I worked alone on this project.

This project implements a peer-to-peer blockchain network in Java, allowing multiple nodes to connect, mine blocks, and synchronize their blockchain copies. Each node can independently mine new blocks using a proof-of-work (PoW) system, broadcast mined blocks to peers, receive and validate blocks, and maintain blockchain integrity even when nodes disconnect and rejoin. Starting off with singlenode and building off our foundation on distributednode. The system consists of two main files: Block.java, which defines the structure of a blockchain block (including its hash, timestamp, previous hash, and nonce), and BCNode.java, which manages the blockchain, mining, and peer-to-peer communication using Java sockets. Each node listens for incoming connections while maintaining outgoing connections to other nodes. Mining involves adjusting a nonce until a block's hash meets the difficulty requirement, ensuring computational work before adding it to the chain. Once a block is mined, it is broadcasted, validated, and propagated through the network. To test the system, we started three connected nodes and mined blocks on different ones. Each block successfully propagated across the network, confirming that broadcasting and synchronization worked correctly. We also tested resilience by shutting down a node while the remaining nodes continued mining. When the node rejoined, it successfully synced the latest blockchain state from its peers. Additionally, we tested simultaneous mining on multiple nodes, and the system correctly handled conflicts by accepting the longest valid chain. One challenge we faced was ensuring all nodes consistently received broadcasted blocks. Some nodes initially failed to display received blocks despite successful transmissions. Debugging logs helped identify and fix the issue. Another challenge was handling user input errors, which we resolved by adding validation in main(). We also optimized mining performance to prevent inefficient loops. In the end, the system worked as expected, allowing decentralized mining, block validation, and fault tolerance. A potential improvement could be dynamic peer discovery instead of manually entering remote ports. Overall, this project demonstrated the core principles of blockchain and distributed systems while highlighting the challenges of decentralized data consistency.

The out put for all 3 was Received new block:
0000bf6e243d31014d4a50aac37c156cee2dbb3838c501947d1ff2cad31e04a4

And for the broadcaster it was Block mined:
0000bf6e243d31014d4a50aac37c156cee2dbb3838c501947d1ff2cad31e04a4

Block added successfully: 0000bf6e243d31014d4a50aac37c156cee2dbb3838c501947d1ff2cad31e04a4
Broadcasting block to peers...
Block sent to peer.

Also when one was killed the others preformed properly.