 **Ethereum** → One of the most popular blockchains where you can build apps.

 **Sepolia** → A **practice playground version of Ethereum** (called a testnet). It looks and works like the real Ethereum but uses fake money (test ETH) — so you don’t lose real money while testing.

 **MetaMask** → A **digital wallet app** (browser extension/mobile app) used to store your blockchain account. You need it to log in, send money, and connect with apps.

 **Remix IDE** → A **website where you write and deploy smart contracts** (the blockchain programs). It makes it easy to test and upload your code to Ethereum or Sepolia.

 **Smart Contract** → A **robot agreement** on blockchain. Once conditions are met, it acts automatically. Example: hold tourist payment until guide finishes the tour → then release it.

 **ABI** (Application Binary Interface) → A kind of **menu card** of your contract. It tells your Flutter app what functions exist in the blockchain program.

 **RPC Provider (like Infura or Alchemy)** → A **bridge** that connects your Flutter app to the blockchain (since your phone can’t talk to Ethereum directly).

**Your Project in Plain Words**

You’re building **Pravass**, a tourism app.  
You want to use blockchain for three things:

1. **Guide Verification** → tourists can trust that guides are genuine.
2. **Handicraft Verification** → tourists can trust that crafts are real, not fake.
3. **Secure Payments** → tourist pays money, and it’s only released when service is done.
4.  **Flutter** (to make your app) → install Flutter SDK + Android Studio/VS Code.
5.  **MetaMask** (for your blockchain account) → browser extension or phone app.
6.  **Remix IDE** (no install, just open in browser: https://remix.ethereum.org).
7.  **RPC Provider** (Infura or Alchemy free account) → gives you an API link to connect Flutter → blockchain.
8.  **Git + Node.js** (optional, only if you want extra blockchain dev tools later)

Step-by-Step Overview

 **Set up Flutter app** → basic app working on your emulator/phone.

 **Install MetaMask** → create wallet (write down secret recovery phrase, don’t lose it). Switch to Sepolia Testnet (the free practice network).

 **Get test ETH** → from Sepolia faucet (free play money). This will let you test without using real money.

 **Write smart contracts** → (in Remix IDE). These are tiny programs for your features (guide verification, craft verification, secure payments).

 **Deploy contracts** → from Remix to Sepolia. This puts your program onto blockchain. After this, you’ll get:

* Contract Address (like house number of your contract).
* ABI (the menu card of functions).

 **Copy ABI + Address** → save them for Flutter side.

 **Connect Flutter with blockchain** → use a package like web3dart. In your app, paste the ABI + Address + RPC link (from Infura/Alchemy).

 **Use MetaMask account** → when your app sends a transaction, MetaMask wallet will ask for confirmation (like “Do you want to pay?”).

 **Test your app** →

* Tourist sees “Verified Guide”.
* Craft shows “Verified Original”.
* Payment flow works (money locked until guide completes tour).

**Security**

* **Private Key** = your blockchain password. Never paste this in your app. For demo, it’s okay, but in real app always let user log in with MetaMask.
* **Hacking Risk** = if private keys are stored in your app code, hackers can steal money. Solution: use WalletConnect/MetaMask SDK in real app.
* **Reentrancy Attack** = a known smart contract bug in payments. Use OpenZeppelin templates to stay safe

## Tips to Avoid Problems While Deploying

1. Always select **Sepolia Testnet** in MetaMask (not Mainnet).
2. Always copy **new contract address + ABI** after each deploy.
3. Use only **test ETH** (free) for practice.
4. Double check **Flutter connects to Infura/Alchemy RPC URL**.
5. Don’t share your **private key** with anyone.
6.  Flutter = your app frontend.
7.  Smart Contracts = rules/logic on blockchain.
8.  MetaMask = wallet/login.
9.  Infura/Alchemy = internet bridge.
10.  Sepolia = safe playground to test.