseningsersättning), lost and found (Hittegods), and contact (Kontakta oss)

2.9 US9: Navigation Menu

User Story: As a user, I want to navigate between main sections of the website using the menu, so that I can easily find the information I need.

Acceptance Criteria:

- The main navigation contains links to Biljetter, Reseinformation, and Kundservice
- Clicking each link navigates to the correct section with the expected URL path

2.10 US10: Accessibility Information

User Story: As a traveler with accessibility needs, I want to view accessibility information, so that I can plan accessible trips.

Acceptance Criteria:

- The accessibility page contains information for all three transport types (Buss, Tåg, Båt)
- Accessibility statement links are present

3 Development and Execution Times

Table 1 presents the development time and execution time for each user story.

Table 1: Development and execution times per user story

#	**User Story**	**Tests**	**Dev Time**	**Exec Time**
US1	Journey Planning	1	45 min	2 s
US2	Extended Journey Search	2	40 min	4 s
US3	Traffic Information	2	25 min	4 s
US4	Stop/Station Search	4	45 min	7 s
US5	Timetables	2	25 min	4 s
US6	Ticket Information	5	30 min	9 s
US7	Zone Information	2	25 min	3 s
US8	Customer Service	2	20 min	4 s
US9	Navigation Menu	3	25 min	7 s
US10	Accessibility Info	2	25 min	4 s
Project setup & infrastructure		–	2 h	–
Total		**25**	**∼7 h**	**∼58 s**

4 Test Execution Results

All 25 test cases executed successfully. The following log shows the full test run output.

```
$ dotnet test --logger "console;verbosity=detailed"

  Determining projects to restore...
  All projects are up-to-date for restore.
  BlekingetrafikenTests -> bin/Debug/net8.0/BlekingetrafikenTests.dll
Test run for BlekingetrafikenTests.dll (.NETCoreApp,Version=v8.0)
VSTest version 17.11.1 (arm64)

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.
NUnit Adapter 4.5.0.0: Test execution started
Running all tests in BlekingetrafikenTests.dll
   NUnit3TestExecutor discovered 25 of 25 NUnit test cases using Current Discovery mode,
        Non-Explicit run
  Passed JourneyPlanner_ShouldBeDisplayedWithTimeSelection [2 s]
  Passed JourneyResults_ExtendedSearchPageShouldExist [1 s]
  Passed JourneyResults_FormFieldsShouldBePresent [3 s]
  Passed TrafficInfo_ShouldContainDisruptionContent [2 s]
  Passed TrafficInfo_ShouldHaveTrafikverketLink [2 s]
  Passed Stations_ShouldDisplayKnownStation("Karlskrona") [2 s]
  Passed Stations_ShouldDisplayKnownStation("Ronneby") [2 s]
  Passed Stations_ShouldListMultipleStations [2 s]
  Passed Stations_ShouldShowTransportTypeInfo [1 s]
  Passed Timetables_ShouldDisplayCorrectHeading [2 s]
  Passed Timetables_ShouldHaveTimetableLinks [2 s]
  Passed Tickets_ShouldDisplaySpecificTicketType("Enkelbiljett") [2 s]
  Passed Tickets_ShouldDisplaySpecificTicketType("Flexbiljett") [2 s]
  Passed Tickets_ShouldDisplaySpecificTicketType("24-timmarsbiljett") [1 s]
  Passed Tickets_ShouldDisplaySpecificTicketType("30-dagarsbiljett") [2 s]
  Passed Tickets_ShouldDisplaySpecificTicketType("365-dagarsbiljett") [2 s]
  Passed Zones_ShouldHaveDownloadableContent [1 s]
  Passed Zones_ShouldHaveSubSections [2 s]
  Passed CustomerService_ShouldDisplayHeading [2 s]
  Passed CustomerService_ShouldHaveAllServiceLinks [2 s]
  Passed Navigation_BiljettLink_ShouldNavigateToTicketsPage [2 s]
  Passed Navigation_KundserviceLink_ShouldNavigateToCustomerServicePage [2 s]
  Passed Navigation_ReseinformationLink_ShouldNavigateToTravelInfoPage [3 s]
  Passed Accessibility_ShouldHaveAccessibilityStatementLinks [2 s]
NUnit Adapter 4.5.0.0: Test execution complete
  Passed Accessibility_ShouldHaveAllTransportSections [2 s]

Test Run Successful.
Total tests: 25
     Passed: 25
 Total time: 58.5132 Seconds
```

Figure 1 shows the test execution in the IDE test runner from a separate run. Small differences in execution time between runs are expected due to network and server response times.

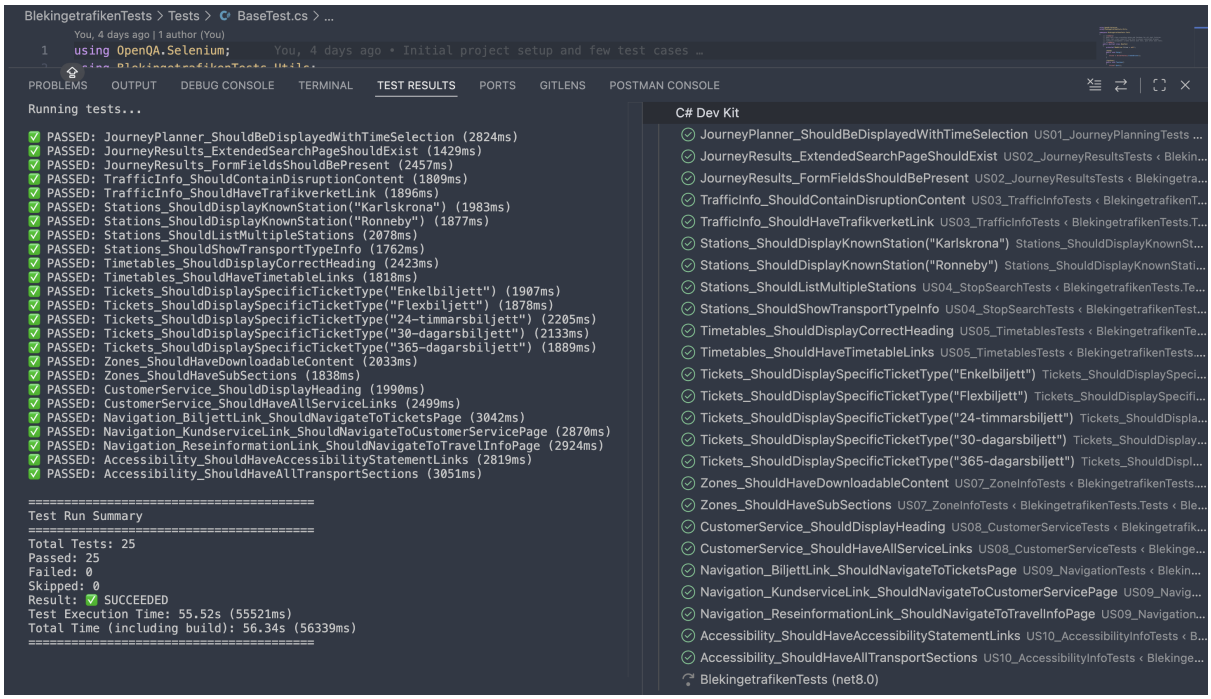

Figure 1: Screenshot of successful test execution (25/25 passed)

5 Test Suite Quality and Discussion

5.1 Quality Practices

The test suite is built to be easy to maintain and reliable. The following practices are used:

- Each page has its own Page Object class that holds all locators and actions. When the website HTML changes, only that page class needs to be updated.
- Every test opens a fresh browser via `[SetUp]` and closes it in `[TearDown]`. This stops tests from affecting each other.
- `WebDriverWait` with expected conditions is used instead of `Thread.Sleep`. This makes tests faster and more stable.
- NUnit's `[TestCase]` runs the same test with different inputs (e.g., US4 checks two stations, US6 checks five ticket types) without repeating test code.
- All URLs are stored in a single `TestConfig` class and browser settings in `DriverFactory`. Changes only need to be made in one file.
- The Cookiebot banner is closed once in a shared base class, so each test does not need to handle it.

5.2 Benefits of Script-Based GUI Testing

- Tests check the application from the user's point of view. They can find issues that unit or API tests miss, such as broken links, missing content, or navigation

problems.

- The full suite runs in about one minute. This is much faster than manually checking 25 scenarios across 10 pages.
- Tests act as living documentation. A failing test shows right away which feature broke, which is useful for regression testing after website updates.
- Running against the live production site can find real issues that a test environment might not show.

5.3 Drawbacks of Script-Based GUI Testing

- Changes to CSS selectors, page layout, or the cookie consent banner can break tests even when the actual features still work. POM reduces this problem but does not remove it fully.
- Tests against a live website are affected by network speed, server load, and content changes that the tester cannot control.
- Each test opens a new browser, loads the page, waits, and closes. This is much slower than unit tests.
- GUI tests can only check what is visible on the page. They cannot test backend logic, database state, or API responses directly.

This suite focuses on checking that content is present and that pages are structured correctly. It does not test full user workflows. For example, it checks that the journey planner form exists but does not submit a search and check the results. It also does not cover responsive design, mobile views, or other browsers. All tests run only in desktop Chrome.