Function Contracts Documentation ----------------- 3ngineer_342

Contract: loadCSVData ()

File: csvLoader.js
Operation: () -> Array<TripSegment>
Purpose:
Synchronously reads data/train_connections.csv and returns parsed rows as TripSegment objects.

Preconditions

- File data/train_connections.csv exists and is readable.
- CSV header contains expected columns.

Postconditions

- Returns an array of TripSegment objects (may be empty).
- Skips rows whose column count does not match header.
- Logs number of rows loaded to console.

Outputs: Array<TripSegment> (strings for CSV fields)

Contract: parseTime(str)

File: csvLoader.js
Operation: (str: string) -> Number
Purpose:
Converts "HH:MM" to minutes since midnight. Returns 0 if invalid or empty.

Preconditions

- str is a string.

Postconditions

- Returns integer minutes (≥ 0), or 0 if input missing/invalid.

Inputs: String
Outputs: Integer

Contract: calculateConnectionDuration(tripSegments)

File: csvLoader.js
Operation: (tripSegments: Array<TripSegment>) -> String
Purpose:
Computes formatted total duration, including travel and wait times.

Preconditions

- tripSegments is non-empty.
- Each segment has valid Departure Time and Arrival Time strings ("HH:MM" or with (+1d)).

Postconditions

- Returns a string representing total duration.

Inputs: Array<TripSegment>
Outputs: String (Time)

Contract: calculateChangeTimes(tripSegments)

File: csvLoader.js
Operation: (tripSegments: Array<TripSegment>) -> Array<String>
Purpose:
Computes each change/wait time between consecutive segments.

Preconditions

- tripSegments.length >= 2

Postconditions

- Returns array of strings of length = segments − 1.
- Direct trips → returns empty array.

Contract: findAllConnections(routes, query)

File: csvLoader.js
Operation: (routes: Array<TripSegment>, query: Query) -> Array<Trip>
Purpose:
Given dataset routes and optional filters, returns all matching trips (direct, 1-stop, or 2-stop).

Preconditions

- routes is an array (may be empty).
- query is an object; missing keys = no filter.

Postconditions

- Returns Array<Trip> matching query fields.
- Each trip includes:
  - firstClassPrice & secondClassPrice as Number
  - duration as String
  - changeTimes as Array<String>

Inputs

routes: Array<TripSegment>

query: {

departure?, arrival?, departureTime?, arrivalTime?,

trainType?, daysOfOperation?,

firstClassPrice?, secondClassPrice?

}

Outputs: Array<Trip>

Contract: searchConnections(req, res)

File: connectionsController.js
Operation: (req: Request, res: Response) -> void
Purpose:
Express GET handler for /api/connections. Reads req.query, calls findAllConnections(), and returns JSON.

Preconditions

- routes initialized at startup via loadCSVData().
- req.query contains optional filters.

Postconditions

- Sends res.json(Array<Trip>) (HTTP 200).

- Returns “Empty” if no matches.

Inputs: req.query (Query)
Outputs: HTTP JSON response (Array<Trip>)

Contract: getAllConnections ()

File: api.js
Operation: () -> Promise<Array<Trip>>
Purpose:
Fetches all connections from backend API (GET /api/connections).

Preconditions

- Backend reachable at API URL.

Postconditions

- Resolves to Array<Trip> or error.

.