

 PROJECT SUMMARY

pdk



PROJECT SUMMARY

You will build a **full-stack API Testing Tool** that allows users to:

- Enter an API URL
- Select HTTP method (GET, POST, PUT, DELETE, PATCH)
- Add headers, params, and request body JSON
- Send the request
- See the response status, headers, and body
- Save request history
- Save API collections (like Postman folders)
- User authentication (optional)

This will be a **production-level project** that showcases frontend–backend communication, API proxying, database usage, and UI design.

Use Cases



USE CASES

1. Send API Requests

- **Use Case:** Developer tests REST APIs
- **Acts like:** Postman/Insomnia basic request tool

2. Manage Request History

- **Use Case:** User wants to review previously tested APIs
- **Save:** URL, method, body, headers, timestamp

3. Save Collections

- **Use Case:** Testers store grouped API endpoints
- Example: "User APIs", "Order APIs", "Auth APIs"

4. Response Viewer

- **Use Case:** View response JSON prettified with syntax highlighting
- Show:
 - Response status
 - Time taken
 - Headers
 - Body

5. Authentication

- **Use Case:** User logs in and saves data securely

 TECH STACK



TECH STACK (Recommended)

Frontend → React + Vite

or Next.js (app router)

Backend → Node.js + Express

Database → Supabase (best option for quick auth + DB)



2-WEEK STEP-BY-STEP DEVELOPMENT PLAN



2-WEEK STEP-BY-STEP DEVELOPMENT PLAN



WEEK 1 — Build Core Features

Day 1 — Setup Everything

Tasks:

- Initialize React (Vite or Next.js)
- Setup Tailwind CSS (for UI)
- Setup Supabase project (or Firebase/Convex)
- Setup Node.js + Express server

Output:

- Folder structure ready
 - Basic connection check for Supabase from the backend
-

Day 2 — Build UI Layout

Tasks:

- Build the main screen:

- Left sidebar → History + Collections
 - Center → API Input Form
 - Right → Response Viewer
- Use Tailwind for clean layout

Output:

- Static UI without functionality
-

Day 3 — API Input Form

Tasks:

- Create fields:
 - URL
 - Method dropdown(GET, POST, PUT, DELETE)
 - Headers editor
 - Params editor
 - Body JSON editor (textarea)
- Add “Send” button

Output:

- Fully functional form UI
-

Day 4 — Backend Proxy Endpoint

Why?

Some APIs require CORS... browsers can't call them directly.

Tasks:

- Create POST `/proxy` endpoint in Express
 - Accept `url`, `method`, `headers`, `body`
 - Forward request using `fetch` or `axios`
 - Return response

Output:

- Backend forwarder working
-

Day 5 — Connect Frontend → Backend

Tasks:

- Call `/proxy` when user clicks Send
- Display response:
 - Status code
 - Time
 - Headers
 - Body (formatted JSON)

Output:

- Core API testing functionality fully working

Day 6 — Save Request History

Tasks:

- Create Supabase table: `history`
 - `id, user_id, url, method, body, headers, created_at`
- When a request is sent → save to DB
- Show history list in sidebar
- Clicking history loads data back into form

Output:

- Working request history system
-

Day 7 — Build Collections System

Tasks:

- Table: `collections, collection_items`
- User creates folder (e.g., Auth APIs)
- Add API requests to a collection
- Display collections in sidebar

Output:

- Collections feature complete



WEEK 2 — Advanced Features + Polish

Day 8 — Authentication (Optional but Recommended)

Tasks:

- Implement Supabase Auth / Firebase Auth
- Save history + collections per user
- Create:
 - Login
 - Signup
 - Logout

Output:

- Auth + protected dashboard
-

Day 9 — Better JSON Editor

Tasks:

- Install a code editor library:
 - `react-ace` or
 - `react-json-view`

- Add syntax highlighting
- Add expand/collapse

Output:

- Modern API request/response viewer
-

Day 10 — Environment Variables

Tasks:

- Allow multiple envs:
 - Dev
 - Staging
 - Production
- User can define keys & values
- Insert env variable into URL/body

Output:

- Working environment system (like Postman ENV)
-

Day 11 — Error Handling UI

Tasks:

- Catch CORS errors

- Show descriptive errors
- Handle no internet
- Show loading spinner

Output:

- Stable and user-friendly tool
-

Day 12 — UI Polishing

Tasks:

- Improve styling
- Add dark/light mode
- Add animations

Output:

- Professional grade UI
-

Day 13 — Deployment

Tasks:

- Deploy backend on Render/railway
- Deploy frontend on Vercel/Netlify
- Connect everything with env variables

Output:

- Live project URL
-

Day 14 — Documentation & Final Testing

Tasks:

- Write README:
 - Summary
 - Features
 - Tech stack
 - Screenshots
 - Installation
- Record demo video
- Test thoroughly

Output:

- Final polished portfolio project 



SKILLS YOU WILL LEARN



SKILLS YOU WILL LEARN

Frontend Skills

- React components
- State management
- Fetch / Axios
- JSON editor integrations
- UI/UX thinking

Backend Skills

- Express routing
- Proxy servers
- Handling headers, body, params
- Error handling
- Deployment

Database Skills

- Supabase tables
- Auth
- Insert/select/update/delete
- Real-time updates (optional)

Other Important Skills

- Debugging APIs
 - Designing REST interactions
 - Organizing large projects
 - Mapping real-world tools (Postman) into your own app
-



FREE RESOURCES (YouTube + Docs)



FREE RESOURCES (YouTube + Docs)

React (or Next.js)

YouTube

- Code with Harry React playlist
- Dave Gray React tutorials
- Net Ninja React series

Docs

- <https://react.dev/>
 - <https://nextjs.org/docs>
-

Node.js + Express

YouTube

- Thapa Technical Node.js playlist
- Traversy Media Express.js Crash Course

Docs

- <https://nodejs.org/en/docs>
- <https://expressjs.com/en/guide/routing.html>

Supabase

YouTube

- Supabase official channel
- Fireship Supabase crash course

Docs

- <https://supabase.com/docs>
-

Firebase

YouTube

- Firebase official playlist
- Traversy Media Firebase series

Docs

- <https://firebase.google.com/docs>
-

Convex

YouTube

- Convex official tutorials
- Beginner-friendly “Convex + React” videos

Docs

- <https://docs.convex.dev/>

Tailwind CSS

YouTube

- Fireship Tailwind in 15 mins
- Net Ninja Tailwind series

Docs

- <https://tailwindcss.com/docs>

Extra Useful Libraries

JSON Editor

- react-ace → good code editor
- react-json-view → great for pretty response

Install:

```
npm i react-ace react-json-view
```

