1. **Requirements**:  
   - Node.js (v14.0 or higher)  
   - Testnet cryptocurrency wallets (e.g., Bitcoin, Ethereum)  
   - PayPal Developer account and sandbox credentials  
  
2. **Set up the project folder and initialize it:**  
  
apt install npm

mkdir crypto-exchange  
cd crypto-exchange

npm init -y  
  
**3. Install required packages:**  
  
npm install express body-parser axios dotenv web3  
```

4. Go to **Paypal Developer Dashboard to create a new sandbox account**. In the dashboard, create a new sandbox app and retrieve the client ID and secret for the app.

5. Choose a **cryptocurrency testnet wallet** (Bitcoin or Etherum), and obtain an API key from the testnet provider like Infura or **BlockCypher**

6. **Create a `.env` file to store your credentials:**  
  
PAYPAL\_CLIENT\_ID=<your\_paypal\_client\_id>  
PAYPAL\_SECRET=<your\_paypal\_secret>  
```  
  
7. **Create an `app.js` file and add the following code:**  
const express = require("express");  
const bodyParser = require("body-parser");  
const axios = require("axios");  
const dotenv = require("dotenv");  
  
dotenv.config();  
  
const app = express();  
app.use(bodyParser.json());  
  
// PayPal API URL and credentials  
const PAYPAL\_API\_URL = "[https://api-m.sandbox.paypal.com](https://api-m.sandbox.paypal.com/)";  
const PAYPAL\_CLIENT\_ID = process.env.PAYPAL\_CLIENT\_ID;  
const PAYPAL\_SECRET = process.env.PAYPAL\_SECRET;  
  
// Testnet cryptocurrency wallets  
const wallets = {  
  bitcoin: "your\_testnet\_bitcoin\_wallet\_address",  
  ethereum: "your\_testnet\_ethereum\_wallet\_address",  
};  
  
// Helper function to get a PayPal access token  
async function getPayPalAccessToken() {  
  const response = await [axios.post](http://axios.post/" \t "_blank)(  
    `${PAYPAL\_API\_URL}/v1/oauth2/token`,  
    "grant\_type=client\_credentials",  
    {  
      headers: {  
        "Content-Type": "application/x-www-form-urlencoded",  
        Authorization: `Basic ${Buffer.from(  
          `${PAYPAL\_CLIENT\_ID}:${PAYPAL\_SECRET}`  
        ).toString("base64")}`,  
      },  
    }  
  );  
  
  return response.data.access\_token;  
}  
  
// API endpoint for creating a PayPal order  
[app.post](http://app.post/)("/create-order", async (req, res) => {  
  try {  
    const { currency, amount } = req.body;  
    const access\_token = await getPayPalAccessToken();  
  
    const response = await [axios.post](http://axios.post/" \t "_blank)(  
      `${PAYPAL\_API\_URL}/v2/checkout/orders`,  
      {  
        intent: "CAPTURE",  
        purchase\_units: [  
          {  
            amount: {  
              currency\_code: currency,  
              value: amount,  
            },  
          },  
        ],  
      },  
      {  
        headers: {  
          "Content-Type": "application/json",  
          Authorization: `Bearer ${access\_token}`,  
        },  
      }  
    );  
  
    res.json(response.data);  
  } catch (error) {  
    console.error(error);  
    res.status(500).json({ error: "Failed to create order" });  
  }  
});  
  
// API endpoint for capturing a PayPal order  
[app.post](http://app.post/)("/capture-order", async (req, res) => {  
  try {  
    const { orderID } = req.body;  
    const access\_token = await getPayPalAccessToken();  
  
    const response = await [axios.post](http://axios.post/" \t "_blank)(  
      `${PAYPAL\_API\_URL}/v2/checkout/orders/${orderID}/capture`,  
      {},  
      {  
        headers: {  
          "Content-Type": "application/json",  
          Authorization: `Bearer ${access\_token}`,  
        },  
      }  
    );  
  
    // Simulate sending cryptocurrency to the user's wallet  
    console.log(  
      `Sent ${response.data.purchase\_units[0].amount.value} ${response.data.purchase\_units[0].amount.currency\_code} to wallet: ${wallets.bitcoin}`  
    );  
  
    res.json(response.data);  
  } catch (error) {  
    console.error(error);  
    res.status(500).json({ error: "Failed to capture order" });  
  }  
});  
  
// Start the server  
const port = process.env.PORT || 3000;  
app.listen(port, () => {  
  console.log(`Server running on port ${port}`);  
});  
```

8) **Test your exchange system**

Start your server by running node server.js

Now you have a simple cryptocurrency exchange system with e-payment through PayPal using testnet and sandbox accounts. You can expand this example by adding more functionality, such as handling different cryptocurrencies, checking wallet balances, and integrating with real cryptocurrency APIs.