**📚 Beginner-Friendly Guide for Uniswap Web3 UI**

This guide will provide extremely detailed, beginner-friendly instructions for building your Uniswap Web3 UI application using Scaffold-ETH. We'll cover everything from setting up your computer, installing the right tools, deploying your contracts to the test network, building the frontend UI, and deploying your application on Vercel.

We'll break down the process into clear steps, making sure you can follow along easily. You will take screenshots at important steps to document your progress and ensure everything works correctly.

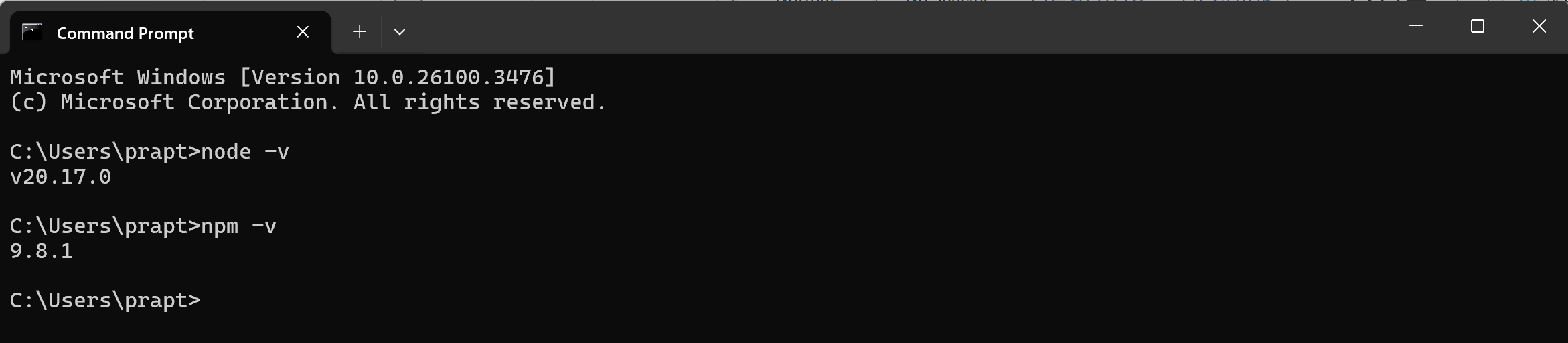
**✅ Step 1: Setting Up Your Development Environment**

**📌 Install Node.js and npm (JavaScript Runtime and Package Manager)**

1. **What is Node.js?** It's a tool that allows your computer to run JavaScript code outside a web browser. It's essential for building and running your application.
2. **Installing Node.js:**
   * Go to [Node.js official website](https://nodejs.org/).
   * Click on the green button that says **'LTS (Recommended for most users)'**.
   * This will download the installer file (e.g., node-v18.x.x-x64.msi for Windows).
   * Open the installer and click through the prompts (**Next > I accept the agreement > Next > Next > Install > Finish**).

**📌 Verify Installation**

1. **Open your Terminal:**
   * **Windows:** Press Win + R, type cmd, and press Enter.
   * **Mac/Linux:** Open your Terminal application.
2. **Check Node.js version:** Type node -v and press Enter. It should show something like v18.x.x.
3. **Check npm version:** Type npm -v and press Enter. It should show something like 8.x.x.

****

**✅ Step 2: Installing Yarn (Package Manager)**

**📌 What is Yarn?**

Yarn is a package manager that helps you download and manage all the packages (libraries) you need for your application. It’s faster and more reliable than npm for larger projects.

**📌 Install Yarn**

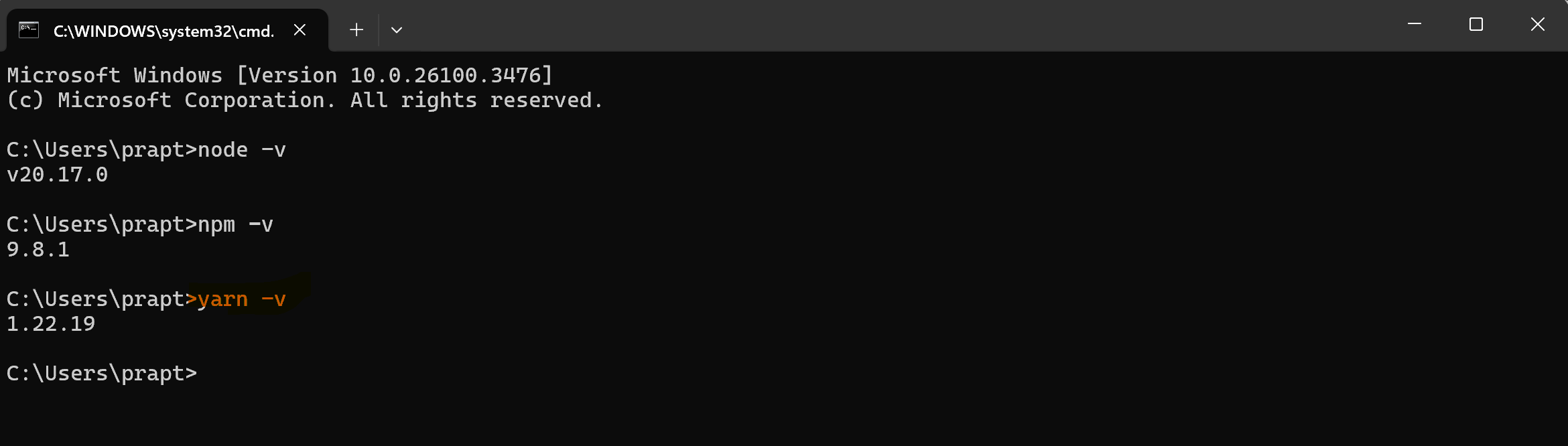
1. **Open your Terminal (cmd on Windows, Terminal on Mac/Linux)**.
2. **Type the following command and press Enter:**

npm install -g yarn

1. **Verify installation:**

yarn -v

You should see a version number like 1.x.x or 2.x.x.

****

**✅ Step 3: Installing Git (Version Control Tool)**

**📌 What is Git?**

Git is a tool that helps you download and manage code from the internet (like GitHub). It's essential for getting Scaffold-ETH.

**📌 Install Git**

1. **Windows Users:**
   * Go to [Git for Windows](https://gitforwindows.org/) and download the installer.
   * Open the installer and click through all the default options (Click Next until you see Install).
   * After installation, click Finish.
2. **Mac Users:**
   * Open Terminal and type:

brew install git

(If you don't have brew, install it from [Homebrew](https://brew.sh/)).

1. **Linux Users:**

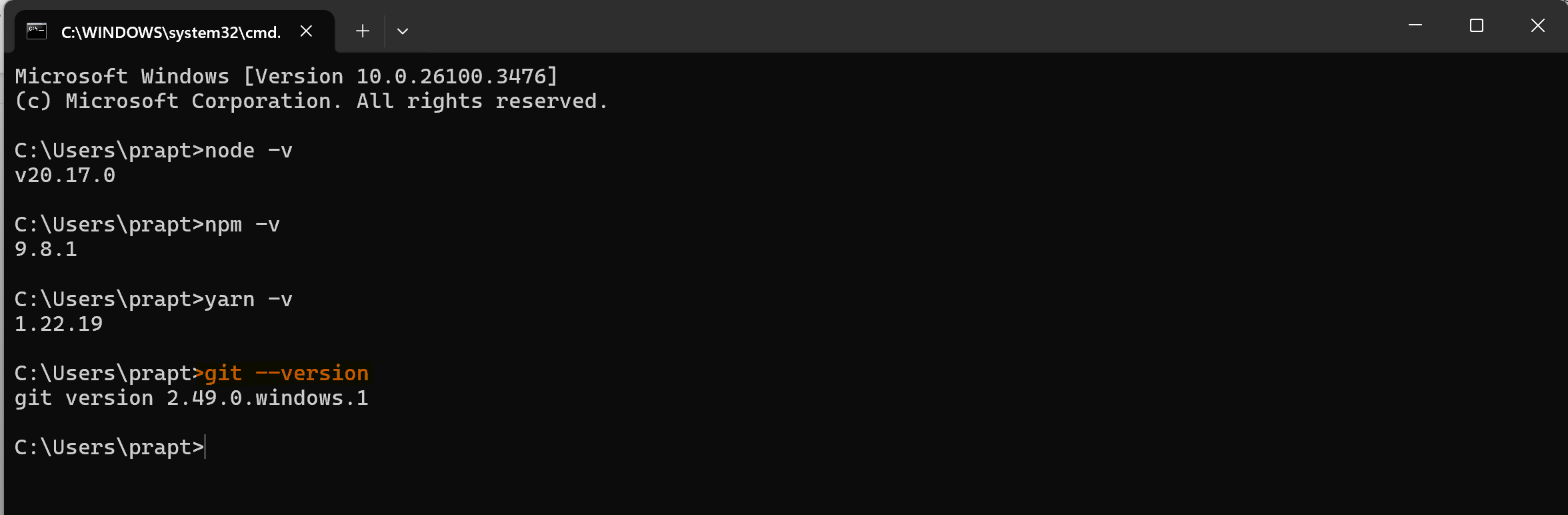
sudo apt-get update

sudo apt-get install git

1. **Verify installation:**

git --version

You should see something like git version 2.x.x.

****

**✅ Step 4: Cloning Scaffold-ETH**

**📌 What is Scaffold-ETH?**

Scaffold-ETH is a pre-built Ethereum development framework that provides an easy-to-use template for building decentralized applications (dApps).

**📌 Clone the Repository (Download the code)**

1. **Open your Terminal (cmd/Terminal/PowerShell)**.
2. **Navigate to the folder where you want to download Scaffold-ETH.**
   * For example, if you want to download it to your Desktop, type:

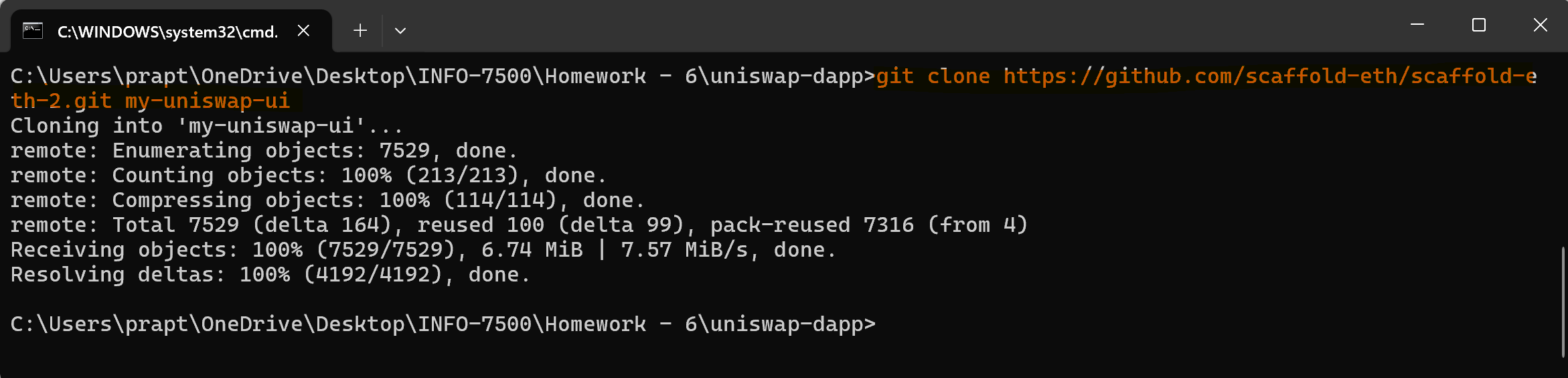
cd Desktop

1. **Download Scaffold-ETH:**

git clone https://github.com/scaffold-eth/scaffold-eth-2.git my-uniswap-ui

1. **Move into the project folder:**

cd my-uniswap-ui



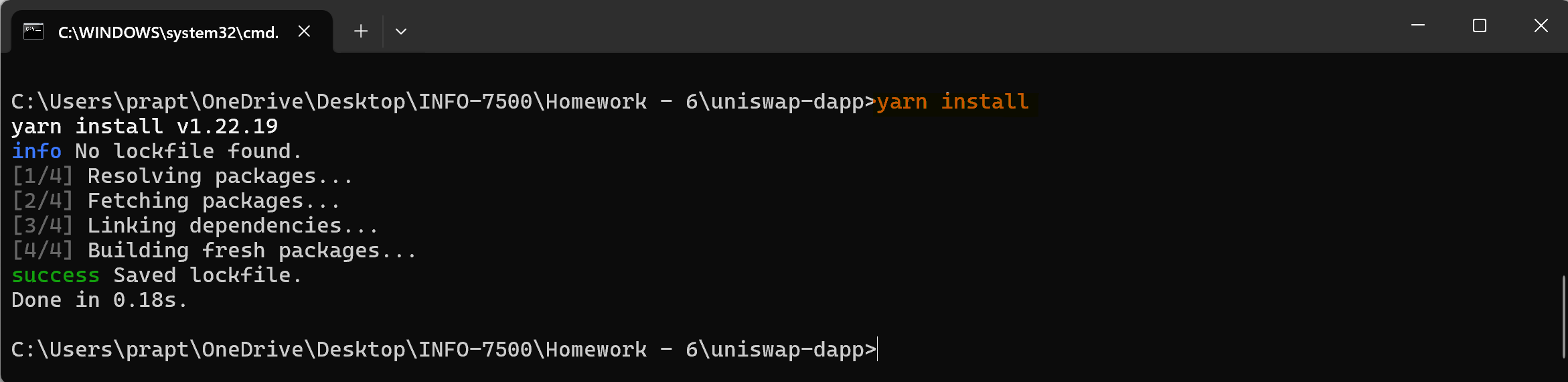
**✅ Step 5: Installing Dependencies**

**📌 Installing Required Packages**

1. **Open your terminal in the my-uniswap-ui folder.**
2. **Run the following command to install all necessary packages:**

yarn install

This command will take a few minutes as it downloads all required packages (React, Ethers.js, Hardhat, etc.).

****

**✅ Step 6: Running Scaffold-ETH Locally**

**📌 Starting the Application**

1. **Start the local Ethereum blockchain:**

yarn chain

1. **Start the React front-end application:**

yarn dev

1. **Open your browser and visit:**

http://localhost:3000

You should see the Scaffold-ETH interface running on your machine.

**📸 Screenshot:** Take a screenshot of the webpage running on localhost:3000.

**✅ Step 7: Connecting MetaMask to Scaffold-ETH**

**📌 Install MetaMask**

1. **Download MetaMask Extension:**
   * Open Google Chrome.
   * Go to [MetaMask Download Page](https://metamask.io/download.html).
   * Click on Install MetaMask for Chrome.
2. **Set Up MetaMask Wallet:**
   * Click the MetaMask icon in your browser toolbar.
   * Click Get Started > Create a Wallet > Agree to Terms.
   * Set up a strong password.
   * Save your Secret Recovery Phrase in a safe place.
3. **Enable Goerli Test Network:**
   * Open MetaMask and click the network selector (shows 'Ethereum Mainnet' by default).
   * Click Show/Hide test networks.
   * Toggle Show test networks to ON.
   * Select Goerli Test Network from the list.
4. **Get Test ETH:**
   * Visit [QuickNode Multi-Chain Faucet](https://faucet.quicknode.com/) or [Goerli Faucet](https://goerlifaucet.com/).
   * Paste your wallet address and click Send me test ETH.

**📸 Screenshot:** Take a screenshot of MetaMask showing connected to the Goerli Test Network with test ETH.

**📚 Beginner-Friendly Guide for Uniswap Web3 UI - Integration & Components**

Now that we've set up your development environment, deployed contracts, and connected MetaMask, let's proceed with integrating your deployed contracts into the frontend UI and building the necessary components.

**✅ Step 8: Integrating Deployed Contracts with Frontend UI**

**📌 Setting Up Ethers.js for Contract Interaction**

1. **Install Ethers.js:**

yarn add ethers

1. **Update App.jsx or App.tsx (wherever you have your main React code):**

import { ethers } from 'ethers';

// Connect to MetaMask

async function connectWallet() {

if (window.ethereum) {

const provider = new ethers.providers.Web3Provider(window.ethereum);

await provider.send('eth\_requestAccounts', []);

const signer = provider.getSigner();

return { provider, signer };

} else {

alert('MetaMask not installed!');

}

}

**📌 Adding Deployed Contract Address & ABI**

1. **Copy your deployed contract addresses (e.g., Uniswap Pool, Token Contracts) from Goerli deployment.**
2. **Create a config.js file:**

export const UNISWAP\_CONTRACT\_ADDRESS = 'YOUR\_UNISWAP\_CONTRACT\_ADDRESS';

export const TOKEN\_A\_ADDRESS = 'YOUR\_TOKEN\_A\_CONTRACT\_ADDRESS';

export const TOKEN\_B\_ADDRESS = 'YOUR\_TOKEN\_B\_CONTRACT\_ADDRESS';

1. **Create a UniswapABI.json file:**

* Copy the ABI (Application Binary Interface) of your Uniswap contract from your deployment folder (e.g., artifacts/contracts/YourContract.json).
* Paste the ABI into UniswapABI.json.

**✅ Step 9: Creating UI Components**

**📌 Deposit Component**

import React, { useState } from 'react';

import { ethers } from 'ethers';

import { UNISWAP\_CONTRACT\_ADDRESS } from './config';

import UniswapABI from './UniswapABI.json';

const Deposit = () => {

const [amountA, setAmountA] = useState('');

const [amountB, setAmountB] = useState('');

const handleDeposit = async () => {

const { provider, signer } = await connectWallet();

const uniswapContract = new ethers.Contract(UNISWAP\_CONTRACT\_ADDRESS, UniswapABI, signer);

await uniswapContract.deposit(ethers.utils.parseEther(amountA), ethers.utils.parseEther(amountB));

};

return (

<div>

<h2>Deposit Liquidity</h2>

<input type="text" placeholder="Amount A" value={amountA} onChange={(e) => setAmountA(e.target.value)} />

<input type="text" placeholder="Amount B" value={amountB} onChange={(e) => setAmountB(e.target.value)} />

<button onClick={handleDeposit}>Deposit</button>

</div>

);

};

export default Deposit;

**📌 Swap Component**

import React, { useState } from 'react';

import { ethers } from 'ethers';

import { UNISWAP\_CONTRACT\_ADDRESS } from './config';

import UniswapABI from './UniswapABI.json';

const Swap = () => {

const [amount, setAmount] = useState('');

const handleSwap = async () => {

const { provider, signer } = await connectWallet();

const uniswapContract = new ethers.Contract(UNISWAP\_CONTRACT\_ADDRESS, UniswapABI, signer);

await uniswapContract.swap(ethers.utils.parseEther(amount));

};

return (

<div>

<h2>Swap Tokens</h2>

<input type="text" placeholder="Amount" value={amount} onChange={(e) => setAmount(e.target.value)} />

<button onClick={handleSwap}>Swap</button>

</div>

);

};

export default Swap;

**✅ Step 10: Visualizing Data**

**📌 Adding Charts**

We will use **Recharts**, a powerful chart library for React.

1. **Install Recharts:**

yarn add recharts

1. **Creating a Pool Reserves Curve Visualization:**

import React from 'react';

import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, Legend, ResponsiveContainer } from 'recharts';

const PoolReservesChart = ({ data }) => {

return (

<ResponsiveContainer width="100%" height={400}>

<LineChart data={data}>

<CartesianGrid strokeDasharray="3 3" />

<XAxis dataKey="timestamp" />

<YAxis />

<Tooltip />

<Legend />

<Line type="monotone" dataKey="reserveA" stroke="#8884d8" />

<Line type="monotone" dataKey="reserveB" stroke="#82ca9d" />

</LineChart>

</ResponsiveContainer>

);

};

export default PoolReservesChart;

**✅ Step 11: Deploying to Vercel**

1. **Push your code to GitHub:**

git init

git add .

git commit -m "Initial commit"

git branch -M main

git remote add origin https://github.com/YOUR\_USERNAME/YOUR\_REPOSITORY.git

git push -u origin main

1. **Deploy to Vercel:**
   * Go to [Vercel Dashboard](https://vercel.com/).
   * Click New Project and select your GitHub repository.
   * Click Deploy.
2. **Copy the URL provided by Vercel and test your deployed application!**

I'll continue by adding more components for redeeming, pooling, and improving the visualization. Let me proceed! 😊