**Easy Trip Bus Ticket Booking - Home Component Test Documentation(Website Module)**

**Overview**

This document provides detailed test documentation for the Home Component of the Easy Trip Bus Ticket Booking application. The tests ensure that the component behaves as expected, including form validation, navigation, and date handling.

**Test Environment**

* **Testing Framework**: Jasmine + Karma
* **Angular Version**: Standalone component testing
* **Browser**: Chrome 138.0.0.0 (Windows 10)
* **Test Execution Command**: ng test --include='\*\*/home.component.spec.ts'

**Test Suite Summary:**

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Description** | **Status** |
| should create | Verifies component initialization | Pass |
| should initialize with today's date | Checks default date setting | Pass |
| should update to today's date when setTodayDate is called | Tests setTodayDate() | Pass |
| should update to tomorrow's date when setTomorrowDate is called | Tests setTomorrowDate() | Pass |
| should swap locations correctly | Validates location swapping | Pass |
| should show alert when locations are empty | Tests form validation | Pass |
| should navigate when form is valid | Checks successful navigation | Pass |

**4. Detailed Test Cases**

**4.1 Component Initialization**

**Test Case**: should create

* **Objective**: Verify that the component initializes correctly.
* **Test Logic**: expect(component).toBeTruthy();
* **Expected Result**: Component instance should be created successfully.

**4.2 Date Handling**

**Test Case**: should initialize with today's date

* **Objective**: Ensure the component initializes with the current date.
* **Test Logic**:

const today = new Date().toISOString().split('T')[0];

expect(component.departureDate).toBe(today);

* **Expected Result**: departureDate should match today's date.

**Test Case**: should update to today's date when setTodayDate is called

* **Objective**: Test setTodayDate() updates the date correctly.
* **Test Logic**:

component.setTodayDate();

expect(component.departureDate).toBe(today);

* **Expected Result**: departureDate updates to today's date.

**Test Case**: should update to tomorrow's date when setTomorrowDate is called

* **Objective**: Test setTomorrowDate() updates to tomorrow's date.
* **Test Logic**:

const tomorrow = new Date();

tomorrow.setDate(tomorrow.getDate() + 1);

const tomorrowStr = tomorrow.toISOString().split('T')[0];

component.setTomorrowDate();

expect(component.departureDate).toBe(tomorrowStr);

* **Expected Result**: departureDate updates to tomorrow's date.

**4.3 Location Handling**

**Test Case**: should swap locations correctly

* **Objective**: Verify swapLocations() correctly swaps departure and destination.
* **Test Logic**:

component.leavingFrom = 'Bangalore';

component.goingTo = 'Chennai';

component.swapLocations();

expect(component.leavingFrom).toBe('Chennai');

expect(component.goingTo).toBe('Bangalore');

* **Expected Result**: leavingFrom and goingTo values are swapped.

**4.4 Form Validation**

**Test Case**: should show alert when locations are empty

* **Objective**: Ensure an alert appears if departure/destination is missing.
* **Test Logic**:

spyOn(window, 'alert');

component.leavingFrom = '';

component.goingTo = '';

component.onSearchSubmit();

expect(window.alert).toHaveBeenCalledWith('Please enter both departure and destination');

* **Expected Result**: Alert is shown when fields are empty.

**Test Case**: should navigate when form is valid

* **Objective**: Test successful navigation on valid form submission.
* **Test Logic**:

component.leavingFrom = 'Bangalore';

component.goingTo = 'Chennai';

component.onSearchSubmit();

expect(router.navigate).toHaveBeenCalledWith(['/availablebuses'], {

queryParams: {

from: 'Bangalore',

to: 'Chennai',

date: component.departureDate

}

});

* **Expected Result**: Navigation occurs with correct query parameters.

**5. Total Execution Result:**

* **Total Specs**: 7
* **Passed**: 7
* **Failed**: 0
* **Randomized Seed**: 38132

**6. Conclusion**:

All test cases for HomeComponent pass successfully. The component handles date management, location swapping, and form validation as expected. Further improvements can be made to eliminate warnings and increase test coverage.

**✅ Test Suite Status: PASSED**

**Easy Trip Bus Ticket Booking -Dash Board Component Test Documentation(Admin Module)**

**1. Overview**

This document provides detailed test documentation for the DashboardComponent of the EasyTrip Bus Ticket Booking application. The tests ensure that the dashboard correctly displays analytics, handles data processing, and manages charts.

**2. Test Environment**

* **Testing Framework**: Jasmine + Karma
* **Angular Version**: Standalone component testing
* **Browser**: Chrome 138.0.0.0 (Windows 10)
* **Test Execution Command**: ng test --include='\*\*/dashboard.component.spec.ts'

**3**. **Test Suite Summary:**

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Description** | **Status** |
| should create | Verifies Component initialization | Pass |
| should initialize with loading state | Checks initial Loading state | Pass |
| should fetch dashboard data on init | Tests data fetch on initialization | Pass |
| should handle API error | Tests error handling | Pass |
| should process dashboard data correctly | Validates data processing | Pass |
| should format currency correctly | Tests Currency formatting | Pass |
| should get correct status class | Checks CSS class assignment | Pass |
| should get correct status display text | Tests status text display | Pass |
| should initialize charts after view init | Tests chart initialization | Pass |
| should destroy charts on component destroy | Tests chart cleanup | Pass |
| should set revenue period and reinitialize chart | Tests period switching | Pass |
| should get correct bus type color | Tests bus type colours | Pass |
| Recent Bookings Processing (4 sub-tests) | Tests booking data processing | Pass |

**Total Specs**: 17  
**Passed**: 17  
**Failed**: 0

**4. Detailed Test Cases**

**4.1 Component Initialization**

**Test Case**: should create

* **Objective**: Verify component creation
* **Test Logic**:

expect(component).toBeTruthy();

**Test Case**: should initialize with loading state

* **Objective**: Verify initial state
* **Test Logic**:

expect(component.isLoading).toBe(true);

expect(component.error).toBeNull();

**4.2 Data Handling**

**Test Case**: should fetch dashboard data on init

* **Objective**: Test data loading
* **Test Logic**:

component.ngOnInit();

tick();

expect(component.isLoading).toBe(false);

**Test Case**: should handle API error

* **Objective**: Test error scenarios
* **Test Logic**:

dashboardService.getDashboardData.and.returnValue(throwError(...));

component.ngOnInit();

tick();

expect(component.error).toBeDefined();

**Test Case**: should process dashboard data correctly

* **Objective**: Verify data processing
* **Test Logic**:

expect(component.totalBuses).toBe(42);

expect(component.recentBookings.length).toBe(2);

**4.3 UI Functionality**

**Test Case**: should format currency correctly

* **Objective**: Test currency display
* **Test Logic**:

expect(component.formatCurrency(1200)).toBe('₹1,200');

**Test Case**: should get correct status class/text

* **Objective**: Test status displays
* **Test Logic**:

expect(component.getStatusClass('confirmed')).toBeDefined();

expect(component.getStatusDisplay('COMPLETED')).toBeDefined();

**4.4 Chart Management**

**Test Case**: should initialize charts after view init

* **Objective**: Test chart setup
* **Test Logic**:

spyOn(component, 'initRevenueChart');

component.ngAfterViewInit();

expect(initRevenueChart).toHaveBeenCalled();

**Test Case**: should destroy charts on component destroy

* **Objective**: Test cleanup
* **Test Logic**:

component['revenueChart'] = new MockChart();

component.ngOnDestroy();

expect(MockChart.prototype.destroy).toHaveBeenCalled();

**4.5 Booking Data Processing**

**Test Case**: should handle array of passenger details

* **Objective**: Test multi-passenger handling
* **Test Data**:

passengerDetails: [{name: 'Alice'}, {name: 'Bob'}]

* **Verification**:

expect(booking.passengerName).toBe('Alice, Bob');

**Test Case**: should handle seat data

* **Objective**: Test seat display
* **Test Data**:

seats: ['A1', 'A2']

* **Verification**:

expect(booking.seats).toEqual(['A1', 'A2']);

**5. Test Data Samples**

**Mock Dashboard Data**

typescript

{

totalBuses: 42,

totalRevenue: 1250000,

monthlyBookings: 385,

recentBookings: [

{

id: 'BK001',

passengerDetails: [{ name: 'Sreekesh' }],

seats: ['U8'],

fare: 1200

}

]

}

**6. Code Coverage Analysis**

| **Area** | **Coverage** |
| --- | --- |
| Component Initialization | 100% |
| Data Fetching | 100% |
| Error Handling | 100% |
| Data Processing | 100% |
| Chart Management | 100% |
| Booking Processing | 100% |

**8. Conclusion**

All test cases for DashboardComponent pass successfully. The component handles:

* Data loading and processing
* Error scenarios
* Dynamic chart management
* Complex booking data formatting

**✅ Test Suite Status: PASSED**