```
import hashlib
import json
import os
from time import time


class Block:
  def init(self, index, previous_hash, timestamp, data, hash):
      self.index = index
      self.previous_hash = previous_hash
      self.timestamp = timestamp
      self.data = data
      self.hash = hash


def calculate_hash(block):
  """Calculate the SHA-256 hash of a block."""
  block_string = json.dumps(block.dict, sort_keys=True)
  return hashlib.sha256(block_string.encode()).hexdigest()


class Blockchain:
  def init(self):
      self.chain = []
      self.create_genesis_block()


  def create_genesis_block(self):
      """Creates the first block in the blockchain."""
      genesis_block = Block(0, "0", time(), "Genesis Block", "")
      genesis_block.hash = calculate_hash(genesis_block)
      self.chain.append(genesis_block)
  def add_block(self, data):
      """Adds a new block to the blockchain."""
      previous_block = self.chain[-1]
      new_index = previous_block.index + 1
      new_timestamp = time()
      new_block = Block(new_index, previous_block.hash, new_timestamp, data, "")
      new_block.hash = calculate_hash(new_block)
      self.chain.append(new_block)


  def display_chain(self):
      """Displays the entire blockchain."""
      for block in self.chain:
          print(f"Index: {block.index}")
          print(f"Timestamp: {block.timestamp}")
          print(f"Data: {block.data}")
          print(f"Hash: {block.hash}")
          print(f"Previous Hash: {block.previous_hash}\n")


class BlockchainFileHandler:
  def init(self, filename='blockchain_data.txt'):
      self.filename = filename


  def save_blockchain(self, blockchain):
      """Saves the blockchain to a JSON file."""
      with open(self.filename, 'w') as file:
          chain_data = [block.dict for block in blockchain.chain]
          json.dump(chain_data, file, indent=4)
      print(f"Blockchain saved to {self.filename}.")
  def load_blockchain(self):
      """Loads the blockchain from a JSON file."""
      if not os.path.exists(self.filename):
          print("No existing blockchain found. Creating a new one.")
          return Blockchain()


      with open(self.filename, 'r') as file:
          chain_data = json.load(file)
          blockchain = Blockchain()
          blockchain.chain = [Block(**block) for block in chain_data]
      print(f"Blockchain loaded from {self.filename}.")
      return blockchain


# Example usage
if name == "main":
  # File handling object
  blockchain_handler = BlockchainFileHandler()


  # Load existing blockchain or create a new one
  health_blockchain = blockchain_handler.load_blockchain()


  # Adding new health records
  health_blockchain.add_block({"patient_name": "gowtham", "age": 18, "condition": "Heart problem", "treatment": "Get quality sleep and manage
stress"})
  health_blockchain.add_block({"patient_name": "santhosh", "age": 18, "condition": "Cold and fever", "treatment": "Rest and hydration"})
  health_blockchain.add_block({"patient_name": "sagar", "age": 18, "condition": "Covid-19", "treatment": "Isolation and vaccination"})
  health_blockchain.add_block({"patient_name": "sanjay", "age": 18, "condition": "Diabetes", "treatment": "Insulin and diet management"})
  health_blockchain.add_block({"patient_name": "prem", "age": 18, "condition": "Throat infection", "treatment": "Drinking warm water and
antibiotics"})
  health_blockchain.add_block({"patient_name": "karthik", "age": 18, "condition": "Nose infection", "treatment": "Drinking warm water and
antibiotics"})


  # Display the blockchain
  health_blockchain.display_chain()


  # Save the blockchain to a file
  blockchain_handler.save_blockchain(health_blockchain)
```