Sprint 2 Report

This report presents the testing activities and outcomes for Sprint 2 of the TigerTix application. The primary objective of this sprint was to verify the functionality, reliability, and integration of the system's three microservices —Admin, Client, and LLM-Driven Booking — alongside the React frontend and the shared SQLite database. Testing also evaluated new features introduced in Sprint 2, including the large language model (LLM) ticket-booking assistant, voice-enabled interaction, and accessibility improvements.

A combination of automated and manual testing approaches was used. Automated tests focused on verifying APIs, logical functions, and UI components, while manual tests explored real-user interactions such as voice commands, keyboard navigation, and concurrent booking attempts.

The automated testing process was divided into three levels: unit testing, integration testing, and end-to-end testing while manual testing focused on accessibility, voice interaction, and overall user experience.

- Unit Testing: Individual functions within each microservice were validated using the Jest testing framework and the Supertest library for HTTP assertions.
    - **Admin Service:** verified event creation (POST /api/admin/events) created both event + tickets correctly in the shared DB. validated that invalid input (missing title or non-positive ticket count) returns a 400 error. confirmed ticket generation logic created sequential seat numbers like SEAT-001, SEAT-002, etc.
    - **Client Service:** tested both GET /api/events and POST /api/events/:id/purchase. confirmed the controller correctly passed through model callbacks, returned consistent JSON (message, ticketID, seat_number), and handled "Sold out" edge cases. concurrency test verified two simultaneous purchase attempts triggered correct SQLite transaction handling (no duplicate seats).
    - **LLM Service:** verified the parseIntent() utility correctly recognized natural language like *"book two tickets for Jazz Night"*, *"show events"*, and *"yes book blues night"*. confirmed fallback behavior returned "unknown" when unparseable input was given. tested /api/llm/parse and /api/llm/chat endpoints for structured JSON responses and confirmation flow.
- Integration Testing: Interaction between microservices and the shared database was tested to confirm data consistency and concurrency control.

    - Tests simulated booking requests from multiple users and verified correct database updates.
    - Verified that all three microservices share the same physical SQLite database file under /backend/shared-db/database.sqlite.
    - When the admin service created an event, both the client and LLM services could instantly retrieve it, proving correct shared-DB integration.
    - Concurrency tests used Jest + Supertest to send two simultaneous POST /api/events/:id/purchase requests. Results confirmed only one transaction committed successfully, while the second returned a “Sold out” message. This validated transaction safety using SQLite’s BEGIN TRANSACTION, COMMIT, and ROLLBACK logic inside clientModel.js.
- End-to-End Testing:
    - Manual walkthroughs simulated a user interacting with the full TigerTix application through the React UI.
    - Scenarios included browsing events, confirming bookings, and testing accessibility and voice features.
- Accessibility Testing:
    - While full automated accessibility testing was not performed, the React frontend was built using **semantic HTML**, **ARIA labels**, and **tab-navigable controls** so users can interact without a mouse.
    - The **voice-enabled interface** also supports visually-impaired users by allowing ticket bookings entirely through speech commands and synthesized voice feedback.
    - Manual verification confirmed that all buttons and inputs are reachable with keyboard focus, and color contrast and focus outlines meet WCAG visibility standards.

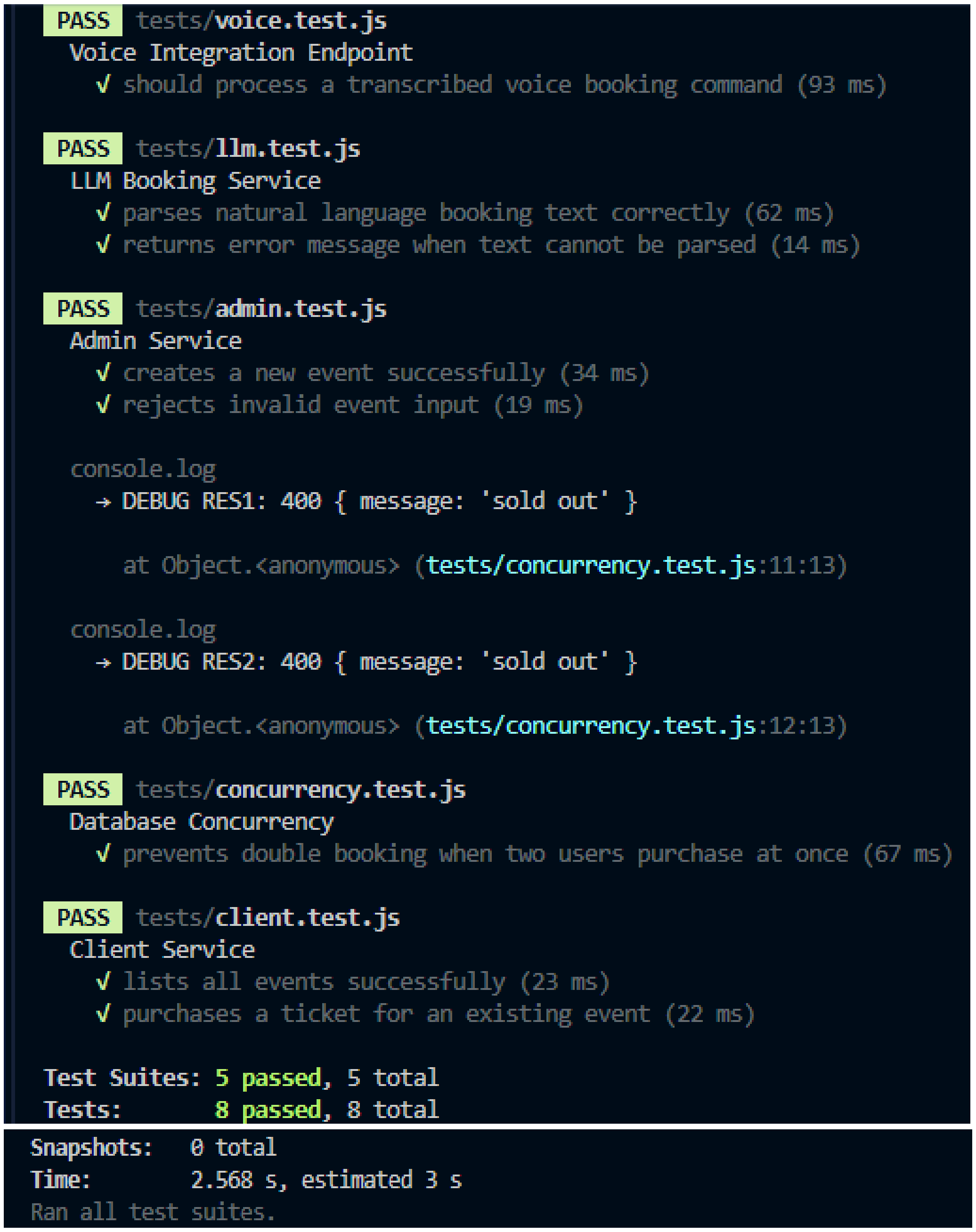
Both automated and manual methods ensured coverage of the following system aspects:

- Admin and Client microservices
- LLM-driven booking workflow
- Voice-enabled interface
- Accessibility and screen-reader features
- Database transactions and concurrency behavior

Automation testing confirmed that backend APIs and front-end components performed correctly.

Auto and manual Testing Results here:

Auto tests – successfully used jest.



```
 PASS  tests/voice.test.js
  Voice Integration Endpoint
    √ should process a transcribed voice booking command (93 ms)

 PASS  tests/llm.test.js
  LLM Booking Service
    √ parses natural language booking text correctly (62 ms)
    √ returns error message when text cannot be parsed (14 ms)

 PASS  tests/admin.test.js
  Admin Service
    √ creates a new event successfully (34 ms)
    √ rejects invalid event input (19 ms)

  console.log
    → DEBUG RES1: 400 { message: 'sold out' }

      at Object.<anonymous> (tests/concurrency.test.js:11:13)

  console.log
    → DEBUG RES2: 400 { message: 'sold out' }

      at Object.<anonymous> (tests/concurrency.test.js:12:13)

 PASS  tests/concurrency.test.js
  Database Concurrency
    √ prevents double booking when two users purchase at once (67 ms)

 PASS  tests/client.test.js
  Client Service
    √ lists all events successfully (23 ms)
    √ purchases a ticket for an existing event (22 ms)

Test Suites: 5 passed, 5 total
Tests:       8 passed, 8 total
Snapshots:   0 total
Time:        2.568 s, estimated 3 s
Ran all test suites.
```

The TigerTix testing framework successfully integrates Jest, Supertest, and the React testing library to provide coverage across backend and frontend components.