Product Requirements Document (PRD)

🚀 Product Overview

Build a graph-based engine integrated with flight search to compute optimal points-based redemption paths for Canadian users. Users input an origin, destination, and date, and the system searches for bookable flights, maps them to loyalty programs, and outputs the most efficient point transfer path based on user preferences (value, time, or hops).

🚧 Problem Statement

Canadian users face two challenges: 1) finding award flight availability on specific dates/routes, and 2) figuring out the optimal point transfer path to book those flights. No unified tool exists to search award space and dynamically compute the most efficient program path.

👥 User Stories

- **As a user, I want to search flights by origin, destination, and date to see available award options.**
- **As a user, I want to see which loyalty programs can book a given flight.**
- **As a user, I want to see the best points transfer path based on value, time, or hops.**
- **As a user, I want to be notified if no bookable route exists through points.**

🔄 Feature Table

Feature	Description
Flight Search Input	UI for users to input origin, destination, and date

Award Availability Layer	**API to search for available award flights**
Flight-to-Program Mapping	**Logic to map flight options to bookable loyalty programs**
Source Program Selector	**UI to pick where user's points start from**
Optimization Mode Selector	**UI for user to choose value, time, or hops**
Pathfinding Engine	**Computes optimal point transfer path using graph logic**
Graph Data Store	**Supabase-backed table of programs and transfer paths**
Transfer Result Display	**Shows path: program hops, conversion ratios, time**
No Route Handling	**Shown if no valid path to any available award exists**
Dropdown Program Metadata	**Adds program type, tooltips, and descriptions to improve selection clarity**

⚖️ Atomic Feature Breakdown

1. Flight Search Input

- **UI Element: Three fields (Origin, Destination, Date)**
- **Logic: Validates inputs, triggers search**
- **Data Source: User input**

2. Award Availability Layer

- **UI Element: Hidden (backend API call)**
- **Logic: Searches award availability via partner API (Seats.aero, AwardLogic, etc.). May include caching to prevent API rate limits and failures.**
- **Data Source: 3rd-party availability APIs**

3. Flight-to-Program Mapping

- **UI Element: None (logic only)**
- **Logic: Maps each flight (e.g. LX73) to programs that can book it (e.g. Aeroplan). Distinguishes between "can theoretically book" and "has award space." Bookable programs filtered post-availability check.**
- **Data Source: Static airline-to-program map (stored in Supabase)**

4. Source Program Selector

- **UI Element: Dropdown (Start Program)**
- **Logic: Filters graph to valid paths from selected start node**
- **Data Source: `programs` table in Supabase**

5. Optimization Mode Selector

- **UI Element: Radio buttons or dropdown ("Best Value", "Fastest Transfer", "Fewest Steps")**
- **Logic: Routes selected mode to correct cost function**
- **Data Source: Local enum / logic map**

6. Pathfinding Engine (Cursor)

- **UI Element: Triggered post flight + program match**
- **Logic: Cursor executes Dijkstra's algorithm using:**
  - **Value mode: edge weight = 1 / ratio**
  - **Time mode: edge weight = transfer_time_hours**
  - **Step mode: edge weight = 1 per edge**
  - **Includes source node, filtered destination programs (based on award space), and selected optimization mode**
- **Data Source: `transfer_paths` table + award-compatible destination programs**

7. Graph Data Store

- **UI Element: Admin UI (future)**
- **Logic: Seeded from static table of Canadian reward partners. Includes optional `valid_until` date for each edge to account for partner devaluations.**
- **Data Source:**
  - **`programs (id, name, type, description, logo_url)`**

    - `transfer_paths (id, from_program_id, to_program_id, ratio, transfer_time_hours, last_verified, valid_until)`

8. Transfer Result Display

- **UI Element: Stepper showing each transfer**
- **Logic: Shows total points required, transfer time, and path taken. Optionally flags slow transfers (>48h).**
- **Data Source: Result from graph search + flight match**

9. No Route Handling

- **UI Element: Display message with specific state:**
    - **(a) "No award flights found for selected date and route"**
    - **(b) "No valid point transfer path available to bookable programs"**
- **Logic: Conditional display based on root cause of failure**
- **Data Source: Combined availability and graph results**

10. Dropdown Program Metadata

- **UI Element: Dropdown enriched with icons, descriptions, and program type labels**
- **Logic: Uses metadata from `programs` table to improve UX**
- **Data Source: `programs (name, type, description, logo_url)`**