Meeting Brief — Travel Info Final Project

Short guide for the meeting (1 hour): summary, demo steps and short answers to expected questions.

1) Quick (30–60s) — What we have

- Full-stack MVP: Backend (Node.js + TypeScript + Express + Prisma) and Frontend (React + TypeScript + Vite).
- Data model: `Destination`, `Tour`, `Booking` (Prisma + SQLite dev DB: backend/prisma/dev.db, seed data included).
- Dev setup: Backend runs at http://localhost:4000; Frontend served by Vite (Local URL in terminal, typically http://localhost:5173 or 5175). Vite dev proxy forwards /api to the backend.
- Repo: top-level helper files present: .env.example, docker-compose.yml, LICENSE, .gitignore, and simple Dockerfiles for backend and frontend.

2) Demo flow (what to show)

1. Start locally (quick):

- Backend: cd backend && npm install && npm run dev
- Frontend (new terminal): cd frontend && npm install && npm run dev

2. Open browser: Local URL from the Vite terminal (e.g. http://localhost:5175/).

3. In the browser show:

- Home — short project intro.
- Destinations — list of destinations (loaded via GET /api/destinations).
- TourDetails — tour detail view; show the booking form and create a test booking (POST /api/bookings).

4. Optionally: show terminal/logs (e.g. Vite log or backend log) to demonstrate incoming requests.

3) Important commands (for questions or quick debugging)

- Check backend response: curl -sS http://localhost:4000/api/destinations | jq
- Check through Vite proxy: curl -sS http://localhost:5175/api/destinations | jq
- Docker Compose (if used): docker compose up --build

Port conflict (common question): If docker compose fails with "address already in use" for port 5173, a Vite server is already running. Stop it or map a different host port in docker-compose.yml.

4) Short answers to expected questions

- Who can see the code? — The GitHub repo is currently public; specific collaborators (Abdullah-Jlilati, Maik-Protze) have admin rights.
- What technologies? — Backend: Node.js, TypeScript, Express, Prisma (SQLite). Frontend: React, TypeScript, Vite, React Router.
- What endpoints? — e.g. GET /api/destinations, GET /api/tours/:id, POST /api/bookings, GET /api/bookings.
- Is it production ready? — Not yet: Dockerfiles are development-oriented; tests, production build/deployment and secret management are missing.

5) Next steps (short proposals)

- Short term (1–2 days): expand README, add .dockerignore, create production Dockerfiles (build → serve), add simple integration tests for booking flow.
- Mid term: extend CI/CD (deploy stage), deploy to a hosting provider (Azure/Heroku), improve UX and validation/error handling.

6) One-liner to say in the meeting

"We have built a Full-Stack MVP with an Express/Prisma API and a React/Vite UI. Right now we can show the destinations list and create a test booking live — next steps are production images, tests and a short deployment README."

File: MEETING_BRIEF_en.md (repo root).