ATM design problem

This project should take approximately four hours to complete.

Write a program that allows a user to deposit or get money from an ATM. Once the program is started, it should accept input until the END command is sent.

You should:

1. Store the amount of money that is in the machine. The machine initially contains $10,000.
2. Validate any request via an account ID and PIN.
3. Store individual account data, including transaction history. Attached is a CSV file containing the initial account data. Assume there is no preexisting transaction history.
4. Handle any bogus requests with relevant error handling and logging.
5. Receive and dispense money.
6. Write unit tests.

Specification

AUTHORIZE

Authorizes an account locally until they are logged out. If there is no activity for 2 minutes, your program should automatically log out the account.

`authorize <account_id> <pin>`

A successful authorization returns:

```
<account_id> successfully authorized.
```

An unsuccessful authorization returns:

```
Authorization failed.
```

Attempts to access any other commands, with the exception of `logout` and `end`, without an active authorization should result in:

```
Authorization required.
```

WITHDRAW

Removes value from the authorized account. The machine only contains $20 bills, so the withdrawal amount must be a multiple of 20.

`withdraw <value>`

If account has not been overdrawn, returns balance after withdrawal in the format:

```
Amount dispensed: $<x>
Current balance: <balance>
```

If the account has been overdrawn with this transaction, removes a further $5 from their account, and returns:

```
Amount dispensed: $<x>
You have been charged an overdraft fee of $5. Current balance: <balance>
```

The machine can't dispense more money than it contains. If in the above two scenarios the machine contains less money than was requested, the withdrawal amount should be adjusted to be the amount in the machine and this should be prepended to the return value:

```
Unable to dispense full amount requested at this time.
```

If instead there is no money in the machine, the return value should be this and only this:

```
Unable to process your withdrawal at this time.
```

If the account is already overdrawn, do not perform any checks against the available money in the machine, do not process the withdrawal, and return only this:

```
Your account is overdrawn! You may not make withdrawals at this time.
```

DEPOSIT

Adds value to the authorized account. The deposited amount does not need to be a multiple of 20.

`deposit <value>`

Returns the account's balance after deposit is made in the format:

```
Current balance: <balance>
```

BALANCE

Returns the account's current balance.

`balance`

Returns the account's balance in the format:

```
Current balance: <balance>
```

HISTORY

Returns the account's transaction history.

`history`

If there is no history, returns:

```
No history found
```

Otherwise, returns the transaction history in reverse chronological order (most recent transaction first) in the format:

```
<date> <time> <amount> <balance after transaction>
```

For example:

```
2020-02-04 13:04:22 -20.00 140.67
2020-02-04 13:04:01 60.44 160.67
2020-02-04 13:03:49 35.00 100.23
```

LOG OUT

Deactivates the currently authorized account.

`logout`

If an account is currently authorized, returns:

```
Account <account_id> logged out.
```

Otherwise, returns:

```
No account is currently authorized.
```

END

Shuts down the server.

`end`

Returns nothing, and ends the program.