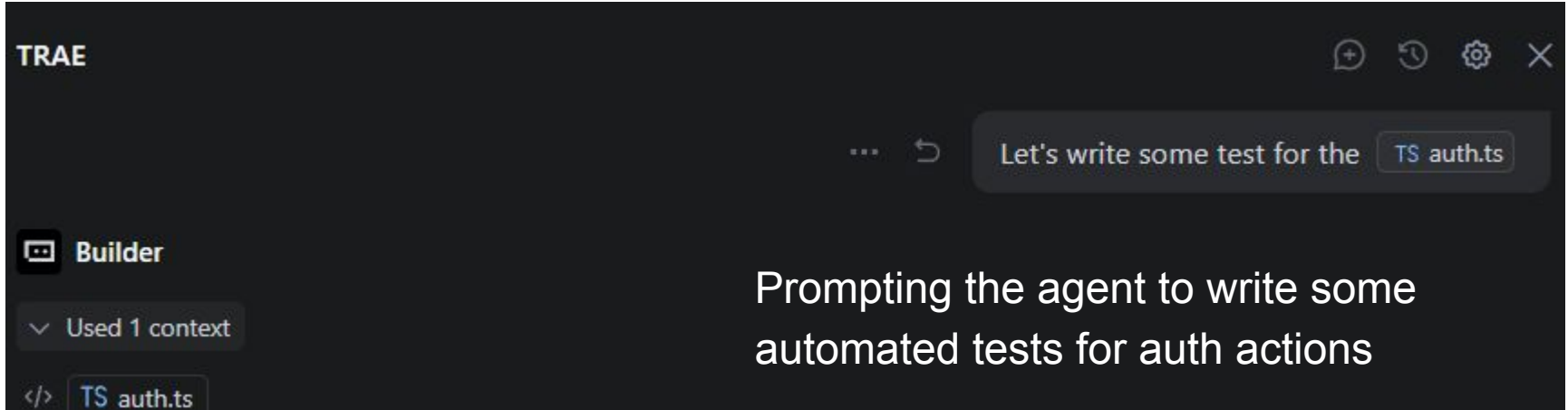


# ALX\_AIDEV

#doinghardthings

# Automated Test Generation - Testing Auth module



# Automated Test Generation - Testing Auth module

alx-polly\lib\supabase\client.ts 1-20

Now I'll set up the testing environment by installing Jest and related dependencies, then create a test file for auth.ts.

ALX\_AIDEV

Agent installs jest library

Canceled

```
npm install --save-dev jest @types/jest ts-jest jest-environment-jsdom @testing-library/react @testing-library/jest-dom
```

# Automated Test Generation - Testing Auth module

VS Code Explorer sidebar showing project structure:

- alx-polly
  - lib
    - actions
      - \_\_tests\_\_
        - auth-error-m... (U)
        - auth.test.ts (U)
        - README.md (U)
        - auth.ts
  - supabase
    - auth-utils.ts
    - utils.ts
  - node\_modules
  - public
  - types
  - .env.local
  - .gitignore
  - ALX\_AIDEV - Context and...
  - ALX\_AIDEV-Scaffolding a...
  - ALX\_AIDEV-Using @ base...
  - JS eslint.config.mjs
  - JS jest.config.js (U)
  - JS jest.setup.js (U)
  - TS middleware.ts

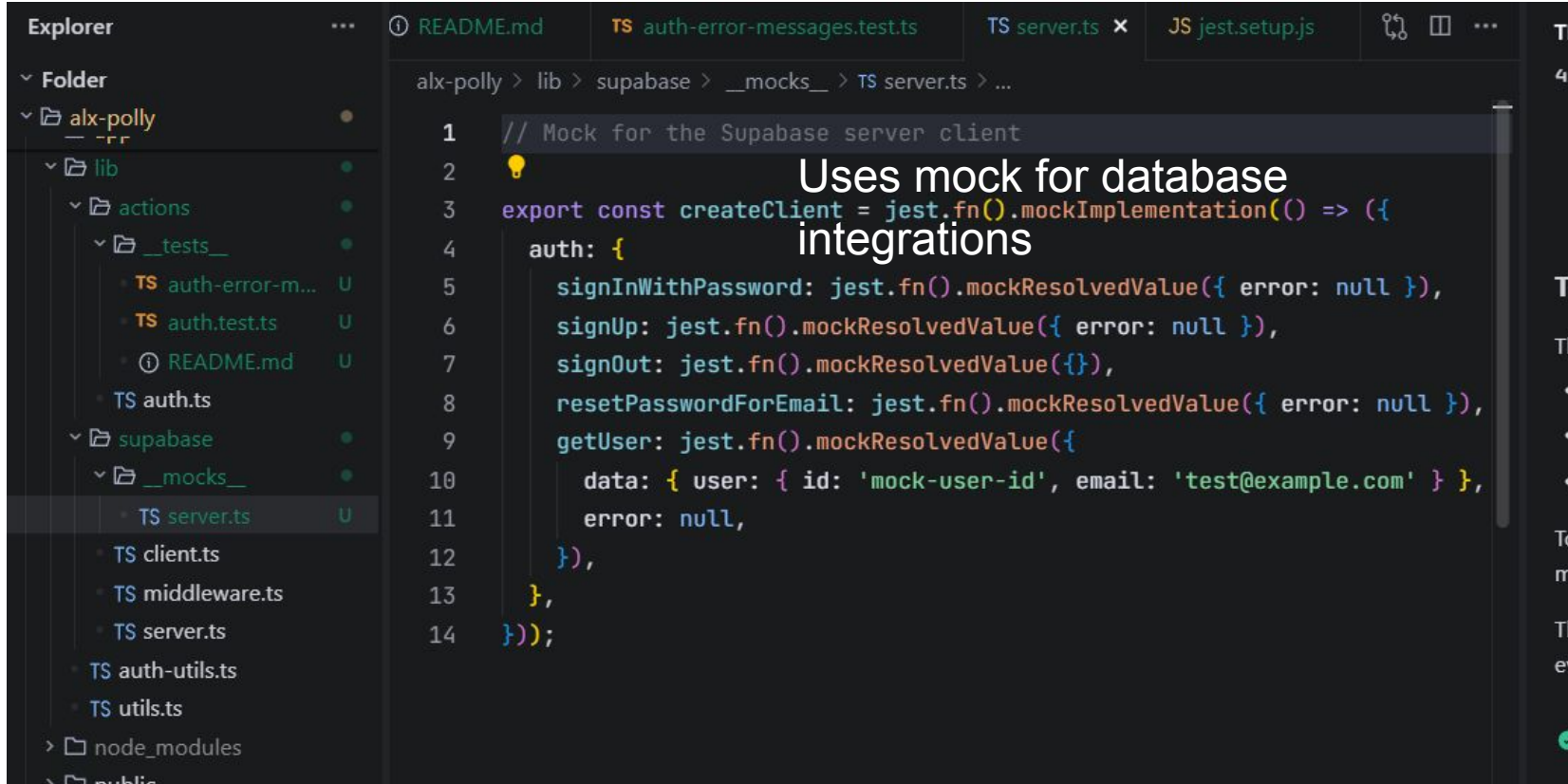
Main editor content (auth.test.ts):

```
1+ import { AuthError } from '@supabase/supabase-js';
2+
3+ // Import the function indirectly since it's not exported
4+ // We'll use a workaround to test it
5+ import * as authModule from '../auth';
6+
7+ // Access the private function using type assertion
8+ const getAuthErrorMessage = (authModule as any).getAuthErrorMessage;
9+
10+ describe('getAuthErrorMessage', () => {
11+   it('should return appropriate message for invalid login credentials', () => {
12+     const error = { message: 'Invalid login credentials' } as AuthError;
13+     expect(getAuthErrorMessage(error)).toBe('Invalid email or password');
14+   });
15+
16+   it('should return appropriate message for unconfirmed email', () => {
17+     const error = { message: 'Email not confirmed' } as AuthError;
18+     expect(getAuthErrorMessage(error)).toBe('Please check your email');
19+   });
20+
21+   it('should return appropriate message for already registered user', () => {
22+     const error = { message: 'User already registered' } as AuthError;
23+     expect(getAuthErrorMessage(error)).toBe('User already registered');
24+   });
25+ });
```

Right editor content (auth.test.ts):

```
7+ jest.mock('@lib/supabase/server', () => ({
8+   createClient: jest.fn(),
9+ }));
10+
11+ jest.mock('next/navigation', () => ({
12+   redirect: jest.fn(),
13+ }));
14+
15+ jest.mock('next/headers', () => ({
16+   cookies: jest.fn(),
17+ }));
18+
19+ describe('Auth Actions', () => {
20+   // Reset mocks before each test
21+   beforeEach(() => {
22+     jest.clearAllMocks();
23+   });
24+
25+   describe('signInAction', () => {
26+     it('should redirect to login page with error when email is', () => {
27+       // Arrange
28+       const formData = new FormData();
```

# Automated Test Generation - Testing Auth module



The screenshot shows a VS Code editor with a project structure on the left and a TypeScript file on the right. The project structure includes a folder named `alx-polly` with subfolders `lib` and `node_modules`. The `lib` folder contains `actions` and `supabase` subfolders. The `supabase` folder contains `__mocks__` and `TS server.ts`. The `__mocks__` folder contains `TS server.ts`. The `TS server.ts` file is open in the editor, showing a mock for the Supabase server client. The code uses Jest's `jest.fn().mockImplementation()` to mock the `createClient` function, which returns an object with `auth` methods like `signInWithPassword`, `signUp`, `signOut`, `resetPasswordForEmail`, and `getUser`. The `getUser` method returns a mock user object with `id` and `email` properties. The code is annotated with a lightbulb icon and the text "Uses mock for database integrations".

```
1 // Mock for the Supabase server client
2
3 export const createClient = jest.fn().mockImplementation(() => ({
4   auth: {
5     signInWithPassword: jest.fn().mockResolvedValue({ error: null }),
6     signUp: jest.fn().mockResolvedValue({ error: null }),
7     signOut: jest.fn().mockResolvedValue({}),
8     resetPasswordForEmail: jest.fn().mockResolvedValue({ error: null }),
9     getUser: jest.fn().mockResolvedValue({
10       data: { user: { id: 'mock-user-id', email: 'test@example.com' } },
11       error: null,
12     }),
13   },
14 }));
```

# Automated Test Generation - Testing Auth module

