Updated Integration Plan for OSM Data Structures

1. Modify Node Class Completed

- Converted to record
- Added toCoordinates() method
- Implemented getTag(String key) method

2. Enhance Location Class Completed

- Added optional Node field
- Implemented constructors for both OSM and non-OSM locations
- Added methods to access OSM data when available

3. Update Route Class Completed

- Modified to use Node objects instead of Coordinates
- Updated distance calculation methods

4. Create Graph Class Completed

- Implemented using Node and Way objects
- Added methods for pathfinding and routing

5. Update Existing Classes

- Way Completed (converted to record)
- Coordinates No changes needed

6. Service Layer Updates

- OSMDataService Completed
- MapService Completed
    - Updated to use the new Graph class for routing
    - Implemented methods to convert between Location and Node

7. Data Loading and Processing Completed

- OSMDataService now handles loading and processing efficiently

8. UI and API Updates In Progress

- Update to work with enhanced Location class
- Add functionality to display OSM data when available
- TODO:
    - Implement route display and interaction in UI
    - Enhance search functionality to leverage OSM tags and route calculations

9. Testing Completed

- Updated existing tests for new Location and Node structures
- Added tests for Graph and updated Route classes
- Implemented integration tests for OSM data flow

10. Documentation In Progress

- Update JavaDocs for modified classes
- Create overall architecture documentation explaining OSM integration

11. Performance Optimization In Progress

- Implemented basic spatial indexing for nearest-node queries
- TODO:
    - Implement more robust spatial indexing (e.g., k-d tree or quadtree)
    - Profile and optimize Graph operations

12. Additional Features To Be Considered

- Implement tag-based filtering for locations
- Add support for OSM relations if needed

13. Pathfinding Algorithm Implementation Completed

- Implemented Dijkstra's algorithm in the Graph class
- Optimized for performance and accuracy

14. Route Class Enhancements In Progress

- Added methods to calculate total distance and estimated time
- TODO: Implement turn-by-turn directions generation

15. MapService Integration Completed

- Fully integrated Graph and Route classes into MapService
- Implemented methods to find nearest nodes for non-OSM locations

16. Error Handling and Edge Cases In Progress

- Basic handling for scenarios where no route is found
- TODO:
    - Enhance error handling for unreachable destinations
    - Implement fallback mechanisms for incomplete OSM data

17. Caching and Performance To Be Considered

- Implement caching for frequently requested routes
- Optimize graph traversal for large datasets

18. User Interface for Route Display In Progress

- Basic map interaction implemented with Leaflet.js
- TODO:
    - Create UI components to display calculated routes
    - Implement route-specific interactive features

19. API Endpoints for Routing In Progress

- Implemented endpoints for fetching locations, searching, and finding nearest/within radius
- TODO:
    - Create RESTful endpoints for route calculation
    - Implement request/response formats for routing data

20. Logging and Monitoring In Progress

- Added basic logging for pathfinding operations
- TODO: Implement specific performance monitoring for route calculations

Next Steps

1. Focus on UI and API Updates:
    - Implement route display and interaction in the UI
    - Create API endpoints for route calculation
2. Complete Documentation:
    - Ensure JavaDocs are updated
    - Create the overall architecture documentation
3. Enhance Performance Optimization:
    - Implement a more robust spatial index
    - Profile and optimize Graph operations
4. Address Error Handling and Edge Cases:
    - Implement more comprehensive error handling
    - Consider fallback mechanisms for incomplete data
5. Consider Additional Features:
    - Implement tag-based filtering
    - Evaluate the need for OSM relation support
6. Think about Caching and Performance:
    - Plan for caching frequently used routes
    - Optimize graph traversal for larger datasets