Design My Trip — TravelsTREM

Full implementation plan, front-end (React + Sass) and back-end (Node ES6 + MongoDB). Includes UI flow, component code, SASS, preloader, routes, controllers, MongoDB schemas, sample seed data, and example JSON responses formatted to work with `useComponentData` hook.

---

1. Overview & Flow

Two tabs inside `DesignMyTrip`:

- **Design by Local Operator** (human-powered)
- Step form (destination, travel dates, budget, group size, preferences, contact info)
- On submit: save a `LocalTripRequest` document, notify operators (email/WhatsApp/webhook), return JSON with handler for admin dashboard

- Operators can claim request in admin panel

- **Design by AI** (instant itinerary)

- Chat-style Q&A (initial preferences form then follow-ups)
- Frontend calls `/api/ai/design` which returns an itinerary (mocked in controller or integrated with OpenAI later)
- Save `AITripPlan` to DB and return JSON

Common UI considerations: - Preloader while fetching component metadata (use provided `useComponentData` hook) - Responsive: mobile-first, collapsible fields, sticky header for tabs - Accessible: labels, aria attributes

---

2. File Structure (recommended)

```
client/
  src/
    components/
      DesignMyTrip/
        DesignMyTrip.jsx
        DesignByLocalForm.jsx
        DesignByAIBot.jsx
        Stepper.jsx
        Preloader.jsx
        TripCard.jsx
        styles/
          design-my-trip.scss
    hooks/
      useComponentData.js  // (you already have this)
    utils/
      fetchData.js
      api.js
    App.jsx
server/
  src/
    models/
      LocalTripRequest.model.js
      AITripPlan.model.js
    controllers/
      designMyTrip.controller.js
    routes/
      designMyTrip.routes.js
    app.js
    db.js
    seed/
      seedData.js
```

---

3. Frontend — React + Sass (code)

> NOTE: The full code for each file is below. Paste into your `client/src/components/DesignMyTrip/` folder.

`DesignMyTrip.jsx` (main container)

```
// client/src/components/DesignMyTrip/DesignMyTrip.jsx
import React, { useState } from 'react';
import DesignByLocalForm from './DesignByLocalForm';
import DesignByAIBot from './DesignByAIBot';
import Preloader from './Preloader';
import useComponentData from '../../hooks/useComponentData';
import '../../components/DesignMyTrip/styles/design-my-trip.scss';

export default function DesignMyTrip() {
  const [tab, setTab] = useState('local');

  // fetch metadata for the component (filters / form structure)
  const { loading, error, componentData } = useComponentData('/api/design-metadata', { auto: true });

  if (loading) return <Preloader />;
  if (error) return <div className="dm-error">Error loading feature: {error}</div>;

  return (
    <section className="design-my-trip">
      <div className="dm-inner container">
        <h2 className="dm-title">Design My Trip</h2>
        <p className="dm-description">{componentData.description || 'Design a custom trip with AI or a local operator.'}</p>
        <div className="dm-tabs" role="tablist">
          <button
            className={tab === 'local' ? 'active' : ''}
            role="tab"
            aria-selected={tab === 'local'}
            onClick={() => setTab('local')}
          >Design by Local Operator</button>

          <button
            className={tab === 'ai' ? 'active' : ''}
            role="tab"
            aria-selected={tab === 'ai'}
            onClick={() => setTab('ai')}
          >Design by AI</button>
        </div>

        <div className="dm-panel">
          {tab === 'local' ? (
            <DesignByLocalForm structure={componentData.structure}
config={componentData.config} />
          ) : (
            <DesignByAIBot structure={componentData.structure}
config={componentData.config} />
          )}
        </div>
      </div>
    </section>
  );
}
```

`Preloader.jsx` **(simple skeleton)**

```
// client/src/components/DesignMyTrip/Preloader.jsx
import React from 'react';

export default function Preloader() {
  return (
    <div className="dm-preloader">
      <div className="dm-skeleton">
        <div className="s-hero" />
        <div className="s-lines">
          <div className="s-line" />
          <div className="s-line short" />
          <div className="s-line" />
        </div>
      </div>
    </div>
  );
}
```

`DesignByLocalForm.jsx` **(step form)**

```
// client/src/components/DesignMyTrip/DesignByLocalForm.jsx
import React, { useState } from 'react';
import { post } from '../../utils/api';

const defaultState = {
  destination: '',
  startDate: '',
  endDate: '',
  budget: '',
  groupSize: 2,
  preferences: '',
  name: '',
  email: '',
  phone: '',
};

export default function DesignByLocalForm({ structure, config }) {
  const [form, setForm] = useState(defaultState);
  const [loading, setLoading] = useState(false);
  const [success, setSuccess] = useState(null);
  const [error, setError] = useState(null);

  const handleChange = (e) => setForm({ ...form, [e.target.name]:
e.target.value });

  const handleSubmit = async (e) => {
    e.preventDefault();
    setLoading(true); setError(null); setSuccess(null);
    try {
      const res = await post('/api/design/local', form);
      if (res.status === 'success') {
        setSuccess(res.message || 'Request submitted!');
        setForm(defaultState);
      } else {
        setError(res.message || 'Failed');
      }
    } catch (err) {
      setError(err.message || 'Unknown error');
    } finally { setLoading(false); }
  };

  return (
    <form className="dm-local-form" onSubmit={handleSubmit}>
      <div className="row">
        <label>Destination</label>
        <input name="destination" value={form.destination}
onChange={handleChange} required />
      </div>

      <div className="row two">
        <div>
          <label>Start date</label>
          <input type="date" name="startDate" value={form.startDate}
onChange={handleChange} />
        </div>
        <div>
          <label>End date</label>
          <input type="date" name="endDate" value={form.endDate}
onChange={handleChange} />
        </div>
      </div>

      <div className="row two">
        <div>
          <label>Budget (approx)</label>
          <input type="number" name="budget" value={form.budget}
onChange={handleChange} />
        </div>
        <div>
          <label>Group size</label>
          <input type="number" name="groupSize" min="1"
value={form.groupSize} onChange={handleChange} />
        </div>
      </div>

      <div className="row">
        <label>Trip preferences</label>
        <textarea name="preferences" value={form.preferences}
onChange={handleChange} />
      </div>

      <div className="row two">
        <div>
          <label>Name</label>
          <input name="name" value={form.name} onChange={handleChange}
required />
        </div>
        <div>
          <label>Email</label>
          <input type="email" name="email" value={form.email}
onChange={handleChange} required />
        </div>
      </div>

      <div className="row">
        <label>Phone</label>
        <input name="phone" value={form.phone} onChange={handleChange} />
      </div>

      <div className="actions">
        <button type="submit" className="btn" disabled={loading}>{loading ?
'Submitting...' : 'Submit Request'}</button>
      </div>

      {success && <div className="dm-success">{success}</div>}
      {error && <div className="dm-error">{error}</div>}
    </form>
  );
}
```

`DesignByAIBot.jsx` **(chat-style simplified)**

```
// client/src/components/DesignMyTrip/DesignByAIBot.jsx
import React, { useState } from 'react';
import { post } from '../../utils/api';

export default function DesignByAIBot() {
  const [messages, setMessages] = useState([
    { from: 'bot', text: 'Hi — tell me where you want to go or what kind of
trip you like.' }
  ]);
  const [input, setInput] = useState('');
  const [loading, setLoading] = useState(false);

  const send = async () => {
    if (!input.trim()) return;
    const userMsg = { from: 'user', text: input };
    setMessages((m) => [...m, userMsg]);
    setInput('');
    setLoading(true);
    try {
      const res = await post('/api/design/ai', { prompt: input, history:
messages });
      if (res.status === 'success') {
        setMessages((m) => [...m, { from: 'bot', text:
res.componentData?.itineraryText || 'Here is your plan.' }]);
      } else {
        setMessages((m) => [...m, { from: 'bot', text: 'Sorry — failed to
generate. Try again.' }]);
      }
    } catch (err) {
      setMessages((m) => [...m, { from: 'bot', text: 'Error: ' +
(err.message || 'unknown') }]);
    } finally { setLoading(false); }
  };
  return (
    <div className="dm-ai-bot">
      <div className="dm-chat">
        {messages.map((m, i) => (
          <div key={i} className={`msg ${m.from}`}><div className="msg-
text">{m.text}</div></div>
        ))}
      </div>
      <div className="dm-compose">
        <input value={input} onChange={(e) => setInput(e.target.value)}
placeholder="Type your trip idea, e.g. 7 days Japan food + culture" />
        <button onClick={send} disabled={loading}>{loading ? 'Thinking...' :
'Ask AI'}</button>
      </div>
    </div>
  );
}
```

`utils/api.js` **(tiny wrapper)**

```
// client/src/utils/api.js
export async function get(url) {
  const res = await fetch(url);
  return res.json();
}

export async function post(url, body) {
  const res = await fetch(url, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(body),
  });
  return res.json();
}
```

---

4. Frontend Styling (SASS)

Create: `client/src/components/DesignMyTrip/styles/design-my-trip.scss`

```
// design-my-trip.scss
.design-my-trip {
  padding: 2rem 1rem;
  background: #fff;
  .dm-inner { max-width: 980px; margin: 0 auto; }
  .dm-title { font-size: 1.6rem; margin-bottom: 0.25rem; }
  .dm-description { color: #555; margin-bottom: 1rem; }
  .dm-tabs { display: flex; gap: 0.5rem; margin-bottom: 1rem;
    button { padding: 0.5rem 1rem; border-radius: 8px; border: 1px solid
#eee; background: transparent; cursor: pointer; }
    button.active { background: #0b79ff22; border-color: #0b79ff; }
  }

  .dm-panel { background: #fafafa; padding: 1rem; border-radius: 12px; }

  .dm-local-form { max-width: 720px; .row { margin-bottom: .75rem; label {
display:block; font-weight:600; margin-bottom: .25rem; } input, textarea {
width:100%; padding:.5rem; border-radius:6px; border:1px solid #ddd; } }
    .row.two { display:flex; gap: .75rem; > div { flex:1; } }
    .actions { margin-top: 0.75rem; .btn{ padding:.6rem 1rem; border-radius:
8px; border:none; background:#0b79ff; color:#fff; cursor:pointer;} }
  }

  .dm-ai-bot { .dm-chat { max-height: 360px; overflow:auto; padding: .
5rem; .msg { margin-bottom:.5rem; .msg-text{ padding:.5rem 0.75rem; border-
radius:8px; display:inline-block; } } .msg.user .msg-text {
background:#e6f0ff; } .msg.bot .msg-text { background:#fff; border:1px solid
#eee; } } .dm-compose { display:flex; gap:.5rem; margin-top:.5rem;
input{flex:1;padding:.5rem;border-radius:6px;border:1px solid #ddd}
button{padding:.5rem 1rem;border-radius:
6px;background:#0b79ff;color:#fff;border:none} } }

  .dm-preloader { .dm-skeleton{ background:#fff; padding:1rem; border-radius:
8px; .s-hero{ height:120px; background:#f0f0f0; border-radius:6px } .s-
lines{ margin-top: .75rem; .s-line{ height:14px; background:#f0f0f0; margin-
bottom:.5rem; border-radius:4px } .s-line.short{ width:40%; } } }
}

// responsive
@media (max-width: 600px) {
  .design-my-trip { padding: 1rem; .dm-title{font-size:1.2rem} .dm-local-
form .row.two{ flex-direction: column; } }
}
```

---

5. Back-end (Node ES6 + Express + MongoDB)

Use Node 18+, ES modules (`"type": "module"` in package.json)

`server/src/db.js`

```
// server/src/db.js
import mongoose from 'mongoose';

export async function connectDB(uri) {
  mongoose.set('strictQuery', false);
  await mongoose.connect(uri, { useNewUrlParser: true, useUnifiedTopology:
true });
  console.log('MongoDB connected');
}
```

Models

`LocalTripRequest.model.js`

```
// server/src/models/LocalTripRequest.model.js
import mongoose from 'mongoose';

const LocalTripRequestSchema = new mongoose.Schema({
  destination: String,
  startDate: Date,
  endDate: Date,
  budget: Number,
  groupSize: Number,
  preferences: String,
  name: String,
  email: String,
  phone: String,
  createdAt: { type: Date, default: () => new Date() },
  status: { type: String, default: 'new' }, // new, contacted, closed
  operatorAssigned: { type: mongoose.Schema.Types.ObjectId, ref: 'Operator',
default: null }
});

export default mongoose.model('LocalTripRequest', LocalTripRequestSchema);
```

`AITripPlan.model.js`

```
// server/src/models/AITripPlan.model.js
import mongoose from 'mongoose';

const AITripPlanSchema = new mongoose.Schema({
  prompt: String,
  itineraryText: String,
  structuredPlan: mongoose.Schema.Types.Mixed,
  createdAt: { type: Date, default: () => new Date() },
  userInfo: { name: String, email: String, phone: String },
});

export default mongoose.model('AITripPlan', AITripPlanSchema);
```

Controller: `designMyTrip.controller.js`

```
// server/src/controllers/designMyTrip.controller.js
import LocalTripRequest from '../models/LocalTripRequest.model.js';
import AITripPlan from '../models/AITripPlan.model.js';

export const getMetadata = async (req, res) => {
  // return the componentData JSON shape expected by useComponentData
  const payload = {
    status: 'success',
    message: 'Design My Trip metadata',
    componentData: {
      title: 'Design My Trip',
      description: 'Choose to design your trip with a local operator or
instantly with AI.',
      data: [],
      structure: {
        fields: {
          destination: { type: 'text', label: 'Destination' },
          startDate: { type: 'date', label: 'Start date' },
          endDate: { type: 'date', label: 'End date' },
          budget: { type: 'number', label: 'Budget' },
          groupSize: { type: 'number', label: 'Group size' },
          preferences: { type: 'textarea', label: 'Preferences' },
          name: { type: 'text', label: 'Your name' },
          email: { type: 'email', label: 'Email' },
          phone: { type: 'text', label: 'Phone' }
        }
      },
      config: {
        defaults: {
          destination: '', startDate: '', endDate: '', budget: '',
groupSize: 2, preferences: '', name: '', email: '', phone: ''
        },
        options: {
          maxGroupSize: 20,
          dateRange: { earliest: '2025-01-01', latest: '2026-12-31' }
        }
      }
    }
  };

  return res.json(payload);
};

export const submitLocalRequest = async (req, res) => {
  try {
    const body = req.body;
    const doc = await LocalTripRequest.create(body);
    // TODO: notify operators via email/SMS/webhook

    return res.json({ status: 'success', message: 'Request submitted',
componentData: { requestId: doc._id } });
  } catch (err) {
    console.error(err);
    return res.status(500).json({ status: 'error', message: 'Failed to
submit', componentData: {} });
  }
};

export const generateAIPlan = async (req, res) => {
  try {
    const { prompt, history, userInfo } = req.body;

    // For now: simple mocked AI generation (replace with OpenAI or other LLM
integration)
    const itineraryText = `AI-generated itinerary for: ${prompt}.\nDay 1: ...
\nDay 2: ...`;
    const structuredPlan = {
      title: `Trip for: ${prompt}`,
      days: [ { day: 1, summary: 'Arrival & city tour' }, { day: 2, summary:
'Main sights' } ]
    };

    const doc = await AITripPlan.create({ prompt, itineraryText,
structuredPlan, userInfo });

    return res.json({ status: 'success', message: 'AI itinerary generated',
componentData: { itineraryText, structuredPlan, planId: doc._id } });
  } catch (err) {
    console.error(err);
    return res.status(500).json({ status: 'error', message: 'Failed to
generate AI plan', componentData: {} });
  }
};
```

Routes: `designMyTrip.routes.js`

```
// server/src/routes/designMyTrip.routes.js
import express from 'express';
import { getMetadata, submitLocalRequest, generateAIPlan } from '../
controllers/designMyTrip.controller.js';

const router = express.Router();

router.get('/metadata', getMetadata); // returns componentData for
useComponentData
router.post('/local', submitLocalRequest);
router.post('/ai', generateAIPlan);
```

```
export default router;
```

App entry `app.js`

```
// server/src/app.js
import express from 'express';
import designRoutes from './routes/designMyTrip.routes.js';
import { connectDB } from './db.js';
import cors from 'cors';

const app = express();
app.use(cors());
app.use(express.json());

app.use('/api/design', designRoutes);

const PORT = process.env.PORT || 5000;
connectDB(process.env.MONGODB_URI || 'mongodb://localhost:27017/travelstrem')
  .then(() => app.listen(PORT, () => console.log('Server running on', PORT)))
  .catch((e) => console.error('DB connect failed', e));

export default app;
```

---

6. DB: Collections & example documents

- **localtriprequests** (LocalTripRequest)
- _id, destination, startDate, endDate, budget, groupSize, preferences, name, email, phone, createdAt, status, operatorAssigned
- **aitripplans** (AITripPlan)
- _id, prompt, itineraryText, structuredPlan, createdAt, userInfo

Example LocalTripRequest document

```
{
  "_id": "64f8a1...",
  "destination": "Nepal",
  "startDate": "2025-11-10T00:00:00.000Z",
  "endDate": "2025-11-17T00:00:00.000Z",
  "budget": 1200,
  "groupSize": 4,
  "preferences": "Trekking, moderate fitness",
  "name": "Akshat",
  "email": "akshat@example.com",
  "phone": "+91-9...",
  "createdAt": "2025-08-10T...",
  "status": "new"
}
```

Example AITripPlan document

```
{
  "_id": "650abc...",
  "prompt": "7 days Japan food + culture",
  "itineraryText": "AI-generated itinerary...",
  "structuredPlan": { "title": "7 days Japan", "days": [...] },
  "createdAt": "2025-09-16T...",
  "userInfo": { "name": "Akshat", "email": "a@b.com" }
}
```

---

7. API endpoints (summary)

```
GET  /api/design/metadata      -> returns componentData JSON (for
useComponentData)
POST /api/design/local         -> accepts local operator requests, returns
{status, message, componentData: { requestId }}
POST /api/design/ai            -> accepts { prompt, history, userInfo },
returns AI plan
```

Each endpoint follows your JSON wrapper convention `status, message, componentData`.

---

8. Notes & Next steps

- **Notifications**: add email / Twilio / WhatsApp integration inside `submitLocalRequest` to notify local operators.
- **LLM Integration**: swap mocked AI logic in `generateAIPlan` with OpenAI or your preferred LLM; keep structuredPlan so you can later convert into cards.
- **Admin Dashboard**: build endpoints for operators to list/claim requests and update `status`.
- **Security**: add rate-limit, validation (Joi or express-validator) and sanitization for inputs.
- **Tests**: add unit tests for controllers and integration tests for routes.

---

9. Paste-ready snippets

If you'd like, I can export these files as a zip or create the files directly in your repo structure. Tell me which format you prefer and I will prepare that for you.

End of document.