DSC Protocol API Documentation

This API interacts with a smart contract system for a decentralized, overcollateralized stablecoin (DSC), backed by WBTC/WETH and pegged to $1. It provides endpoints to mint, burn, deposit, redeem, and liquidate collateral.

✅ Approval Flow

POST /approve-tokens

Purpose: Approve a contract to spend a user's ERC-20 tokens.

Payload:

json

```
{
  "token_address": "0xTokenAddress",
  "spender_address": "0xSpenderAddress",
  "amount": 1000
}
```

Returns:

- Transaction hash if approval was sent.
- Message if allowance was already sufficient.

□ Minting WBTC/WETH (for testing)

POST /mint-wbtc

POST /mint-weth

Purpose: Mint WBTC or WETH tokens (mock/testing only).

Payload:

json

```
{
  "amount": 1000
}
```

Returns:
Transaction hash of the mint.

💰 Deposit & Mint

POST /deposit-collateral

Purpose: Deposit collateral (WBTC or WETH) to back your DSC.

Payload:

json

```
{
  "token_address": "0xTokenAddress",
  "amount": 500
}
```

Returns:
Transaction hash of deposit.

POST /mint-dsc

Purpose: Mint DSC (stablecoin) against deposited collateral.

Payload:

json

```
{
  "amount": 100
}
```

Returns:
Transaction hash of mint.

POST /deposit-collateral-and-mint-dsc

Purpose: Deposit collateral and mint DSC in a single transaction.

Payload:

json

```
{
  "token_address": "0xTokenAddress",
  "amount": 500,
  "amount_dsc_to_mint": 100
}
```

Returns:
Transaction hash of combined action.

🔁 Burn & Redeem

POST /redeem-collateral

Purpose: Redeem previously deposited collateral.

Payload:

json

```
{
  "token_address": "0xTokenAddress",
  "amount": 100
}
```

Returns:
Transaction hash of redeem.

POST /burn-dsc

Purpose: Burn DSC (reduces debt).

Payload:

json

```
{
  "amount": 100
```

}

Returns:
Transaction hash of burn.

POST /redeem-collateral-for-dsc

Purpose: Burn DSC and redeem collateral in one go.

Payload:

json

```
{
  "token_address": "0xTokenAddress",
  "amount": 100,
  "amount_dsc_to_burn": 100
}
```

Returns:
Transaction hash.

⚠ Liquidation

POST /liquidate

Purpose: Liquidate an undercollateralized position.

Payload:

json

```
{
  "collateral": "0xTokenAddress",
  "user": "0xUserAddress",
  "debt_to_cover": 100
}
```

Returns:
Transaction hash if liquidation is successful.

📊 Account Info

POST /account-information

Purpose: Get a user's total DSC minted and collateral USD value.

Payload:

json

```
{
  "user": "0xUserAddress"
}
```

Returns:

json

```
{
  "total_dsc_minted": 100,
  "collateral_value_in_usd": 150
}
```

GET /collateral-balance

Purpose: Get user's collateral balance of a specific token.

Query Payload:

json

```
{
  "user": "0xUserAddress",
```

```
  "token": "0xTokenAddress"
}
```

Returns:

json

```
{
  "balance": 500
}
```

GET /health-factor

Purpose: Get user's health factor.

Query Payload:

json

```
{
  "user": "0xUserAddress"
}
```

Returns:

json

```
{
  "health_factor": "1000000000000000000"
}
```

Health factor below 1e18 = subject to liquidation.

GET /collateral-tokens

Purpose: Get list of allowed collateral token addresses.

Returns:

json

```
{
  "collateral_tokens": ["0xWBTC", "0xWETH"]
}
```

GET /usd-value

Purpose: Get USD value of a user's collateral of a specific token.

Query Payload:

json

```
{
  "user": "0xUserAddress",
  "token": "0xTokenAddress"
}
```

Returns:

json

```
{
  "usd_value": 123.45
}
```

⚙️ Notes for Frontend Integration

- All endpoints involving on-chain interactions return a **transaction hash**. Your UI should track this using web3/ethers to confirm.
- amount values are in **wei** (not ETH). You may want to convert using 1e18.

- Use /approve-tokens before calling deposit, mint, or any other function involving token transfers.
- Use /health-factor to show users if their position is healthy.