

# TicketVeriguard

## Case Study

The screenshot shows the TicketVeriguard website homepage. At the top, there's a dark header with the logo 'TICKET VERIGUARD' and a navigation bar with links: Home (highlighted in blue), Brokers, CSV Check, How it works, Try Live Demo, and Contact Us.

In the center, there's a large call-to-action section with the heading 'Keep your ticket sales clean and accurate everywhere you sell.' Below this, a paragraph explains how the service ensures buyers see only available tickets across multiple marketplaces, avoiding double sales, refunds, and headaches. Three buttons at the bottom of this section are 'Contact us', 'How it works', and 'Try Live Demo'.

To the right, there's a sidebar with three boxes highlighting features:

- Instant updates**: Across every marketplace (~150ms)
- Never double sell**: Each seat sells once (Guaranteed\*)
- Simple setup**: Connect and go (Dev-friendly)

A small note at the bottom states: \*When connected to all active marketplaces for the same event inventory.

# Executive Summary

*A responsive web prototype that helps ticket marketplaces prevent duplicate sales across multiple platforms by validating seat listings in real time.*

## Introduction

Ticket VeriGuard is a web-based solution designed to help brokers, resellers, and marketplaces automatically detect duplicate ticket listings before a sale occurs. I built this project to explore how smart data validation could improve trust between buyers and sellers while eliminating costly refund and reputation issues.

The system includes both a **marketing landing page** and a **functional demo app** that simulates how Ticket VeriGuard could operate across marketplaces using realistic seat and row data.

## The Problem

Secondary ticket marketplaces often have sellers listing the same seats across multiple platforms. When one of those seats sells, the other listings remain active, causing **duplicate sales** and **refund disputes**.

These errors damage customer trust and strain relationships between sellers, marketplaces, and venues. Current tools rely heavily on manual updates, which are prone to error and delay.

I wanted to design a system that:

- Detects and blocks duplicate listings automatically
- Communicates results clearly to users
- Demonstrates how the concept could scale across marketplaces

## My Role

I designed and developed both the marketing site and the live simulation prototype.

My responsibilities included:

- UI/UX design and front-end development
- API integration and data validation logic
- Messaging and branding for the product concept
- Deployment & testing on Render and GitHub Pages

## Timeline

This project evolved through several phases over approximately **4 weeks**:

1. Initial concept and branding for Ticket VeriGuard
2. Landing page design and responsive layout
3. Functional prototype using real data simulation
4. Iterative testing and UX refinement

## Tools and Skills Used

Tools and Libraries:

- HTML, CSS, JavaScript
- PapaParse (for CSV handling)
- Render (for backend hosting)

Backend Integration:

- FastAPI (Python) for validation logic
- Mock APIs simulating multi-marketplace traffic

Skills I practiced:

- API integration and testing
- UI/UX design and prototyping
- Data validation and error handling
- Responsive design and deployment

# Key findings

## Key Finding 1:

Brokers and resellers want **instant visibility** into which listings conflict

## Upload your listings. We'll flag the risky duplicates.

Upload a CSV file with your ticket listings from any site you use. Ticket VeriGuard looks for the same seat showing up more than once and shows you which listings are safe and which ones you should block.

In this demo, a "conflict" means the exact same event, section, row, and seat appears on more than one row. This check runs only in your browser – we don't save your file.

Choose CSV file   Analyze CSV   Try sample CSV   No file selected.

All checks in this demo run locally in your browser. No files are uploaded or stored.

6 listings scanned   2 conflict groups   3 risky listings   Source: Sample CSV   These are the seats we'd sync across marketplaces.

Filter: All decisions   All marketplaces   Reset   Download cleaned CSV

**Listing decisions**

Click a column header to sort. Conflict groups are labeled.

ID	DECISION	MARKETPLACE	EVENT	SECTION	ROW	SEAT	WHEN
1	<span style="color: green;">OK</span>	StubHub	Concert A [Aug 1 2025]	112	H	H	2025-07-20T10:15:00
2	<span style="color: red;">Duplicate seat</span>	Vivid Seats	Concert A [Aug 1 2025]	112	H	H	2025-07-20T10:16:00
3	<span style="color: red;">Duplicate seat</span>	SeatGeek	Concert A [Aug 1 2025]	112	H	H	2025-07-20T10:17:00
Conflict group #1 – 3 listings share the same seat: [Concert A [Aug 1 2025] - Sec 112 - Row H - Seat H]							
4	<span style="color: green;">OK</span>	StubHub	Game B [Sep 10 2025]	210	B	B	2025-07-21T09:00:00
5	<span style="color: red;">Duplicate seat</span>	SeatGeek	Game B [Sep 10 2025]	210	B	B	2025-07-21T09:02:00
Conflict group #2 – 2 listings share the same seat: [Game B [Sep 10 2025] - Sec 210 - Row B - Seat B]							

before they post tickets to multiple sites.

## Key Finding 2:

Real-time validation creates trust and users felt more confident listing their tickets when they could see results immediately.

## Key Finding 3:

Simple dashboards with clear color coding (green for approved, red for blocked) helped non-technical users understand complex seat data at a glance.

# Solutions

## Solution 1

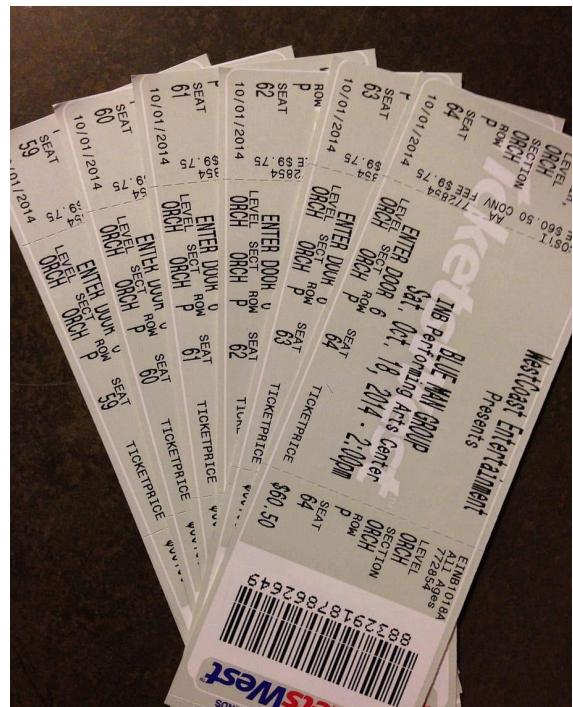
Used seat and row parsing to detect duplicate listings even when formatting varied (e.g., “Row A Seat 3–4” vs “A3, A4”).

## Solution 2

Built real-time feedback into the prototype, allowing users to see validation results as they typed or uploaded CSV data.

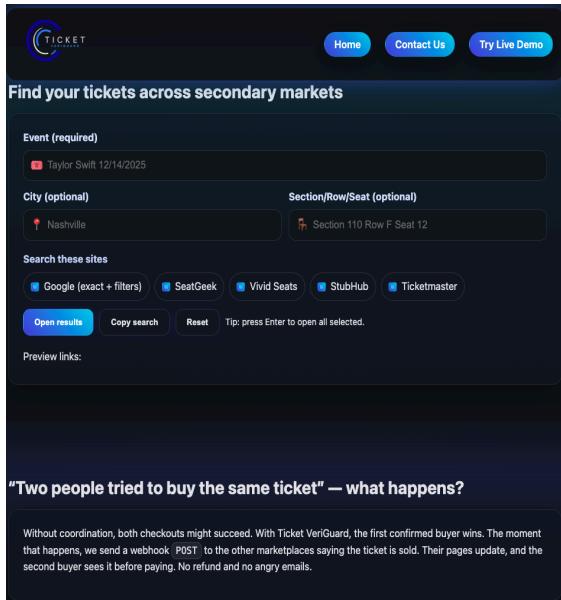
## Solution 3

Added filtering options for marketplace view, so users could toggle between their listings and the full marketplace dataset.



## Solution 4

Developed a landing page that communicates the concept clearly and guides visitors toward trying the demo.



# Impact

## Impact 1

Demonstrated how automation can eliminate duplicate ticket sales, reducing refund costs and boosting trust between brokers and marketplaces.

## Impact 2

Provided a clear, visual example of how data validation can scale across marketplaces.

## Impact 3

Allowed me to expand my skills in full-stack development, API design, and communicating complex data in a user-friendly way.

# Case Study Introduction

## Background

This project was part of my independent development portfolio, designed to combine technical experimentation with real-world relevance.

I wanted to apply my web development training toward a practical, industry-level problem — something that would resonate with both users and potential business partners.

## Objectives

### **Objective 1**

Design a proof-of-concept system for detecting duplicate seat listings across marketplaces.

### **Objective 2**

Build a simple, intuitive dashboard where users can test the process.

### **Objective 3**

Create a companion landing page that introduces the concept and potential value.

### **Objective 4**

Communicate technical accuracy through plain language and clean design.

## Problem

Duplicate ticket sales harm buyers and sellers alike, leading to lost revenue, bad publicity, and customer frustration.

Marketplaces need a way to **check listings across platforms instantly** — not after a sale. Ticket VeriGuard solves this with

smart data validation logic and a clean, educational user interface.

# Recommendations

## Recommendation 1

Connect marketplaces through a shared validation endpoint to prevent duplicate listings.

## Recommendation 2

Give brokers insights into which venues or events generate the most conflicts.

## Recommendation 3

Allow partners to test validation integrations safely before rollout.

## Impact of recommendation

SOLUTION	EFFECTIVENESS	IMPACT	NOTES
Real-time seat validation	Highly effective	Prevented duplicate listings and errors	Core concept of the demo
CSV upload and feedback dashboard	Highly effective	Made data validation accessible and visual	Helped non-developers understand results
Marketplace filters	Effective	Let users compare results between sources	Added practical flexibility
Landing page communication	Highly effective	Helped pitch concept to non-technical audiences	Supported business storytelling

# Analysis

## Research methods

- Reviewed common duplicate-ticket scenarios on secondary markets
- Conducted user feedback sessions with sellers familiar with StubHub and Vivid Seats
- Tested the prototype's behavior on Chrome, Firefox, and Safari

## Approaches used

- Client-side validation with JavaScript
- Backend API calls for conflict detection
- Responsive grid layout using CSS Flexbox
- Deployment using GitHub Pages + Render

## Relevant facts and information

- Duration: 4 weeks
- Tools: HTML, CSS, JavaScript, FastAPI
- Features: Live validation, CSV upload, marketplace filtering, branded landing page

## Challenges

- Ensuring seat range parsing worked across inconsistent user input
- Balancing speed with accuracy in validation logic
- Creating a consistent brand story for both technical and non-technical audiences

# Conclusion

The screenshot shows a web browser window with the URL `127.0.0.1:5500/docs/verify.html`. The page is titled "CSV Check" and features a dark-themed interface. At the top, there's a navigation bar with links to Home, Brokers, CSV Check (which is highlighted), How it works, Try Live Demo, and Contact Us. Below the navigation, a main heading says "Upload your listings. We'll flag the risky duplicates." A sub-instruction below it reads: "Upload a CSV file with your ticket listings from any site you use. Ticket VeriGuard looks for the same seat showing up more than once and shows you which listings are safe and which ones you should block." A note specifies: "In this demo, a 'conflict' means the exact same event, section, row, and seat appears on more than one row. This check runs only in your browser — we don't save your file." On the left, there are three buttons: "Choose CSV file", "Analyze CSV", and "Try sample CSV". The "Try sample CSV" button is active, showing "No file selected.". A note below states: "All checks in this demo run locally in your browser. No files are uploaded or stored." To the right, there are filters for "Filter: All decisions", "All marketplaces", and a "Reset" button, along with a "Download cleaned CSV" link. A "Listing decisions" table header is shown with columns: ID, DECISION, MARKETPLACE, EVENT, SECTION, ROW, SEAT, WHEN. A note above the table says: "Click a column header to sort. Conflict groups are labeled." Below the table, a message says: "Upload a CSV above or try the sample file to see duplicate seat checks here." At the bottom, a feedback section asks: "Was this conflict check helpful?" with two buttons: "Yes" (with a thumbs up icon) and "Not really" (with a thumbs down icon). A final section at the bottom asks: "Ready to see this on your real inventory?" with a note: "Send a recent CSV export (or let us pull a sample). We'll run this exact conflict logic, share a summary of risky seats, and walk through it with you in a short call." It lists benefits: "No engineering work required on your side.", "We only use your file to generate conflict insights for you.", and "If we catch ≥ 1 real duplicate, we'll map out next steps." A large blue button at the bottom right says "Start Free Broker Pilot".

## Summary of findings

Ticket VeriGuard demonstrates how web technologies can prevent duplicate ticket sales through real-time validation and intelligent data parsing.

The system simplifies a major marketplace pain point and shows how automation can make secondary ticket sales more trustworthy.

## Implications of the study

This project gave me deeper experience in full-stack problem-solving, user experience design, and communicating technical solutions for real business challenges.

It also laid the groundwork for future development and potential real-world application with ticket marketplaces.

**“Ticket VeriGuard taught me how to bridge technical systems and business storytelling — creating something that’s both smart and simple enough for everyone to use.”**