MERN Task: Company Website + Contact Submissions + Admin View

Goal

Build a simple 4-page website (Home, About, Services, Contact) using the MERN stack where:

- Each page content is served from the backend APIs.
- Users can submit a contact form, and submissions are saved in MongoDB.
- An Admin can "log in" using a secret key (stored in backend `.env`) to view all contact submissions.

Backend Requirements (Node + Express + MongoDB)

1) Routes (Public Content APIs)

Create **4 GET routes** that return basic JSON content:

1. `GET /api/home`
2. `GET /api/about`
3. `GET /api/services`
4. `GET /api/contact`

Each route must return meaningful content, for example:

- `title`
- `description`
- `highlights` or `servicesList`
- any extra fields you want

> These endpoints will be consumed by the React pages.

---

2) Contact Form Submission Route (Store in DB)

Create a route to handle contact form submission:

- `POST /api/contact/submit`

Request body (JSON):

- `name` (string)
- `email` (string)
- `message` (string)

Behavior:

- Validate required fields (basic validation is enough).
- Save the submission into MongoDB (collection: `contacts`).
- Return a **thank-you message** from the server, e.g.:

```
{ "message": "Thanks for contacting us! We will get back to you soon."
}
```

MongoDB Schema (example fields):

- `name`
- `email`
- `message`
- `createdAt` (timestamp)

---

3) Admin Authentication (Simple Secret Key Verification)

Admin authentication is intentionally simple:

- Store a secret key in backend `.env`, for example:
  - `ADMIN_SECRET=some-secret-string`

Create an endpoint:

- `POST /api/admin/login`

Request body:

- `secretKey`

Behavior:

- If `secretKey` matches `process.env.ADMIN_SECRET`, return success (and optionally a token-like string).
- If not, return an error response.

> This is not real JWT auth—just basic verification for this task.

---

4) Admin: View All Contact Submissions (Protected)

Create an admin-only route:

- `GET /api/admin/contacts`

Protection rule:

- This route must only return data if the request includes the correct secret key (choose one approach):
  - Option A: `Authorization` header contains the key
  - Option B: custom header like `x-admin-key`
  - Option C: query param (least preferred, but allowed for this assignment)

Response:

- Return an array of all contact submissions from MongoDB (latest first is preferred).

Frontend Requirements (React + React Router)

1) Pages and Navigation

Create a React app with routing using `react-router-dom`:

Pages:

- `/` → Home
- `/about` → About
- `/services` → Services
- `/contact` → Contact
- `/admin` → Admin Login
- `/admin/contacts` → Admin Contacts List

Navbar Links:

- Home, About, Services, Contact, Admin

---

2) Fetch Page Content from Backend

Each page must fetch its content from the backend API:

- Home page calls `GET /api/home`
- About page calls `GET /api/about`
- Services page calls `GET /api/services`
- Contact page may call `GET /api/contact` for the page intro text

UI expectation:

- Show loading state while fetching.
- Render data from server (title, description, etc.)

---

3) Contact Page Form + Server Message

On `/contact`, create a form with:

- Name
- Email
- Message

On submit:

- Send data to `POST /api/contact/submit`
- Display the **thank you message received from the server** on the UI.
- Optionally clear the form after success.

---

4) Admin Login + View Contacts

Admin Login page (`/admin`):

- Input: secret key
- Call `POST /api/admin/login`
- If success:
  - Store the returned "authenticated" flag/key in `localStorage` (or React state).
  - Redirect to `/admin/contacts`

Admin Contacts page (`/admin/contacts`):

- Fetch from `GET /api/admin/contacts`
- Send the secret key in headers (based on the approach you chose).
- Display contacts in a table/list with:
  - Name, Email, Message, Date

Route protection (frontend):

- If admin is not logged in (no key stored), redirect to `/admin`.

Deliverables

1. **Backend**
   - Express server with required routes
   - MongoDB connection
   - Contact model + saving submissions
   - Admin login + protected admin contacts route
   - `.env` usage (`MONGO_URI`, `ADMIN_SECRET`, `PORT`)
2. **Frontend**
   - React routing with 4 main pages
   - API-driven content rendering for each page
   - Contact form submission + thank-you message display
   - Admin login + admin contacts list with basic protection

Optional Enhancements (Bonus)

- Add timestamps and show "time ago"
- Add basic email validation
- Add pagination for admin contacts
- Add a reusable API utility function in React
- Add toast notifications for success/error