

main.py

```
import json
import datetime

# load account if exists
def load_account():
    try:
        with open("account.json", "r") as f:
            account = json.load(f)
            return account
    except FileNotFoundError:
        return -1

# create new account
def create_account():
    account = {
        "balance": 0.0,
        "transactions": []
    }
    with open("account.json", "w") as f:
        json.dump(account, f)
    return account

# save account state
def save_account(account):
    with open("account.json", "w") as f:
        json.dump(account, f)

# deposit function
def deposit(account, amount):
    account["balance"] += amount
    account["transactions"].append({
        "transaction_id": transaction_id_generator(len(account["transactions"])+1),
        "type": "deposit",
        "amount": amount,
        "date": str(datetime.datetime.now()),
```

```
        "balance_after": account["balance"]
    })
    save_account(account)

# Generate a unique 10-digit transaction ID
def transaction_id_generator(number):
    return str(number).zfill(10)

# withdraw function
def withdraw(account, amount):
    account["balance"] -= amount
    account["transactions"].append({
        "transaction_id": transaction_id_generator(len(account["transactions"])+1),
        "type": "withdraw",
        "amount": amount,
        "date": str(datetime.datetime.now()),
        "balance_after": account["balance"]
    })
    save_account(account)

# check balance function
def check_balance(account):
    return account["balance"]

# view transactions function
def view_transactions(account):
    if len(account["transactions"]) == 0:
        return -1
    else:
        return account["transactions"]

# search transaction by ID function
def search_transaction_by_id(account, transaction_id):
    for i in account["transactions"]:
        if i["transaction_id"] == transaction_id:
            return i
    return -1

# search transaction by amount function
def search_transaction_by_amount(account, amount):
```

```

results = []
for i in account["transactions"]:
    if i["amount"] == amount:
        results.append(i)
if len(results) == 0:
    return -1
else:
    return results

# search transaction by date function
def search_transaction_by_date(account, date):
    results = []
    for i in account["transactions"]:
        if date in i["date"]:
            results.append(i)
    if len(results) == 0:
        return -1
    else:
        return results

# main program
def main():
    # load or create account
    account = load_account()
    if account == -1:
        account = create_account()

    program_state = True
    while program_state:
        # main menu input
        while True:
            try:
                print("\nInput option:\n1. Deposit\n2. Withdraw\n3. Check Balance\n4. View
Transactions\n5. Search Transaction\n6. Exit")
                choice = int(input("Enter choice (1-6): "))
                if choice >= 1 and choice <= 6:
                    break
            else:
                print("Invalid choice. Please enter a number between 1 and 6.")
        except ValueError:

```

```
print("Invalid input. Please enter a number between 1 and 6.")

# carry out the chosen operations with:
# 1. input with exception handling
# 2. perform operation
# 3. display result with formatting

# options:
# 1. deposit
if choice == 1:
    while True:
        try:
            amount = float(input("Enter amount to deposit: "))
            if amount > 0:
                deposit(account, amount)
                print(f"Deposited: ${amount:.2f}")
                break
            else:
                print("Amount must be positive.")
        except ValueError:
            print("Invalid input. Please enter a valid amount.")

# 2. withdraw
elif choice == 2:
    if account["balance"] <= 0:
        print("Insufficient balance to withdraw.")
    else:
        while True:
            try:
                amount = float(input("Enter amount to withdraw: "))
                if amount > 0 and amount <= account["balance"]:
                    withdraw(account, amount)
                    print(f"Withdrew: ${amount:.2f}")
                    break
            else:
                print("Invalid amount. Please enter a positive amount not
exceeding your balance.")
        except ValueError:
            print("Invalid input. Please enter a valid amount.")
```

```

# 3. check balance
elif choice == 3:
    balance = check_balance(account)
    print(f"Current Balance: ${balance:.2f}")

# 4. view transactions
elif choice == 4:
    transactions = view_transactions(account)
    if transactions == -1:
        print("Record is empty.")
    else:
        print("Transaction History:")
        for i in transactions:
            print(f"ID: {i['transaction_id']} | Type: {i['type']} | Amount:
${i['amount']:.2f} | Date: {i['date']} | Balance After: ${i['balance_after']:.2f}")

elif choice == 5:
    # search transaction submenu
    while True:
        try:
            print("Search options:\n1. By Transaction ID\n2. By Amount\n3. By Date\n4.
Back to Main Menu")
            search_choice = int(input("Enter choice (1-4): "))
            if search_choice >= 1 and search_choice <= 4:
                break
            else:
                print("Invalid choice. Please enter a number between 1 and 4.")
        except ValueError:
            print("Invalid input. Please enter a number between 1 and 4.")

# 1. search by transaction ID
if search_choice == 1:
    # validate id input
    while True:
        try:
            search_id = input("Enter Transaction ID to search: ")
            if len(search_id) == 10 and search_id.isdigit():
                break
            else:
                print("Invalid Transaction ID. It must be a 10-digit number.")

```

```

        except ValueError:
            print("Invalid Transaction ID. It must be a 10-digit number.")

# perform search
result = search_transaction_by_id(account, search_id)
if result == -1:
    print("Transaction not found.")
else:
    print("Transaction Found:")
    print(f"ID: {result['transaction_id']} | Type: {result['type']} | Amount:
    ${result['amount']:.2f} | Date: {result['date']} | Balance After:
    ${result['balance_after']:.2f}")

# 2. search by amount
elif search_choice == 2:
    # validate amount input
    while True:
        try:
            search_amount = float(input("Enter Amount to search: "))
            if search_amount > 0:
                break
            else:
                print("Amount must be positive.")
        except ValueError:
            print("Invalid input. Please enter a valid amount.")

# perform search
results = search_transaction_by_amount(account, search_amount)
if results == -1:
    print("No transactions found with specified amount.")
else:
    print("Transactions Found:")
    for i in results:
        print(f"ID: {i['transaction_id']} | Type: {i['type']} | Amount:
        ${i['amount']:.2f} | Date: {i['date']} | Balance After: ${i['balance_after']:.2f}")

# 3. search by date
elif search_choice == 3:
    # validate date input
    while True:

```

```

    try:
        search_year = input("Enter Year (YYYY) to search: ")
        search_month = input("Enter Month (MM) to search: ")
        search_day = input("Enter Day (DD) to search: ")
        if (len(search_year) == 4 and search_year.isdigit() and
            len(search_month) == 2 and search_month.isdigit() and
            len(search_day) == 2 and search_day.isdigit()):
            search_date = f"{search_year}-{search_month}-{search_day}"
            break
        else:
            print("Invalid date format. Please enter valid year, month, and
day.")

    except ValueError:
        print("Invalid date format. Please enter valid year, month, and day.")

# perform search
results = search_transaction_by_date(account, search_date)
if results == -1:
    print("No transactions found on the specified date.")
else:
    print("Transactions Found:")
    for i in results:
        print(f"ID: {i['transaction_id']} | Type: {i['type']} | Amount:
${i['amount']:.2f} | Date: {i['date']} | Balance After: ${i['balance_after']:.2f}")

# 4. back to main menu
elif search_choice == 4:
    main()

# 6. exit
elif choice == 6:
    program_state = False

# run main program
main()
exit()

```

Revision #1

Created 2026-03-18 23:45:32 UTC by Samuel Lee

Updated 2026-03-18 23:46:12 UTC by Samuel Lee