

# Live Data API Integration Plan

## 1. API Selection: The Racing API

Based on the research, **The Racing API** is the most suitable candidate for the live data feed integration.

Feature	Requirement	The Racing API (Pro Plan)	Notes
<b>Data Type</b>	Racecards, Results, Odds	Daily advanced racecards, 20+ bookmaker odds, Results	Meets core requirements.
<b>Real-Time</b>	Live race updates	Odds change history from market open	Implies near real-time updates, sufficient for the current "continuous prediction" agent.
<b>Coverage</b>	Global/Australia/NZ (to replace Tab.co.nz)	Full Australia coverage add-on available	The existing system uses Tab.co.nz (New Zealand). The Australian coverage add-on is the closest and most likely to provide the necessary data for the existing models, which appear to be trained on data from that region.
<b>Historical Data</b>	Required for model feature engineering	Historical results with advanced queries	Essential for features like <code>days_since_last_race</code> , <code>PREV_RACE_WON</code> , etc., as noted in <code>continuousPredictionAgent.ts</code> .

<b>Cost</b>	Must be evaluated	£99.99/month + £49.99/month (Australia add-on)	The cost is acceptable for a commercial MVP.
-------------	-------------------	--	--

## 2. Integration Architecture

The existing system uses a placeholder `tabDataService`. The new architecture will introduce a dedicated service to handle communication with The Racing API.

1. **New Service:** Create a new TypeScript file, e.g., `/src/services/racingApiDataService.ts`.
2. **Configuration:** The API key and base URL will be stored in environment variables and accessed by the new service.

### 3. Data Fetching Functions:

- `getTodayRacecards()` : Fetches the schedule and race details for the current day. This will replace the existing `getTodaySchedule()` from the placeholder `tabDataService`.
- `getLiveOdds(raceId)` : Fetches the latest odds for a specific race.
- `getRaceResult(raceId)` : Fetches the final result for a completed race.
- `getHorseHistory(horseId)` : Fetches historical performance data for feature engineering.

### 4. Integration Points:

- **continuousPredictionAgent.ts :**
  - Update `import { getTodaySchedule } from "../services/tabDataService";` to use the new service's function.
  - Update `prepareFeatures()` to use the new service to fetch real-time odds (`IMPLIED_PROBABILITY`, `NORMALIZED_VOLUME`) and historical data (`days_since_last_race`, `PREV_RACE WON`, `WIN_STREAK`).
- **resultCollector.ts :**
  - Update `import { getTodaySchedule } from "../services/tabDataService";` to use the new service's function to fetch race results.

## 3. Implementation Steps (Phase 4)

1. Create the `/src/services` directory.
2. Create `/src/services/racingApiDataService.ts` with placeholder functions and API key handling.

3. Modify `continuousPredictionAgent.ts` to use the new service and implement the feature fetching logic.
4. Modify `resultCollector.ts` to use the new service for result collection.
5. **Crucially:** Since the existing code is in the root of `/home/ubuntu/upload`, I will assume the project structure is a standard Node.js/TypeScript project and create the necessary directories for the new service file. I will first move the existing files into a project directory.

**Project Directory Setup:** I will create a project directory named `equine_oracle_project` and move the relevant files into it to establish a proper structure before coding.

- `equine_oracle_project/src/agents/continuousPredictionAgent.ts`
- `equine_oracle_project/src/agents/resultCollector.ts`
- `equine_oracle_project/src/services/tabDataService.ts` (placeholder for now)
- `equine_oracle_project/src/services/racingApiDataService.ts` (new file)
- `equine_oracle_project/package.json` (for dependencies)
- `equine_oracle_project/tsconfig.json` (for TypeScript config)
- `equine_oracle_project/ensemble_prediction_system_large.py` (ML model)
- `equine_oracle_project/Analytics.tsx` (Frontend)
- `equine_oracle_project/csv_upload.ts` (Data utility)
- `equine_oracle_project/todo.pdf` (Documentation)
- `equine_oracle_project/Equine_Oracle_MVP__ML_Integration_and_Admin_Setup_.pdf` (Documentation)