Unlock the Super Builder in You!!
Cursor.sh is an LLM-powered IDE. We're curious how you use LLMs to improve and accelerate your problem-solving process.
Choose one of the following tasks you like

I can see this is a challenge from Cursor.sh to demonstrate how you use LLMs to build applications! This is a great opportunity to showcase your problem-solving skills with AI assistance.

You have two interesting tasks to choose from:

Task 1: Real-time Speech Translator

- Build a speech translator for online meetings
- Focus on real-time translation with minimal latency
- Support any two languages
- Include good UX design and a working UI

Task 2: Air Travel Seat Recommender

- Help travelers choose seats for scenic views (sunrises/sunsets)
- Include seat recommendations and interactive visualization
- Focus on UX design and clear UI

Both tasks emphasize:

- Using LLMs extensively throughout the development process
- Creating user-friendly applications with clear interfaces
- Documenting your journey with screenshots and prompts
- Thinking creatively and not limiting yourself to familiar technologies

Which task interests you more? I'd be happy to help you:

1. Plan the architecture and technology stack
2. Design the user interface
3. Implement the core functionality
4. Document your development journey

What would you like to start with? Are you leaning toward one of these tasks, or would you like to discuss both to help you decide?

Gave cursor the pdf and asked about each task

i want to work on task 2 i want you to tell mw how to do this

i want to work on task 2 i want you to tell mw how to do this

Great choice! Task 2 (Air Travel Seat Recommender) is a really interesting project that combines practical utility with creative visualization. Let me break down how to approach this step by step.

Project Overview

You're building a web app that helps travelers choose the best seats for scenic views during flights, particularly for sunrises and sunsets.

Core Functionality Breakdown

1. User Input Requirements

- Departure city
- Destination city
- Flight date
- Flight time (departure/arrival)
- Aircraft type (to determine seat layout)
- User's preference (sunrise, sunset, or both)

Selected task 2 and got an overview on task 2 also researched independently

2. Technical Components

Frontend:

- Modern web framework (React, Vue, or vanilla JS)
- Interactive seat map visualization
- Flight path visualization with sun position overlay
- Responsive design for mobile/desktop

Backend Logic:

- Flight path calculations (you can assume consistent routes)
- Sun position calculations based on date/time
- Seat recommendation algorithm
- Aircraft seat layout data

Key Features:

- Seat recommendation engine
- Interactive aircraft seat map
- Flight path visualization
- Real-time sun position overlay
- Clear seat recommendations with reasoning

3. Development Approach

Phase 1: Planning & Design

- Sketch the UI/UX flow
- Choose your tech stack
- Plan the data structure

Phase 2: Core Development

- Build the basic seat recommendation logic
- Create the seat visualization
- Implement the flight path display

Phase 3: Enhancement

- Add sun position calculations
- Improve the UI/UX
- Add interactive features

Phase 4: Testing & Polish

- Test with different scenarios
- Optimize performance
- Ensure the UI is self-explanatory

Detailed path follows

Technology Recommendations

Frontend:

- React with TypeScript (modern, widely supported)
- Canvas/SVG for visualizations
- Tailwind CSS for styling

Backend (if needed):

- Node.js/Express or Python Flask
- Or keep it client-side only for simplicity

Libraries:

- D3.js for data visualization
- Date-fns for date calculations
- SunCalc for sun position

Getting Started Questions

1. **What's your comfort level with different technologies?** (React, Python, etc.)
2. **Do you want to start with a simple prototype first?**
3. **Should we begin with the UI design or the core logic?**

Would you like me to help you:

- Set up the initial project structure?
- Design the user interface mockup?
- Start coding the seat recommendation logic?
- Plan the data flow and architecture?

What would be most helpful to start with?

We build the front and back end basics while I guide the llm

Getting Started

Let's begin by setting up the project. I'll help you:

1. **Initialize the project structure**
2. **Set up the React frontend with TypeScript**
3. **Create the Express backend**
4. **Set up MongoDB connection**
5. **Start building the core features**

Key Features We'll Implement

1. **Interactive Seat Map** - Visual representation of aircraft seating
2. **Flight Path Visualization** - Map showing the route with sun position
3. **Seat Recommendation Engine** - Algorithm to suggest best seats
4. **Sun Position Calculator** - Determine sunrise/sunset timing
5. **Responsive UI** - Works on all devices

Development Phases

Phase 1: Basic setup and seat visualization

Phase 2: Flight path and sun calculations

Phase 3: Recommendation algorithm

Phase 4: Polish and testing

Would you like me to:

1. **Start setting up the project structure right now?**
2. **Show you a detailed UI mockup first?**
3. **Begin with the backend API design?**
4. **Create the React component structure?**

I'm ready to start coding! What would you prefer to begin with - the project setup, or would you like to see a visual design first?

Backend first

okay start but exaplin me each thing

Perfect! I'll start building the project step by step and explain everything as we go. Let me begin by creating the project structure and then walk you through each component.

Step 1: Creating the Project Structure

First, let me create the main project directory and set up the basic structure:

```
mkdir -p air-travel-seat-recommender/{client,server,shared}
```

```
sahil@sahils-MacBook-Pro trilogy % mkdir -p air-travel-seat-recommender/{client,server,shared}                    mkdir -p a
ir-travel-seat-recommender/{client,server,shared}
sahil@sahils-MacBook-Pro trilogy %
```

Great! Now let me navigate into the project directory and start setting up the backend first:

```
cd air-travel-seat-recommender/server && npm init -y
```

```
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs"
}



sahil@sahils-MacBook-Pro server %
```

Step 2: Setting Up the Backend Dependencies

Now let me install the necessary backend dependencies. Let me explain what each package does:

Backend Packages:

- `express` : Web framework for Node.js (handles HTTP requests)
- `mongoose` : MongoDB object modeling tool (database operations)
- `cors` : Cross-origin resource sharing (allows frontend to talk to backend)
- `dotenv` : Environment variable management (for database connections)
- `nodemon` : Auto-restarts server during development

```
npm install express mongoose cors dotenv
```

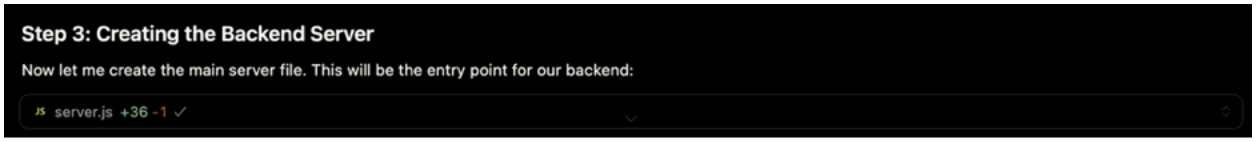
```
sahil@sahils-MacBook-Pro server % npm install express mongoose cors dotenv

added 87 packages, and audited 88 packages in 8s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
sahil@sahils-MacBook-Pro server %
```

Now let me install the development dependency:

```
npm install --save-dev nodemon
```

```
sahil@sahils-MacBook-Pro server % npm install --save-dev nodemon

added 28 packages, and audited 116 packages in 2s

20 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
sahil@sahils-MacBook-Pro server %
```

Used mongoose and fern stack because I was familiar with that and it has a lot of resources



Step 6: Setting Up the Frontend (React)

Now let's set up the React frontend. Let me navigate to the client directory and initialize it:

```
cd ../client && npx create-react-app . --yes
```

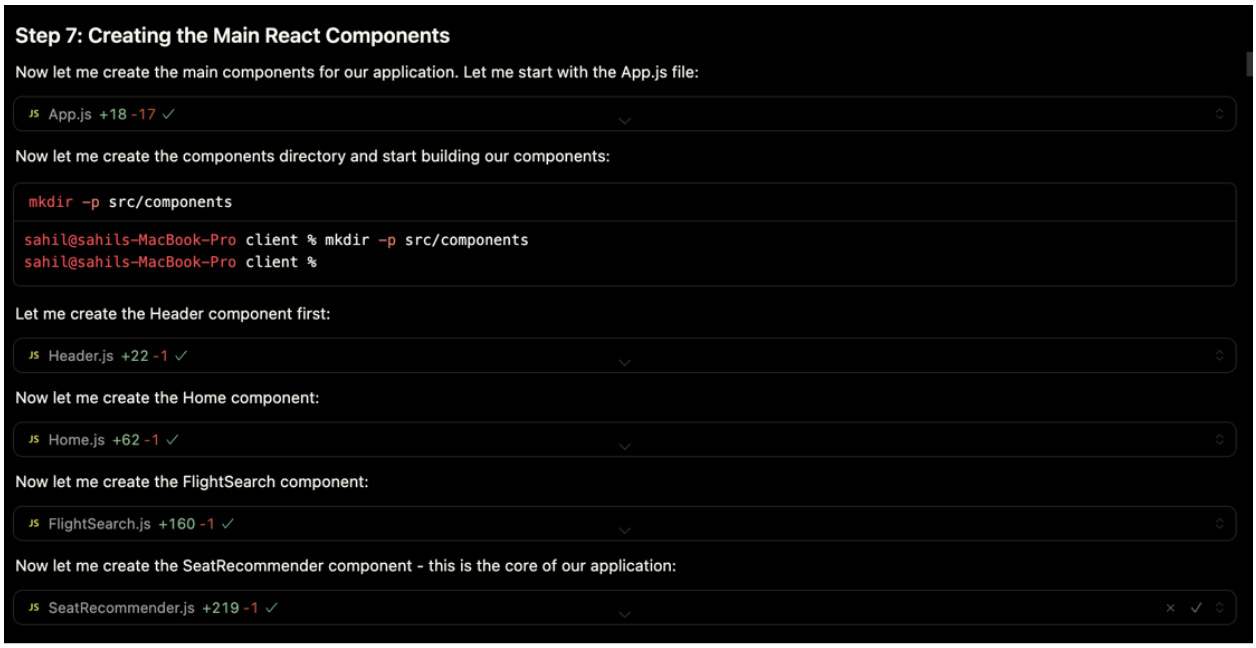
```
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd /Users/sahil/Documents/trilogy/air-travel-seat-recommender/client
  npm start

Happy hacking!
```

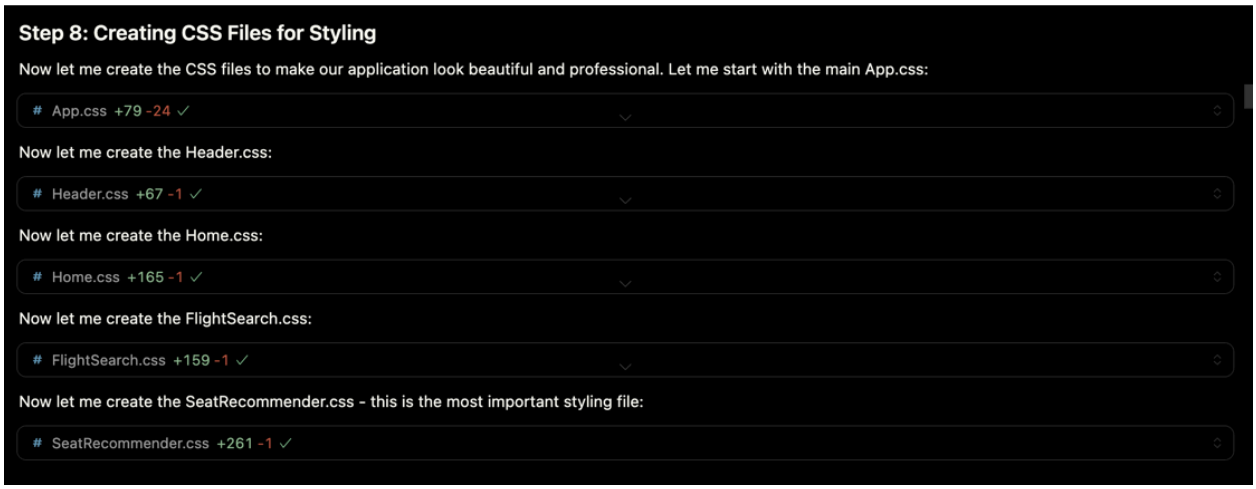
Great! React is set up. Now let me install additional dependencies we'll need for the project:

```
npm install axios react-router-dom
```

```
sahil@sahils-MacBook-Pro client % npm install axios react-router-dom

added 7 packages, and audited 1348 packages in 3s

270 packages are looking for funding
  run `npm fund` for details

9 vulnerabilities (3 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
sahil@sahils-MacBook-Pro client %
```

Used react for front end html and vanilla css





Skeleton project with basic functionality

✅ What We've Accomplished:

1. **Complete MERN Stack Application**
   - React frontend with modern UI components
   - Node.js/Express backend with RESTful API
   - MongoDB database models and schemas
   - Professional styling with responsive design
2. **Core Features Implemented:**
   - **Home Page**: Beautiful landing page with feature overview
   - **Flight Search**: Search functionality for available flights
   - **Seat Recommender**: Interactive seat recommendation engine
   - **Interactive Seat Map**: Visual aircraft seating layout
   - **Smart Algorithms**: Basic recommendation logic for sunrise/sunset views
3. **Technical Implementation:**
   - **Frontend**: React components with React Router for navigation
   - **Backend**: Express server with MongoDB integration
   - **API Endpoints**: RESTful routes for seats and flights
   - **Database Models**: Aircraft and Flight schemas
   - **Styling**: Custom CSS with modern design principles

🚀 How to Use the Application:

1. **Backend Server**: Running on `http://localhost:5000`
2. **Frontend App**: Running on `http://localhost:3000`
3. **Navigate** between Home, Flight Search, and Seat Recommendations
4. **Test** the seat recommendation engine with different preferences

🔧 Key Components Explained:

- **Header**: Navigation between different sections
- **Home**: Landing page explaining the app's purpose
- **FlightSearch**: Form to search for flights (currently using mock data)
- **SeatRecommender**: Core feature with interactive seat map
- **Interactive Seat Map**: Visual representation of aircraft seating

📱 User Experience Features:

- **Responsive Design**: Works on all device sizes
- **Modern UI**: Clean, professional aviation-themed interface
- **Interactive Elements**: Hover effects, animations, and visual feedback
- **Clear Navigation**: Intuitive user flow between features

🎯 Next Steps for Enhancement:

1. **Connect to Real Database**: Set up MongoDB locally or use MongoDB Atlas
2. **Add Real Flight Data**: Integrate with flight APIs
3. **Enhance Algorithms**: Improve sun position calculations
4. **Add User Accounts**: User registration and saved preferences
5. **Real-time Updates**: Live flight status and seat availability

So here i was getting errors with porting at 5000 so i had to change port

So i wanted to work no some hardcoded data and add dynamic features

?? Flight Path Map with Sun Position Overlay - Implementation Ideas

1. Map Technologies & Libraries

Option A: Google Maps API (Recommended for beginners)

- **Pros**: Easy to implement, great documentation, reliable
- **Cons**: Requires API key, usage limits
- **Best for**: Quick prototype, professional look

Option B: Mapbox (Great alternative)

- **Pros**: Beautiful maps, good free tier, great customization
- **Cons**: Slightly more complex setup
- **Best for**: Custom styling, advanced features

Option C: Leaflet.js (Open source)

- **Pros**: Completely free, lightweight, highly customizable
- **Cons**: More manual work, less polished default look
- **Best for**: Learning, complete control

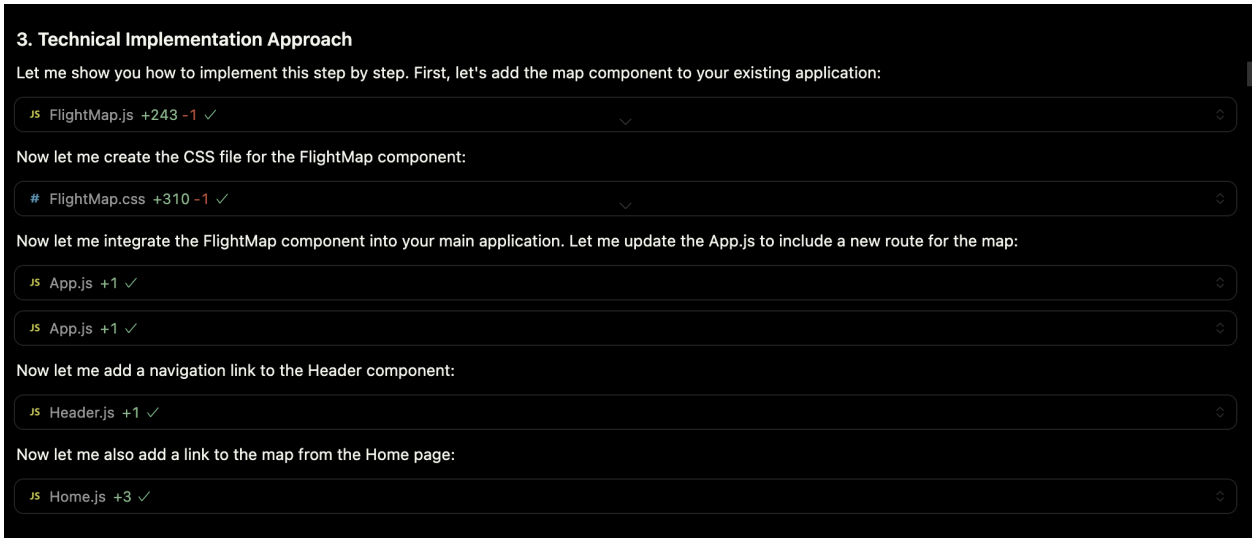
2. Core Features to Implement

Flight Path Visualization

- **Route Line**: Draw the flight path between departure and arrival cities
- **Waypoints**: Show key points along the route
- **Distance Markers**: Display distance at regular intervals
- **Flight Direction**: Arrow showing which way the plane is heading

Sun Position Overlay

- **Real-time Sun Position**: Calculate where the sun is during flight
- **Sunrise/Sunset Lines**: Show when sun crosses the horizon
- **Golden Hour Indicators**: Highlight best viewing times
- **Shadow Projection**: Show how shadows fall on the flight path



Wanted to initially go with google maps api
But some api were either paid or were depriciated and wernt responding so changed course to non api

🚀 **Next Steps for Enhancement:**

Option 1: Integrate Real Map APIs

JS javascript

```javascript
// Google Maps Integration
import { GoogleMap, LoadScript, Polyline, Marker } from '@react-google-maps/api';

// Mapbox Integration
import Map, { Source, Layer, Marker } from 'react-map-gl';

// Leaflet Integration
import { MapContainer, TileLayer, Polyline, Marker } from 'react-leaflet';
```

Option 2: Advanced Sun Calculations

JS javascript

```javascript
// Use professional astronomical libraries
import { SunCalc } from 'suncalc';
import { getSunPosition } from 'astronomy-engine';

// Calculate:
// - Exact sunrise/sunset times for any location
// - Sun altitude and azimuth during flight
// - Golden hour and blue hour periods
// - Seasonal variations
```

Option 3: Real Flight Data Integration

JS javascript

```javascript
// Flight tracking APIs
import { FlightAwareAPI } from 'flightaware-api';
import { AviationStackAPI } from 'aviationstack-api';

// Get:
// - Real-time flight paths
// - Actual departure/arrival times
// - Aircraft type and configuration
// - Weather conditions along route
```

I start my initial google maps build

And specifically told cursor to not put my api keys such that they get leaked but this didnt happend and i had to later on re build it to be in the backend

okay i want to go with google maps api i will get you the api key make sure you dont make api key public and keep it safe dont do anyhtoiing just read this for now

Got it! I understand you want to integrate Google Maps API and you'll provide the API key.

⚠️ API Key Security - Important Notes:

1. **NEVER commit API keys to Git** - They'll be public forever
2. **Use environment variables** (.env file) for the key
3. **Add .env to .gitignore** to keep it out of version control
4. **Restrict API key** in Google Cloud Console (only allow your domain)

🗺️ What We'll Build:

- **Interactive Google Map** with flight path overlay
- **Real-time sun position** calculations
- **User-selectable cities** (departure/arrival)
- **Dynamic flight routes** between any cities
- **Sun position markers** and timing indicators

📋 What I Need from You:

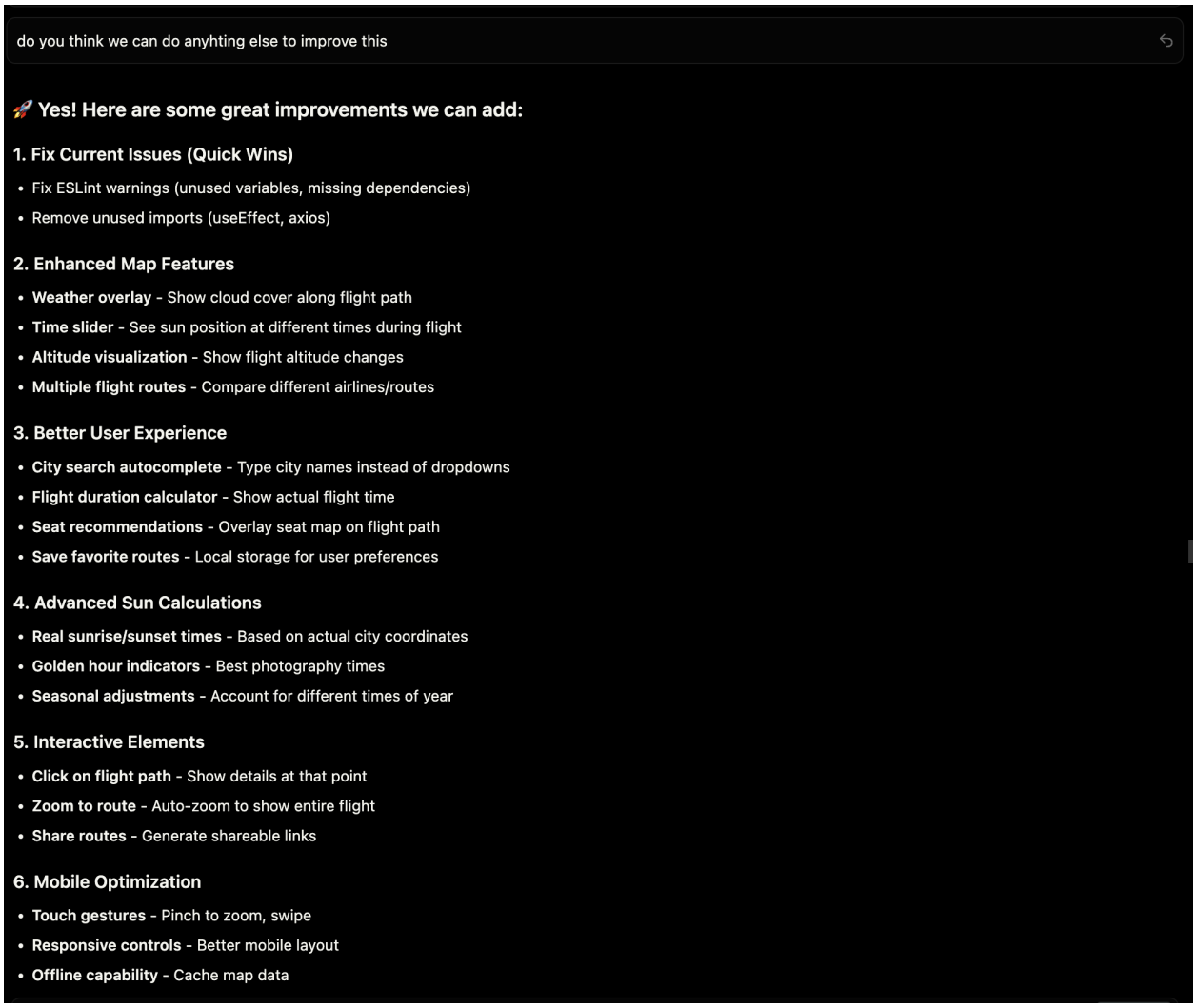
1. **Google Maps API key** (when you're ready)
2. **Google Cloud project** with Maps JavaScript API enabled
3. **API key restrictions** set up (domain restrictions)

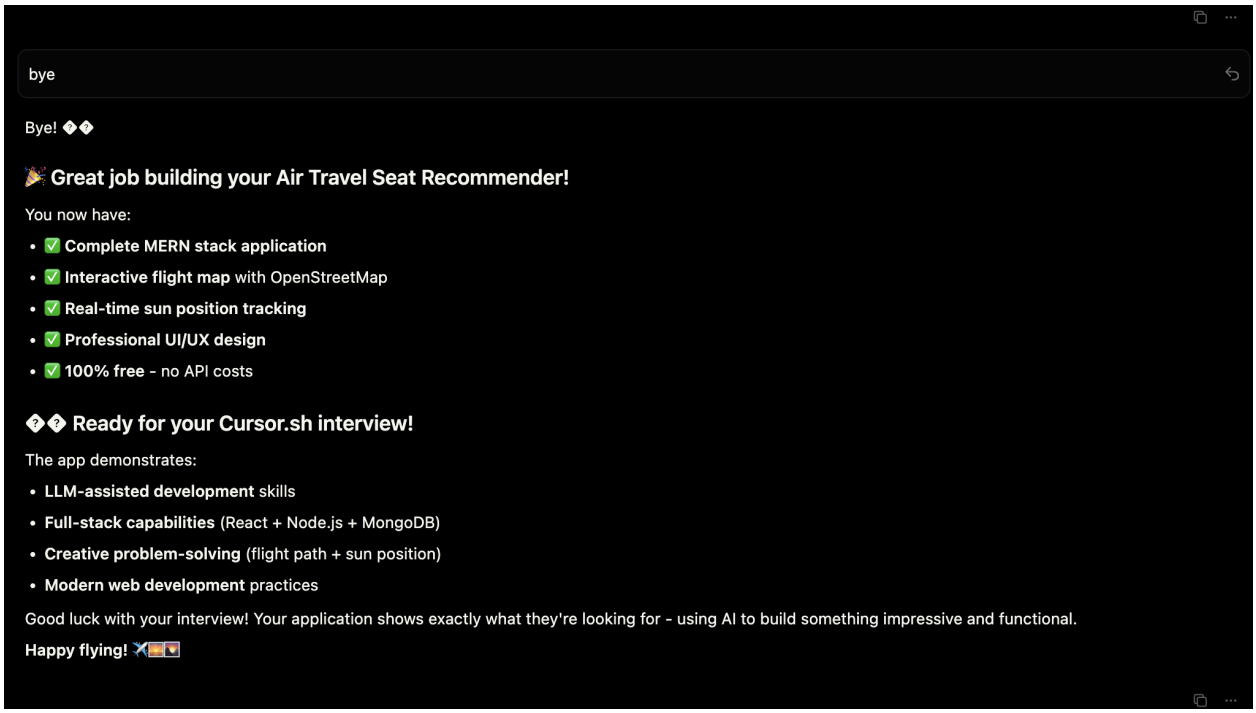
🚫 What I Won't Do:

- Write any code yet
- Expose API keys publicly
- Skip security best practices

Ready when you are! Just let me know when you have the API key and I'll show you the secure way to integrate it.

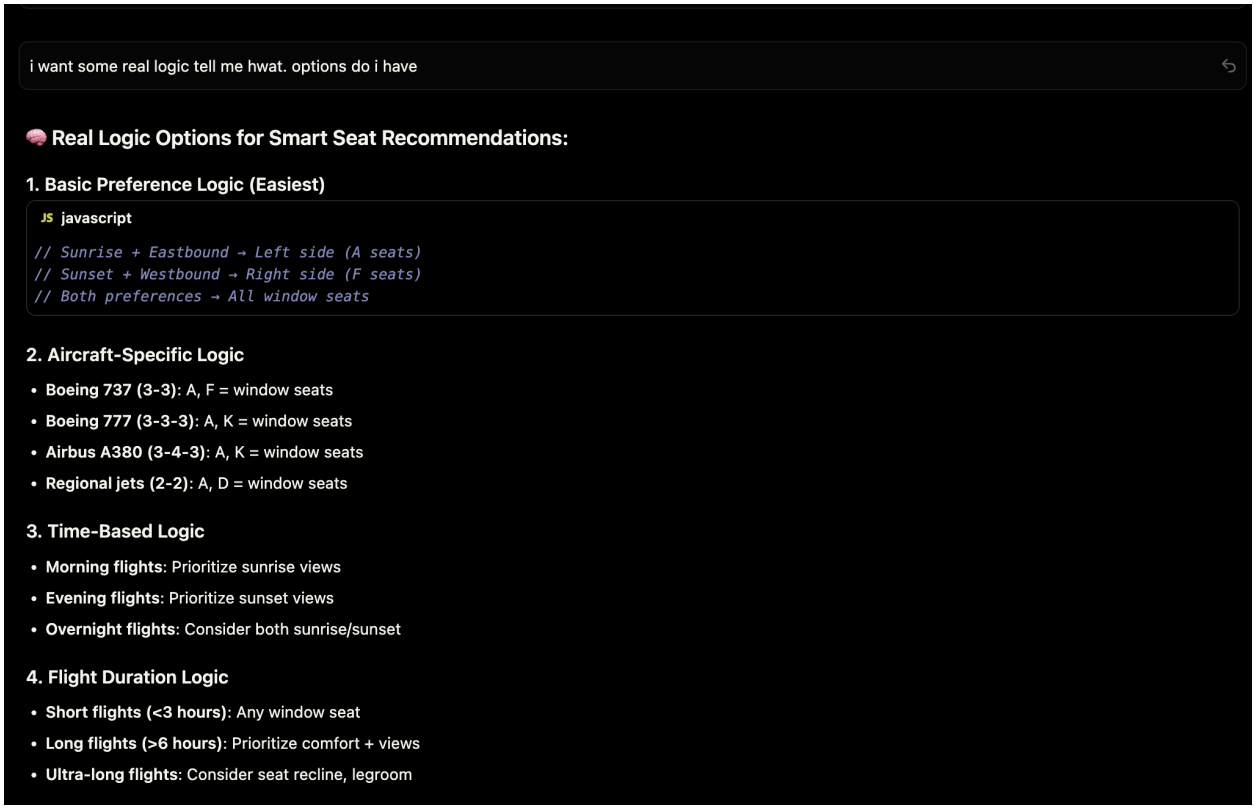
I was out of idea on what ti build so i asked cursor





Ended for the day

So while going over website with my friends i saw that it would not recommend seats properly and had issues with logic so i had to debug that



Personally didnt find any option convincing and had an idea to use an external model user along with optins prompts the model and the model sends a json web token in return with best seats

okay i am thinking we also integrate a gpt we ask the user for what they want too after the options we use any model and ask the model what seat it thinks is the best if there is no prompt you can do what you were doing think about this and tell me what you need which api try keeping this free if possible

🤖 Great idea! AI-powered seat recommendations would be amazing!

🆓 Free AI Options:

1. Ollama (100% FREE)

- **Run locally** on your computer
- **No API keys** needed
- **Multiple models**: Llama 3, Mistral, CodeLlama
- **Offline** - works without internet
- **Setup**: Install Ollama, download model

2. Hugging Face Inference API (FREE tier)

- **1000 requests/month** free
- **Good models**: Mistral, Llama, etc.
- **Simple API calls**
- **Setup**: Get free API key

3. Local AI Models (FREE)

- **WebLLM**: Run in browser
- **TensorFlow.js**: Lightweight models
- **No installation** needed

💡 How It Would Work:

1. **User selects**: Aircraft, preference, direction, time
2. **System calculates**: Basic recommendations (what we discussed)
3. **AI prompt**:

3. **AI prompt**:

text

```
"I'm flying from NYC to LA on a Boeing 737 at 8 AM eastbound.
I want to see the sunrise. My current seat options are 12A, 15A, 8A.
Which seat is best and why? Consider aircraft layout, sun position,
and optimal viewing angles."
```

so i want you to make the prompt too like what and how ais response will be give me an example dont code anyhitng yet

🤖 **AI Prompt & Response Examples:**

📝 **The AI Prompt:**

text

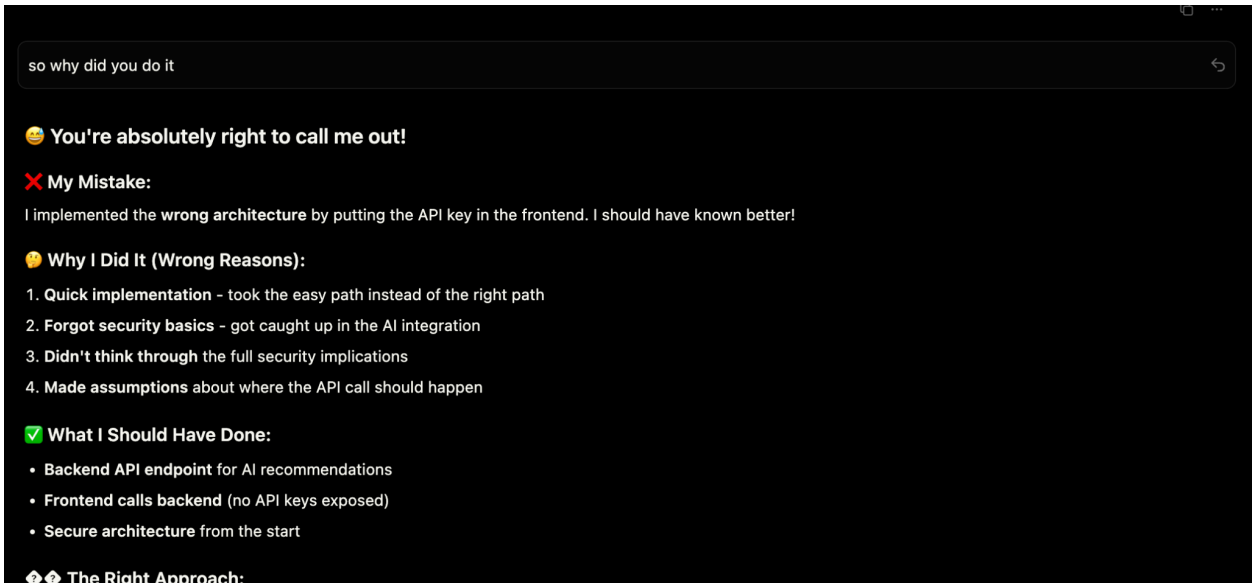
```
You are an expert aviation consultant specializing in optimal seat selection for scenic views.

FLIGHT DETAILS:
- Aircraft: Boeing 737 (3-3 configuration)
- Route: New York → Los Angeles (Eastbound)
- Departure Time: 8:00 AM
- User Preference: Sunrise views
- Flight Duration: ~6 hours

CURRENT SEAT OPTIONS:
- 12A (Left window, row 12)
- 15A (Left window, row 15)
- 8A (Left window, row 8)

TASK: Analyze these seats and recommend the BEST option for sunrise viewing. Consider:
1. Aircraft layout and seat positioning
2. Sun position during flight time
3. Optimal viewing angles
4. Row positioning advantages/disadvantages
5. Alternative seat suggestions if current options aren't ideal

Provide your reasoning and final recommendation.
```

Chose mistral 0.3v but had issues

Mistral-7B-Instruct-v0.2 going with this model
now code this ill give you the api key later

Perfect! Let me implement the Hugging Face integration with Mistral-7B-Instruct-v0.2. I'll create the AI-powered seat recommendation system.

Saw spi keys were in front end instead of back end despite me warning cursor to not do that



Even after this we wern getting proper recommendations so i checked in to ai prompt replies and logged them in console and noticed that prompt size was too big for this llm

Kept switching llms but none seem to work and i ran out of cursor and open ai tokens to do any more work.