Sprint 2 Written Report

Testing Strategy

Objective: Ensure TigerTix system is robust across all components, which includes the backend, frontend, LLM based booking, voice interaction, and accessibility. Jest used for frontend and backend LLM testing.

Approach:
The testing strategy combines **unit tests**, **integration tests**, and **end-to-end tests** to ensure the TigerTix system, including microservices, frontend, LLM features, and accessibility, functions correctly.

Test Coverage:

Admin Microservice	Unit + Integration	Create, update, delete events; retrieve event lists; validate database transactions
Client Microservice	Unit + Integration	Ticket purchase, prepareBooking, confirmBooking, check ticket decrement, handle insufficient tickets
LLM-Driven Booking	Unit + Integration	Parsing user text input, returning event name, ticket count, intent; fallback to keyword parsing
Voice-Enabled Interface	Manual + Integration	Simulate speech-to-text input for booking; verify proper LLM parsing and booking flow
Accessibility Features	Manual	Keyboard navigation, screen reader announcements, focus order, ARIA attributes
Database Transactions & Concurrency	Unit + Manual	Simulate concurrent bookings to check ticket counts, avoid overselling

Backend Tests: Automated testing done with Jest

- **Unit Tests:**
  - clientModel - Event creation and retrieval, ticket availability
  - clientController - API requests and model functions as well as returned JSON and error handling
  - bookingController - booking preparation and confirmation
    - prepareBooking function
    - confirmBooking function
  - llmController- verify llm parsing and fallback
    - parseInput test
    - Fallback test

Frontend Tests: Automated testing done with Jest

- **Unit Tests:**
    - Fetch and display events
    - buyTicket triggers API call and updates status
    - Handles fetch errors gracefully
    - Disable buttons when tickets are 0
- **Integration Tests:**
    - Booking workflow (LLM to backend to frontend)
    - Overbooking Scenario (shows error)
    - Backend/network failure (shows error)
    - Concurrent Booking (ensures counts are respected)

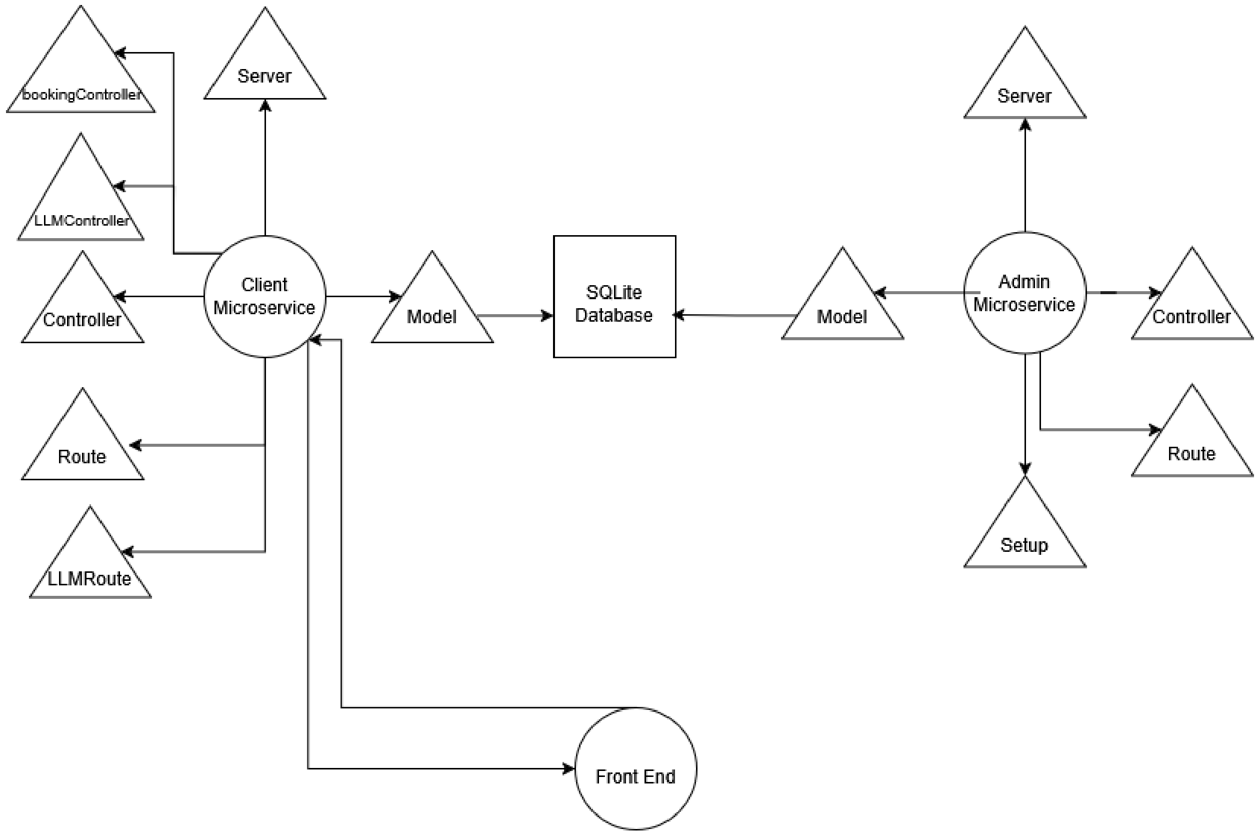
Manual Testing Scenarios

1. **Text Booking via LLM**
    - Input: “Book 2 tickets for Clemson Coding Workshop”
    - Verify: Correct pending UI, correct ticket decrement after confirmation
2. **Voice Booking**
    - Input: Use Microphone text to speech in LLM parse
    - Verify: Booking works same as text input
3. **Accessibility**
    - Navigate with keyboard only
    - Expected: focus order logical, screen reader reads ticket availability and status messages
4. **Concurrent Bookings**
    - Open multiple sessions attempting to book same tickets
    - Expected: Database forces transaction safety, prevents overselling

TEST SUITE	TEST CASE	EXPECTED RESULT	ACTUAL RESULT	CASE NOTES
App.test.js	Fetch/Display Events	Events show on page with correct info	PASS	
App.test.js	buyTicket triggers API and updates status	Message shows success, tickets decrement	PASS	
App.test.js	Disable buy button when tickets = 0	Button disabled	PASS	Edge case: UI correctly handles tickets = 0
llmIntegration.test.js	LLM request books ticket successfully	Status shows success, tickets decrement	FAIL	
llmIntegration.test.js	Concurrent LLM bookings respect ticket limits	Only 1 booking succeeds, tickets decrement correctly	FAIL	
llmIntegration.test.js	Full LLM, backend, frontend flow	LLM Confirmed booking; tickets decrement	FAIL	

llmIntegration.test.js	Overbooking scenario	Error, tickets unchanged	To be implemented	
llmIntegration.test.js	Backend/Network failure	Status shows error message	To be implemented	
Manual/Accessibility	Keyboard Nav	Logical, can active buttons via enter/space	Verified Manually	
Manual/Accessibility	Screen Reader Announcements	Ticket availability and status announced	Verified Manually	
Manual/Voice Booking	Voice to LLM booking	Correct event and ticket booking	Verified Manually	Misheard event names can fail
Database Concurrency	Multiple sessions booking same tickets	No overselling, correct decrement	Verified Manually	High concurrency can expose race conditions if not atomic
Booking Controller	prepareBooking sets pendingBooking, returns JSON	Prepare booking function sends us to pending and returns the proper JSON script	Pass	
Booking Controller	prepareBooking returns 400 if fields are missing	Improper request in prepare booking will yield a 400 error	Pass	
BookingController	confirmBooking works if pendingBooking matches request	Compares requests, checks confirmation	Pass	
BookingController	confirmBooking returns 400 if no pendingBooking	If there isn't a booking pending, trying to confirm will throw a 400 error	Pass	
Client Controller	listEvents returns JSON Array	List events returns a list of events in proper format	Pass	
Client Controller	buyTicket returns success or error JSON	Command returns a success or an error in proper conditions	Pass	
Client Model	Purchasing decrements count or throws error at 0	Count decrements, throws error at 0	Pass	
LLMController	Returns 400 if text missing	Empty request to parser returns error	Pass	
LLMController	Falls back to keyword parsing if LLM fails	Data properly parsed	Pass	Can fail for oddly phrased sentences

Architecture Diagram



Accessibility Description

- Fully keyboard-navigable UI
- Screen reader support for interactive elements
- Voice Input Capture
- LLM Chat Integration through Voice Input Capture and LLM Parsing

Text to Speech Response

Code Quality

- **Function Documentation & Comments**
  - Function headers used
  - Ex voiceChat.js, lines 3-5
  - In-line comments used
- **Variable Naming & Code Readability**
  - Lines are kept short
  - Consistent capitalization of variables and functions
  - Variable and function names are clear
- **Modularization**
  - Helper functions are used

- Small functions are preferred
- **Error Handling**
  - Inputs are validated
  - Proper error codes are used
  - If incorrect inputs, call is canceled and error is retired
  - Ex. lines 46-51, bookingController.js
- **Consistent Formatting**
  - Consistent formatting is used throughout the code