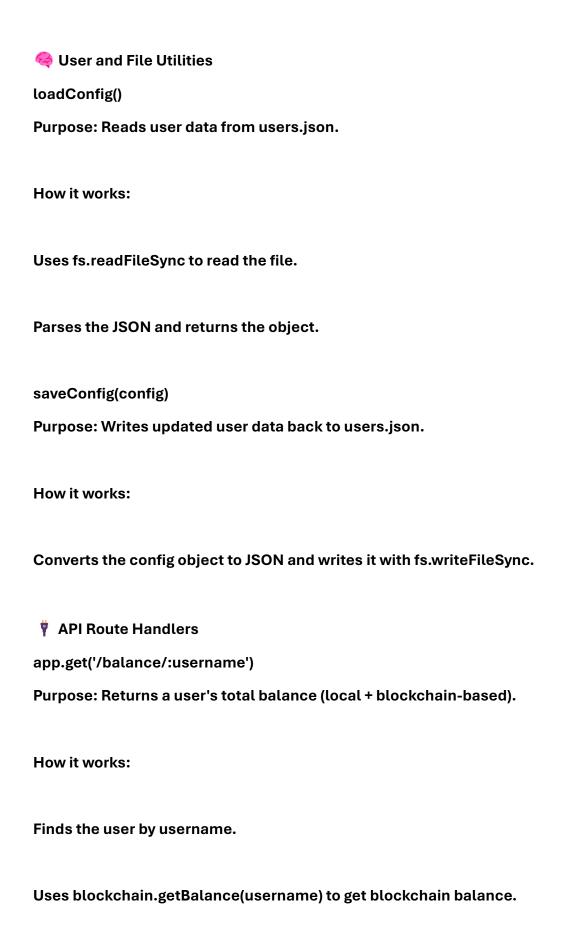
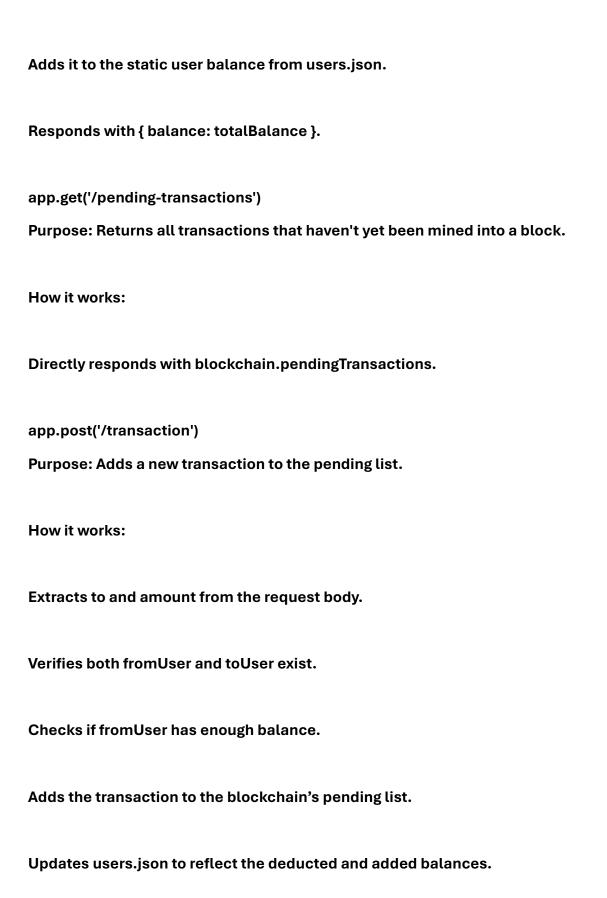
1 Authentication Functions
authenticateToken(req, res, next)
Purpose: Middleware that secures endpoints by verifying JWT tokens.
How it works:
Extracts the token from the Authorization header.
Uses jwt.verify to validate the token using the JWT_SECRET.
If valid, attaches the user info to req.user and proceeds.
If invalid or missing, responds with 401 or 403.
app.post('/auth/login',)
Purpose: Authenticates a user and returns a JWT if credentials are valid.
How it works:
HOW IT WOIRS.
Loads users from users.json.
Checks if the given username and apiKey exist and match.
If valid, generates a JWT token valid for 1 hour.
Responds with { token } or an error.

i Blockchain Functions
calculateHash()
Purpose: Generates a SHA-256 hash representing a block's unique identity.
How it works:
Combines the block's previous Hash, timestamp, transactions, and nonce.
Returns a SHA-256 hash using crypto.createHash.
mineBlock(difficulty)
Purpose: Performs proof-of-work for a block by finding a hash that meets the difficulty criteria.
How it works:
Repeatedly increments the nonce and recalculates the hash.
Stops once the hash starts with difficulty number of zeros.
createGenesisBlock()
Purpose: Creates the very first block of the blockchain.
How it works:
Calls new Block() with an empty previous hash '0' and an empty transaction list.

getLatestBlock()
Purpose: Retrieves the most recent block in the chain.
How it works:
Returns the last element in the chain array.
minePendingTransactions(miningRewardAddress)
Purpose: Creates and mines a block from all pending transactions.
How it works:
Constructs a new block from pendingTransactions.
Sets its previousHash to the latest block's hash.
Calls mineBlock to solve the proof-of-work.
Adds it to the chain.
Clears pendingTransactions and adds a reward transaction for the miner.
Returns the mined block.
addTransaction(transaction)

Purpose: Adds a new transaction to the list of pending ones.
How it works:
Appends a transaction object ({ from, to, amount }) to pendingTransactions.
getBalance(address)
Purpose: Computes the net balance for a user from all confirmed blocks.
How it works:
Iterates through every transaction in the chain:
Subtracts from the balance if the user sent coins.
Adds to the balance if the user received coins.





Responds with success and the new balance.
app.post('/mine')
Purpose: Mines a new block from pending transactions and rewards the miner.
How it works:
Gets the current user's username (as the miner).
Calls blockchain.minePendingTransactions(minerUsername).
Broadcasts the new block to all peers using WebSocket.
Returns success and reward info.
app.get('/blockchain')
Purpose: Returns the entire blockchain.
How it works:
Simply returns blockchain.chain.

☑ WebSocket Functions
wss.on('connection', socket => {})
Purpose: Handles a new peer connecting via WebSocket.
How it works:
Adds the socket to the sockets list.
Logs a connection message.
On receiving a NEW_BLOCK message:
Parses the message.
Appends the new block to the local blockchain.
Rebroadcasts it to all other peers.
broadcast(message)
Purpose: Sends a message to all connected peers.
How it works:
Loops through all sockets and sends the message.