**Real-time Currency Converter**

**MAGDANGAL, JOHN MHYK A.**

**MALIT, PAULEEN KEI S.**

**SANCHEZ, EUNICE JAZMINE S.**

**DALE, KARL KRISTAN R.**

**Real-Time Currency Converter using API and WebSocket Server**

For our final project in Web Development this semester, our group decided to build a Real-Time Currency Converter using a WebSocket Server, ExchangeRate-API and CoinGecko API. This project will showcase our skills in using APIs, fetching data, and understanding the intricacies of both client-side and server-side development.

The main goal of this project is to apply what we've learned throughout the course. We want to demonstrate our ability to work with APIs, manage data in real-time, and effectively utilize a server to create a seamless user experience. We believe this currency converter project is the perfect way to put our knowledge into practice, and we are confident in our ability to successfully complete it.

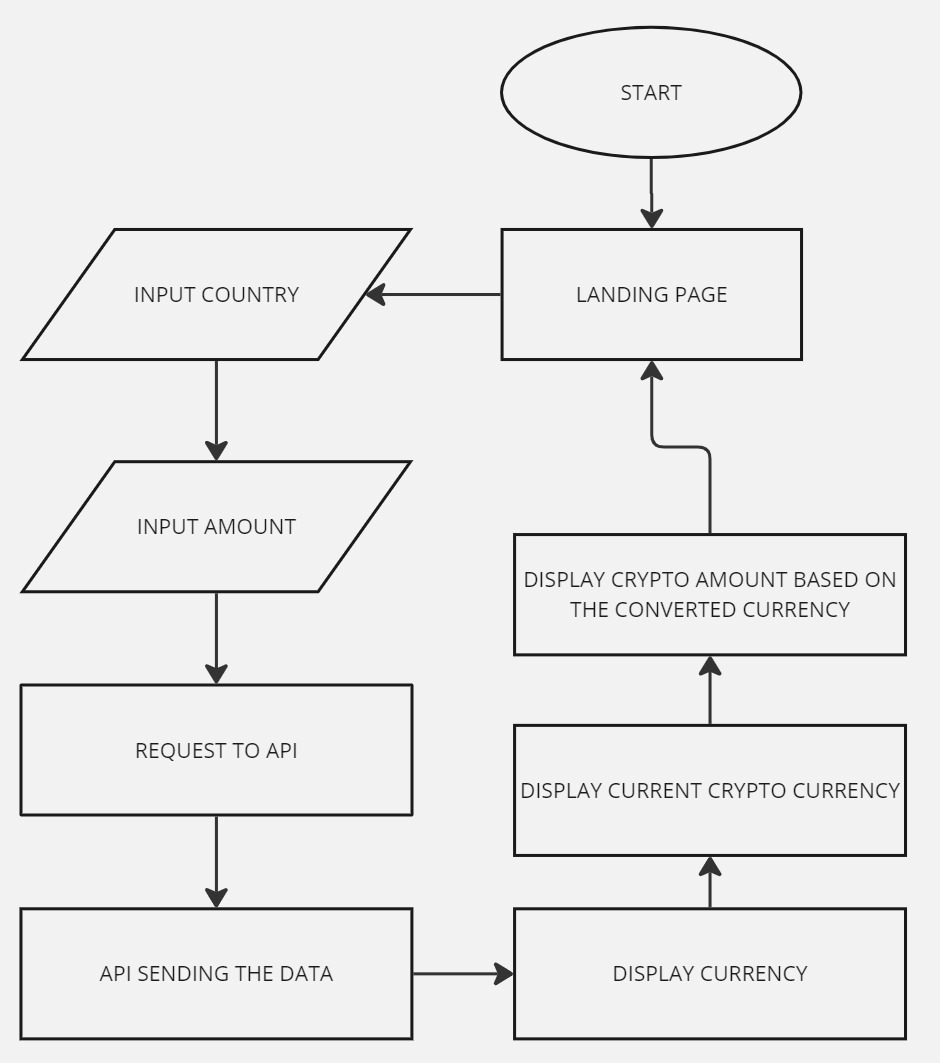
**Main Functionality**

The main purpose of our software is to provide users with a seamless and educational experience in currency conversion, bridging the gap between traditional fiat currencies and cryptocurrencies. By leveraging the power of APIs and real-time updates, our application offers users the ability to convert currencies from different countries and understand their value in popular cryptocurrencies such as Bitcoin, Litecoin, Ethereum, and Dogecoin.

Our software excels in its ability to convert currencies from various countries, allowing users to explore and understand the value of money across different regions of the world. With just a few clicks, users can input an amount in their desired currency and instantly see its equivalent value in another currency of their choice. In addition to traditional currency conversion, our application provides users with valuable insights into the world of cryptocurrencies. By fetching real-time data from the CoinGecko API, users can not only see the equivalent value of their currency in cryptocurrencies but also learn about the current prices of popular digital assets like Bitcoin, Litecoin, Ethereum, and Dogecoin.One of the key features of our project is its educational component. Beyond just providing currency conversion services, our software aims to educate users about the intricacies of both fiat currencies and cryptocurrencies. Through interactive displays and real-time updates, users can gain valuable knowledge about the conversion rates, market trends, and purchasing power of different currencies and cryptocurrencies.

A standout feature of our application is its ability to calculate how much cryptocurrency a user can purchase with their converted currency. By leveraging the exchange rates provided by the ExchangeRate API and the current cryptocurrency prices from the CoinGecko API, our software empowers users to make informed decisions about their investments in the digital asset space.

**FLOWCHART**



**CODE**

**Client-Side (public/index.html, public/js/script.js)**

HTML Structure (public/index.html):

Input Fields:

* Contains an input field where users can enter the amount to convert.

Dropdowns:

* Two dropdowns for selecting the "from" and "to" currencies.
* Each dropdown displays country flags alongside currency codes.

Exchange Rate Display:

* Section to display the current exchange rate between the selected currencies.

Get Exchange Rate Button:

* Button triggers the currency conversion process.

**JavaScript Logic (public/js/script.js):**

Dropdown Population:

* Populates the dropdowns with currency options and corresponding country flags.

Flag Display:

* Updates the country flag based on the selected currency.

WebSocket Setup:

* Establishes a WebSocket connection with the server.

Event Listeners:

* Listens for changes in dropdown selections and triggers flag updates.
* Listens for button clicks to initiate the exchange rate retrieval process.

Exchange Rate Display:

* Updates the UI with the fetched exchange rate.

Currency Conversion:

* Initiates the currency conversion process and updates the UI with converted amounts in cryptocurrencies (Bitcoin, Litecoin, Ethereum, Dogecoin).

**Server-Side (server/app.js)**

WebSocket Server Setup (server/app.js):

* WebSocket Initialization:

Creates a WebSocket server using the ws library.

* Listens for client connections and messages.

Client Connection Handling:

* Logs when a client connects or disconnects from the WebSocket server.

Message Handling:

* Parses incoming messages from clients.
* Handles requests for exchange rates and sends back the calculated rates.

API Integration (server/app.js):

* ExchangeRate-API Integration:
* Fetches exchange rates from ExchangeRate-API using the provided API key.
* Updates exchange rates periodically to ensure they remain up-to-date.

CoinGecko API Integration:

* Fetches current cryptocurrency prices from CoinGecko API.
* Uses the fetched prices for currency conversion calculations.

**Currency Conversion (server/app.js):**

Exchange Rate Calculation:

* Calculates the exchange rate between two currencies based on the provided rates.
* Sends the calculated rate back to the client via WebSocket for display.

**CODE**

|  |  |
| --- | --- |
| <!DOCTYPE html>  <html lang="en" dir="ltr">    <head>      <meta charset="utf-8">      <title>Currency Converter App Using API and WebSocket</title>      <link rel="stylesheet" href="style.css">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"/>      <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>    </head>    <body>      <div class="wrapper">        <header>Currency Converter</header>        <form action="#">            <div class="amount">              <input type="text" placeholder="Enter Amount" autocomplete="off">            </div>          <div class="drop-list">            <div class="from">              <p>From</p>              <div class="select-box">                <img src="https://flagcdn.com/48x36/us.png" alt="flag">                <select> </select>              </div>            </div>            <div class="icon"><i class="fas fa-exchange-alt"></i></div>            <div class="to">              <p>To</p>              <div class="select-box">                <img src="https://flagcdn.com/48x36/np.png" alt="flag">                <select> </select>              </div>            </div>          </div>          <div class="exchange-rate">Getting exchange rate...</div>          <button>Get Exchange Rate</button>        </form>      </div>        <div class="converted-amount">        <h2>Converted Amounts</h2>        <p>Bitcoin: <span id="convertedBitcoin"></span></p>        <p>Litecoin: <span id="convertedLitecoin"></span></p>        <p>Ethereum: <span id="convertedEthereum"></span></p>        <p>Dogecoin: <span id="convertedDogecoin"></span></p>      </div>      <div class="container">        <div class="coin-price">            <div class="logo"><img src="https://blogger.googleusercontent.com/img/a/AVvXsEi32rDKQR2Swi7YfHjDzyZUBGEmxTz77OClnm24SZl7kWuls7fsVCIfAObY\_JRJIReQnBWZIPSVfDLSqDvqeu4CCXCoNQIoUGK-OSDUGMtDJFxh9vmU6WGajIgXH4CsR\_-sXU0qWbyJTJl7N0BSTB8HcAGSpCJ54G1daZPnU6h2oANo2CTGsbHCoaGJHQ" ></div>            <div>                <h1>$<span id="bitcoin"></span></h1>                <h1>Bitcoin</h1>            </div>        </div>        <div class="coin-price">            <div class="logo"><img src="https://blogger.googleusercontent.com/img/a/AVvXsEhrUFoHC68rLHQYMV41awqGtoeU6qI-CkSVmcYK-KBCrOvL-jzwLOx8pN-5B8aALsHh1Zc9mmDm8LVQSSpdpsw0v6vrJsv4r9\_lv0ic5aYbogc3i3h9mG6ZGMc7g9\_cGSRh\_soaKmXtpMEOxBFIsmiTa\_wticu9T07MbqQ42J9NwowHp8tn8OUIBlhjqA" ></div>            <div>                <h1>$<span id="litecoin"></span></h1>                <h1>Litecoin</h1>            </div>        </div>          <div class="coin-price">            <div class="logo"><img src="https://blogger.googleusercontent.com/img/a/AVvXsEgfcodOJm7ZIXw2kiqdo5abN4cUvFYgyqpKt91zHI8710ltPK5Ny\_S5X93w9LSDsF5jW61frn3C8a\_8w2GXu4bf0clzxuJljoQ8n6az5EI5zQOcl5W2LScP-1-41NQwPW5A3JWT9EwejtOnHsd3q2-llUsJJQ3Z74v\_2FOPn0TrI2529NS9\_hmFbvModw" ></div>            <div>                <h1>$<span id="ethereum"></span></h1>                <h1>Ethereum</h1>            </div>        </div>        <div class="coin-price">            <div class="logo"><img src="https://blogger.googleusercontent.com/img/a/AVvXsEj9YzGIFUSMpRoWE4IjGl\_o2zpdPkvtUS6jzIgZWEWl7ztYyV20oXu80A52v0R\_nXpt\_qXVBzxnfse2\_pfeIbVHwQSR3oLqqAyMqVqnzJpdbSCBHA2b\_zlheiLY3Bb0PYCEXQny7q-FnGE01ZtxVFVC8DbLWW-ZC1PC-gaqL7IC7ZfRxFOZufcv8lcY1g" ></div>            <div>                <h1>$<span id="dogecoin"></span></h1>                <h1>Dogecoin</h1>            </div>        </div>    </div>      <script src="js/country-list.js"></script>      <script src="js/script.js"></script>    </body>  </html> | /\* Import Google Font - Poppins \*/  @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600;700&display=swap');  \*{    margin: 0;    padding: 0;    box-sizing: border-box;    font-family: 'Poppins', sans-serif;  }  body{    display: flex;    align-items: center;    justify-content: center;    min-height: 100vh;    padding: 0 10px;    background: #fff;  }  ::selection{    color: #fff;    background: #675AFE;  }  .wrapper{    width: 370px;    padding: 30px;    border-radius: 7px;    background: #fff;    box-shadow: 7px 7px 20px rgba(0, 0, 0, 0.05);    margin-right: 1rem;  }  .wrapper header{    font-size: 28px;    font-weight: 500;    text-align: center;  }  .wrapper form{    margin: 40px 0 20px 0;  }  form :where(input, select, button){    width: 100%;    outline: none;    border-radius: 5px;    border: none;  }  form p{    font-size: 18px;    margin-bottom: 5px;  }  form input{    height: 50px;    font-size: 17px;    padding: 0 15px;    border: 1px solid #999;  }  form input:focus{    padding: 0 14px;    border: 2px solid #88888d;  }  form .drop-list{    display: flex;    margin-top: 20px;    align-items: center;    justify-content: space-between;  }  .drop-list .select-box{    display: flex;    width: 115px;    height: 45px;    align-items: center;    border-radius: 5px;    justify-content: center;    border: 1px solid #999;  }  .select-box img{    max-width: 21px;  }  .select-box select{    width: auto;    font-size: 16px;    background: none;    margin: 0 -5px 0 5px;  }  .select-box select::-webkit-scrollbar{    width: 8px;  }  .select-box select::-webkit-scrollbar-track{    background: #fff;  }  .select-box select::-webkit-scrollbar-thumb{    background: #888;    border-radius: 8px;    border-right: 2px solid #ffffff;  }  .drop-list .icon{    cursor: pointer;    margin-top: 30px;    font-size: 22px;  }  form .exchange-rate{    font-size: 17px;    margin: 20px 0 30px;  }  form button{    height: 52px;    color: #fff;    font-size: 17px;    cursor: pointer;    background: #151516;    transition: 0.3s ease;  }  form button:hover{    background: #5a5959;  }  .container{    display: flex;    justify-content: center;    align-items: center;    border-radius: 5px;    position: absolute;    top: 35rem;  }  .container .coin-price{    display: flex;    justify-content: center;    align-items: center;    padding: 10px 20px;    border-radius: 5px;    box-shadow: 0 0 3px #ccc;    margin: 7px;  }  .container .coin-price .logo{    width: 70px;    margin-right: 10px;  }  .container .coin-price .logo img{    width: 100%;  }  .container .coin-price div{    display: block;  }  .container .coin-price div h1{    font-size: 20px;  } |

|  |  |
| --- | --- |
| const dropList = document.querySelectorAll("form select"),      fromCurrency = document.querySelector(".from select"),      toCurrency = document.querySelector(".to select"),      getButton = document.querySelector("form button"),      exchangeRateTxt = document.querySelector("form .exchange-rate");  let exchangeRates = {};  for (let i = 0; i < dropList.length; i++) {      for(let currency\_code in country\_list){          let selected = i == 0 ? currency\_code == "USD" ? "selected" : "" : currency\_code == "NPR" ? "selected" : "";          let optionTag = `<option value="${currency\_code}" ${selected}>${currency\_code}</option>`;          dropList[i].insertAdjacentHTML("beforeend", optionTag);      }      dropList[i].addEventListener("change", e =>{          loadFlag(e.target);      });  }  function loadFlag(element){      for(let code in country\_list){          if(code == element.value){              let imgTag = element.parentElement.querySelector("img");              imgTag.src = `https://flagcdn.com/48x36/${country\_list[code].toLowerCase()}.png`;          }      }  }  window.addEventListener("load", ()=>{      getExchangeRate();  });  getButton.addEventListener("click", e =>{      e.preventDefault();      getExchangeRate();  });  const exchangeIcon = document.querySelector("form .icon");  exchangeIcon.addEventListener("click", ()=>{      let tempCode = fromCurrency.value;      fromCurrency.value = toCurrency.value;      toCurrency.value = tempCode;      loadFlag(fromCurrency);      loadFlag(toCurrency);      getExchangeRate();  });  const socket = new WebSocket('ws://localhost:3000');  socket.addEventListener('open', function (event) {      console.log('Connected to WebSocket server');  });  socket.addEventListener('message', function (event) {      const data = JSON.parse(event.data);      if (data.type === 'exchangeRate') {          const amount = document.querySelector("form input").value;          exchangeRateTxt.innerText = `${amount} ${fromCurrency.value} = ${data.rate} ${toCurrency.value}`;          displayConvertedAmounts(data.rate);      } else if (data.type === 'error') {          exchangeRateTxt.innerText = data.message;      } else if (data.type === 'ratesUpdate') {          exchangeRates = data.rates;          console.log('Exchange rates updated:', exchangeRates);      }  });  function getExchangeRate(){      const amount = document.querySelector("form input").value;      let amountVal = amount;      if(amountVal == "" || amountVal == "0"){          amountVal = 1;      }      const request = {          type: 'getExchangeRate',          fromCurrency: fromCurrency.value,          toCurrency: toCurrency.value,          amount: amountVal      };      if (socket.readyState === WebSocket.OPEN) {          socket.send(JSON.stringify(request));      } else {          console.error('WebSocket connection is not open');      }  }  let btc = document.getElementById("bitcoin");  let ltc = document.getElementById("litecoin");  let eth = document.getElementById("ethereum");  let doge = document.getElementById("dogecoin");  let liveprice = {      "async": true,      "scroosDomain": true,      "url": "https://api.coingecko.com/api/v3/simple/price?ids=bitcoin%2Clitecoin%2Cethereum%2Cdogecoin&vs\_currencies=usd",      "method": "GET",      "headers": {}  }  $.ajax(liveprice).done(function (response){      btc.innerHTML = response.bitcoin.usd;      ltc.innerHTML = response.litecoin.usd;      eth.innerHTML = response.ethereum.usd;      doge.innerHTML = response.dogecoin.usd;  });  const convertedBitcoin = document.getElementById("convertedBitcoin");  const convertedLitecoin = document.getElementById("convertedLitecoin");  const convertedEthereum = document.getElementById("convertedEthereum");  const convertedDogecoin = document.getElementById("convertedDogecoin");  function displayConvertedAmounts(amount) {      if (!amount || isNaN(amount)) return;      convertedBitcoin.textContent = (amount / btcPrice).toFixed(8); // Display in Bitcoin      convertedLitecoin.textContent = (amount / ltcPrice).toFixed(8); // Display in Litecoin      convertedEthereum.textContent = (amount / ethPrice).toFixed(8); // Display in Ethereum      convertedDogecoin.textContent = (amount / dogePrice).toFixed(8); // Display in Dogecoin  }  let btcPrice, ltcPrice, ethPrice, dogePrice;  $.ajax(liveprice).done(function (response) {      btcPrice = response.bitcoin.usd;      ltcPrice = response.litecoin.usd;      ethPrice = response.ethereum.usd;      dogePrice = response.dogecoin.usd;  }); | const express = require('express');  const http = require('http');  const WebSocket = require('ws');  const path = require('path');  const app = express();  const server = http.createServer(app);  const wss = new WebSocket.Server({ server });  app.use(express.static(path.join(\_\_dirname, 'public')));  let exchangeRates = {};  const API\_KEY = '811f483d4bff3f9925cdaa95';  const BASE\_URL = `https://v6.exchangerate-api.com/v6/${API\_KEY}/latest/USD`;  async function fetchExchangeRates() {      try {          const fetch = (await import('node-fetch')).default;          const response = await fetch(BASE\_URL);          const data = await response.json();          if (data.conversion\_rates) {              exchangeRates = data.conversion\_rates;              console.log('Exchange rates updated');              broadcastRates();          } else {              console.error('Failed to update exchange rates');          }      } catch (error) {          console.error('Error fetching exchange rates:', error);      }  }  function broadcastRates() {      const data = JSON.stringify({ type: 'ratesUpdate', rates: exchangeRates });      wss.clients.forEach(client => {          if (client.readyState === WebSocket.OPEN) {              client.send(data);          }      });  }  fetchExchangeRates();  setInterval(fetchExchangeRates, 3600000);  wss.on('connection', (ws) => {      console.log('Client connected');      ws.on('message', (message) => {          console.log(`Received message: ${message}`);          let parsedMessage = JSON.parse(message);          if (parsedMessage.type === 'getExchangeRate') {              const { fromCurrency, toCurrency, amount } = parsedMessage;              if (exchangeRates[fromCurrency] && exchangeRates[toCurrency]) {                  const rate = (amount \* (exchangeRates[toCurrency] / exchangeRates[fromCurrency])).toFixed(2);                  ws.send(JSON.stringify({ type: 'exchangeRate', rate }));              } else {                  ws.send(JSON.stringify({ type: 'error', message: 'Invalid currency code' }));              }          }      });      ws.on('close', () => {          console.log('Client disconnected');      });  });  const PORT = process.env.PORT || 3000;  server.listen(PORT, () => {      console.log(`Server is listening on port ${PORT}`);  }); |