**Technical Approach**

**Technology Stack**

* **Frontend:** React.js, D3.js (or Recharts) for dynamic dashboards, interactive charts, and KPI visualizations.
* **Backend:** Python, Flask/FastAPI for RESTful APIs; Google OR-Tools CP-SAT for optimization; Pandas/NumPy for data processing.
* **Data Handling:** CSV/JSON for timetable and topology management; NetworkX for graph modeling of railway sections.
* **Machine Learning:** scikit-learn for ETA prediction mockups; potential extension to TensorFlow or PyTorch for advanced ML features.
* **Deployment:** Docker for containerization; GitHub for version control; deployment-ready for cloud (AWS/GCP).
* **Integration:** REST APIs enable communication between frontend and backend, with support for external railway data systems.

**Methodology & Process**

* **Stepwise Implementation:**
  1. Generate or import mock train timetable and section topology data for testing.
  2. Develop conflict detection module to automatically identify scheduling/track/platform conflicts.
  3. Implement optimization engine (OR-Tools CP-SAT) to adjust train schedules, reduce delays, and prioritize train types.
  4. Build interactive dashboard presenting real-time visualizations (before/after stringline charts, KPIs).
  5. Add sandbox “what-if” simulation tools to allow train controllers to experiment with real-time schedule modifications.
  6. Integrate a Machine Learning mock for ETA shift predictions.
  7. Package the system as a working prototype, ready for live demonstration.
* **Prototyping:**  
  Flowcharts and data flow images will be used to illustrate the entire scheduling, detection, and optimization pipeline in presentations.