**FieldTrax Testing Guide**

**Frontend Testing**

**1. Start the Frontend Development Server**

bash

Copy

cd frontend

npm run dev

**2. Test Authentication Flow**

1. Open http://localhost:5173 in your browser
2. Test Registration/Login:
   * Try logging in with invalid credentials
   * Try registering a new account
   * Try logging in with valid credentials
3. Test Password Reset Flow:

Copy

http://localhost:5173/login

Click "Forgot Password?"

Enter email

Check console for reset link (in development)

Test password reset form

Try logging in with new password

1. Test Protected Routes:

Copy

/dashboard (should redirect to login if not authenticated)

/work-orders/new (should redirect to login if not authenticated)

**Backend Testing**

**1. Set up Test Environment**

First, create a .env.test file:

env

Copy

# Database

DATABASE\_URL=sqlite:///./test.db

# JWT Settings

SECRET\_KEY=test-secret-key

ALGORITHM=HS256

ACCESS\_TOKEN\_EXPIRE\_MINUTES=30

# Email Settings

MAIL\_USERNAME=test@example.com

MAIL\_PASSWORD=test-password

MAIL\_FROM=test@fieldtrax.com

MAIL\_PORT=587

MAIL\_SERVER=smtp.gmail.com

# Frontend URL

FRONTEND\_URL=http://localhost:5173

# Test Mode

TEST\_MODE=True

**2. Start the Backend Server**

bash

Copy

cd backend

uvicorn main:app --reload

**3. Test API Endpoints using curl or Postman**

Test Authentication:

bash

Copy

*# Login*

curl -X POST http://localhost:8000/api/auth/login \

-H "Content-Type: application/json" \

-d '{"username": "test@example.com", "password": "password123"}'

*# Request Password Reset*

curl -X POST http://localhost:8000/api/auth/password/reset-request \

-H "Content-Type: application/json" \

-d '{"email": "test@example.com"}'

*# Verify Reset Token*

curl -X GET http://localhost:8000/api/auth/password/verify-token/{token}

*# Reset Password*

curl -X POST http://localhost:8000/api/auth/password/reset \

-H "Content-Type: application/json" \

-d '{"token": "your-token", "new\_password": "newpassword123"}'

**4. Automated Testing**

1. Install testing dependencies:

bash

Copy

*# Frontend*

cd frontend

npm install -D vitest @testing-library/react @testing-library/jest-dom jsdom

*# Backend*

cd backend

pip install pytest pytest-asyncio httpx pytest-cov

1. Run the tests:

bash

Copy

*# Frontend tests*

npm test

*# Backend tests*

pytest

**Test Cases**

**Frontend Test Cases:**

javascript

Copy

*// src/tests/auth/LoginForm.test.jsx*

import { describe, it, expect, vi } from 'vitest';

import { render, screen, fireEvent, waitFor } from '@testing-library/react';

import { LoginForm } from '../../components/auth/LoginForm';

describe('LoginForm', () => {

it('should show error message on invalid credentials', async () => {

*// Test setup & assertions*

});

it('should redirect on successful login', async () => {

*// Test setup & assertions*

});

});

**Backend Test Cases:**

python

Copy

*# backend/tests/test\_auth.py*

import pytest

from httpx import AsyncClient

from main import app

@pytest.mark.asyncio

async def test\_login():

async with AsyncClient(app=app, base\_url="http://test") as ac:

response = await ac.post("/api/auth/login", json={

"username": "test@example.com",

"password": "password123"

})

assert response.status\_code == 200

assert "token" in response.json()

@pytest.mark.asyncio

async def test\_password\_reset\_request():

async with AsyncClient(app=app, base\_url="http://test") as ac:

response = await ac.post("/api/auth/password/reset-request", json={

"email": "test@example.com"

})

assert response.status\_code == 200

**Manual Testing Checklist**

**Authentication**

* User can register with valid credentials
* User cannot register with invalid email
* User cannot register with weak password
* User can login with valid credentials
* User cannot login with invalid credentials
* User can request password reset
* User can reset password with valid token
* User cannot reset password with invalid token
* User can logout

**Protected Routes**

* Unauthenticated users are redirected to login
* Authenticated users can access protected routes
* Token expiration is handled correctly
* Token refresh works correctly

**Form Validation**

* Email validation works
* Password requirements are enforced
* Error messages are displayed correctly
* Loading states are displayed correctly
* Success messages are displayed correctly

**Debugging Tips**

1. Frontend Debugging:

javascript

Copy

*// Enable detailed logging*

localStorage.setItem('debug', 'fieldtrax:\*');

*// Check auth state*

console.log(localStorage.getItem('token'));

1. Backend Debugging:

python

Copy

*# Enable debug mode in FastAPI*

app = FastAPI(debug=True)

*# Add logging*

import logging

logging.basicConfig(level=logging.DEBUG)

1. Network Debugging:

* Use browser DevTools Network tab
* Check CORS settings
* Verify request/response headers
* Monitor WebSocket connections

**Common Issues and Solutions**

1. CORS Issues:

python

Copy

*# backend/main.py*

app.add\_middleware(

CORSMiddleware,

allow\_origins=["http://localhost:5173"],

allow\_credentials=True,

allow\_methods=["\*"],

allow\_headers=["\*"],

)

1. Token Issues:

javascript

Copy

*// Check token expiration*

const decoded = jwtDecode(token);

if (decoded.exp \* 1000 < Date.now()) {

*// Token expired*

}

1. Database Issues:

bash

Copy

*# Reset database*

cd backend

rm test.db

python init\_db.py

Would you like me to:

1. Add more detailed test cases?
2. Create a test data generator?
3. Add performance testing?
4. Add end-to-end testing with Cypress?
5. Add error simulation tests?