TravelsTREM — Architecture Diagram & One-Page Reference

> Single-page architecture overview you can share with devs/stakeholders. Contains component diagram, sequence flow for authentication, deployment notes, endpoints, folder structure, and an implementation checklist.

---

1. High-level component diagram (text + mermaid)

```
flowchart LR
  subgraph Frontend
    A[React App
    (pages: /login /register /profile /tours)]
  end

  subgraph CDN & Hosting
    CDN[Netlify / Vercel]
  end

  subgraph Backend
    B[Node.js + Express]
    B --> M[MongoDB Atlas]
    B --> Auth[Auth Routes]
    B --> Tours[Tours & Services Routes]
  end

  A -->|HTTPS REST| CDN -->|HTTPS| B
  A -- "Bearer JWT" --> B
  B -- "DB requests" --> M

  note right of A: localStorage (access token)
  note right of B: httpOnly cookie (refresh token)
```

---

2. Authentication sequence (JWT + Refresh token)

```
sequenceDiagram
  participant U as User (Browser)
  participant FE as React Frontend
  participant BE as Express Backend
  participant DB as MongoDB

  U->>FE: Submit credentials (email/password)
  FE->>BE: POST /api/auth/login {email,password}
  BE->>DB: find user by email
  DB-->>BE: user record
  BE->>BE: verify password
  alt valid
    BE->>BE: sign accessToken (short-lived)
    BE->>BE: sign refreshToken (longer, httpOnly cookie)
    BE-->>FE: {accessToken}
    Note right of FE: store accessToken in memory/Redux or localStorage
    BE-->>U: set-cookie: refreshToken (httpOnly)
  else invalid
    BE-->>FE: 401 Unauthorized
  end

  Note over FE: For protected API calls, include Authorization: Bearer
<accessToken>
  Note over BE: If accessToken expired, frontend calls POST /api/auth/refresh
(cookies auto-sent)
```

---

3. Key files & folder map (quick reference)

```
my-mern-website/
├── backend/
│   ├── models/       # User, Tour, Booking
│   ├── routes/       # authRoutes.js, tourRoutes.js
│   ├── controllers/  # authController.js
│   ├── middleware/   # authMiddleware.js, errorHandler.js
│   ├── config/       # db.js
│   ├── utils/        # jwt.js, logger.js
│   └── server.js
└── frontend/
    └── src/
        ├── api/      # axios instance (api.js)
        ├── redux/     # authSlice
        ├── components/
        └── pages/
```

---

4. Important endpoints (recommended canonical names)

- `POST /api/auth/register` — register new user
- `POST /api/auth/login` — returns `{ accessToken }`, sets refresh cookie
- `POST /api/auth/refresh` — returns new accessToken (reads refresh cookie)
- `POST /api/auth/logout` — clear refresh cookie
- `GET /api/user/profile` — protected, requires Authorization
- `GET /api/tours` — public
- `GET /api/tours/:id` — public

Note: consider removing `.json` suffixes for REST clarity (use `/api/hero` instead of `/api/hero.json`).

---

5. CORS & Deployment notes

- **Allowed origins**: list all frontends (Netlify preview, onrender.com, localhost ports). Keep env var `FRONTENDS` for extra hosts.
- **Credentials**: `cors({ credentials: true })` and `axios({ withCredentials: true })` for cookie-based refresh tokens.
- **Netlify + Render**: Frontend served from Netlify, backend on Render — ensure `REACT_APP_API_URL` points to the backend origin (https://travelstrem-test.onrender.com).

---

6. Env vars checklist

```
# backend .env
PORT=5000
MONGO_URI=mongodb+srv://<user>:<pass>@cluster.../travelsTREM
JWT_ACCESS_SECRET=xxxx
JWT_REFRESH_SECRET=yyyy
ACCESS_TOKEN_EXPIRES_IN=15m
REFRESH_TOKEN_EXPIRES_IN=30d
FRONTENDS=http://localhost:3000,https://your-preview.netlify.app
```

```
# frontend .env
REACT_APP_API_URL=https://travelstrem-test.onrender.com
```

---

7. Recommended middlewares & best practices

- `helmet()` for security headers
- `express-rate-limit` on auth and contact endpoints
- centralized `errorHandler` middleware
- input validation (Joi/zod/express-validator)
- logging (morgan or pino)
- sanitize user input to prevent NoSQL injection

---

8. Quick checklist before production release

- [ ] HTTPS enforced on backend and frontend
- [ ] Secure cookie flags: `httpOnly`, `secure`, `sameSite=Strict` (or Lax as needed)
- [ ] Rotate JWT secrets and store in secrets manager
- [ ] Add monitoring (Sentry) and health endpoints
- [ ] Backup MongoDB and configure alerts

If you want a **PNG/SVG** export or a prettier diagram for stakeholder slides, tell me which format and I'll generate a downloadable image or a slide-ready version.