Mara Rüsen | ESADE Business School | Barcelona | 2026

DANCE FINDER

Discover dance classes in Barcelona - powered by AI

Live App
dance-finder-ax2tbcexhyhjqsjij2exkw.streamlit.app

GitHub Repository
github.com/MaraRusen/dance-finder

1. The Idea & Value Proposition

The starting point was a real personal problem: as a dancer in Barcelona, finding drop-in dance classes every day means scrolling through dozens of Instagram stories, checking multiple studio websites, and still not knowing if a class is full or who the teacher actually is. There is no central platform for this.

Dance Finder solves this by aggregating all dance classes in Barcelona into one interface — showing who is teaching, where, when, at what price, and how full the class is expected to be. The target users are recreational dancers, dance tourists, and anyone who wants to discover new teachers and styles in the city.

2. How the Prototype Was Built

Step 1 - Designing the Data Model

The first step was designing the data schema. Rather than starting with a given dataset, the classes.json file was created from scratch to simulate the exact output of a Vision LLM pipeline. Each class object contains: style, studio, teacher profile (name, origin, bio, Instagram, today's topic), time, price, level, GPS coordinates, and a crowd_score between 0 and 1.

Step 2 - Building the Streamlit App (app.py)

The app was built in layers, starting from the visual design and working inward to the data logic:

- Global CSS injected via st.markdown to create a dark purple/blue theme, Bebas Neue typography, and frosted-glass card effects
- Hero section with a full-width background video (base64 encoded so it loads reliably on Streamlit Cloud)
- Filter bar: all 5 filters on one horizontal row using st.multiselect (style, teacher, level), st.date_input, and st.toggle (free only) - all widgets not covered in class
- 4 dynamic metric boxes (st.metric) showing live KPIs: classes today, free trials, selling fast, styles available
- Two-column layout: interactive Folium map on the left (color-coded pins by crowd level) and custom HTML/CSS ticket cards on the right
- Each ticket card shows style badge, level badge, crowd % badge, teacher name, location, time, price, and two expandable sections: teacher bio and original IG story viewer

Step 3 - Building the Pipeline Page (pages/about.py)

A second Streamlit page was created to explain the full AI pipeline. It covers the 3 pillars of the prototype (UX, pipeline, accuracy), the step-by-step data flow diagram, both planned AI models, and the full tech stack. The page matches the design system of the main app.

Step 4 - Deployment

- GitHub repository created: github.com/MaraRusen/dance-finder (public)
- Streamlit Cloud connected to GitHub for automatic deployment on every git push
- Live URL: dance-finder-ax2tbcexhyhjqsjij2exkw.streamlit.app

3. The AI Pipeline (Architecture)

The pipeline is split into two phases to keep the app fast at runtime. No API calls happen when a user loads the page.

Offline Phase - runs daily via cronjob

Step 1: scraper.py downloads Instagram story screenshots from Barcelona dance teacher accounts and scrapes studio websites (BeautifulSoup + instaloader). Step 2: extractor.py sends each image to the GPT-4o Vision API with a structured prompt - it returns JSON with time, teacher, style, price, and level. Step 3: Output is saved to classes.json (today's view) and appended to classes_history.csv (for model training). Step 4: model.py trains a Random Forest on the history and saves model.pkl.

Runtime Phase - the Streamlit app

app.py loads classes.json and renders the map, ticket cards, and filters instantly. The crowd_score field from the JSON is displayed as a badge on each card. No scraping or API calls happen at runtime - everything is pre-processed.

In this prototype, classes.json is a mock dataset that simulates the exact output format of the full pipeline. The crowd_score values simulate the Random Forest model output. The architecture is fully production-ready - only the data source is mocked.

4. Planned AI Models

Model	Purpose	Status
Vision LLM (GPT-4o)	Reads IG story screenshots → extracts structured JSON (time, teacher, style, price, level)	Planned — simulated in prototype
Crowd Predictor (Random Forest)	Predicts fill rate per class based on weekday, teacher, style, hour, weather history	Planned — simulated via mock crowd_score
Class Recommender (Collab. Filtering)	Recommends new classes based on user history — like Spotify Discover Weekly for dance	Future feature

5. How Claude (AI) Was Used to Build This

Claude by Anthropic was used extensively throughout the entire development process - not just for code generation, but as a true collaborative thinking partner at every step.

Stage	How Claude Was Used
Concept & ideation	Discussed the problem space, refined the value proposition, and chose Dance Finder as an original use case instead of the standard dataset
Data design	Designed the classes.json schema together, including all fields needed to simulate the Vision LLM output format
app.py development	Claude wrote the full Streamlit app: CSS theme, hero section, filter logic, Folium map integration, and all custom HTML/CSS ticket card components
about.py development	Claude wrote the pipeline explanation page with the full architecture diagram and tech stack

Debugging	Every error (ModuleNotFoundError, file path issues, Git permission errors) was solved by pasting the error into Claude and following the fix
GitHub setup	Claude guided the full Git workflow: init, .gitignore, commit, remote setup, and push with Personal Access Token authentication
Deployment	Claude diagnosed the Streamlit Cloud ModuleNotFoundError and fixed the requirements.txt
Documentation	Claude wrote the README.md and this Word document

Key Insight on Using AI for Prototyping

The most valuable use of Claude was not just generating code - it was the ability to iterate extremely fast. When a design decision needed to change (e.g. switching from red to purple/blue, removing the sidebar, adding the video), Claude rewrote the entire relevant section in seconds. This reduced the design-to-implementation cycle from hours to minutes, allowing much more time to focus on the product concept and user experience.

6. Tech Stack & Widgets

Category	Tool
Frontend framework	Streamlit
Interactive map	Folium + streamlit-folium
Custom UI	HTML/CSS injected via st.markdown
New widgets (not in class)	st.multiselect, st.toggle, st.date_input, st.metric, st.expander
AI extraction (planned)	GPT-4o Vision API
ML model (planned)	scikit-learn Random Forest / XGBoost
Deployment	Streamlit Cloud (automatic from GitHub)
Version control	GitHub (public repository)

7. Conclusion

Dance Finder addresses all three pillars of the PDAI prototype framework: a polished, editorial UX inspired by real Barcelona dance studios; a fully architected two-phase AI pipeline (offline extraction + runtime display); and a transparency-first design where every data point is traceable to its original Instagram story source.

The prototype demonstrates that a genuinely useful AI product can be built rapidly with the right tools and a clear user problem. The combination of Streamlit for the frontend, a Vision LLM for data extraction, and a predictive model for crowd levels creates a product that could realistically be deployed and used by Barcelona dancers today.

What makes this prototype stand out

1. Original use case - not a provided dataset, but a real personal problem | 2. Custom HTML/CSS widget system - not default Streamlit components | 3. Offline/runtime pipeline separation - production-ready architecture | 4. AI transparency built into the UX - users can always verify the source | 5. Fully deployed with a live public URL accessible to anyone. Very happy with it 😊