```
def test_sign_in_with_correct_credentials_and_soft_token_mfa(monkeypatch, mocker):
    class DummyInput:
        def clear(self): pass
        def send_keys(self, x): self.sent = x
        def click(self): self.clicked = True
    driver = mocker.Mock()
    driver.current_url = 'https://mint.intuit.com/'
    driver.implicitly_wait.return_value = None
    driver.get.return_value = None
    # Simulate needed methods so sign_in doesn't fail
    dummy_input = DummyInput()
    driver.find_element.return_value = dummy_input
    driver.find_elements.return_value = [dummy_input]
    driver.execute_script.return_value = None
    driver.find_element_by_id = lambda x: dummy_input
    driver.current_url = 'https://mint.intuit.com/'
    monkeypatch.setattr('mintapi.signIn.handle_same_page_username_password', lambda *a,**kw: None)
    monkeypatch.setattr('mintapi.signIn.handle_login_failures', lambda *a,**kw: None)
    monkeypatch.setattr('mintapi.signIn.bypass_verified_user_page', lambda *a,**kw: True)
    monkeypatch.setattr('mintapi.signIn.account_selection_page', lambda *a,**kw: None)
    monkeypatch.setattr('mintapi.signIn.password_page', lambda *a,**kw: None)
    monkeypatch.setattr('mintapi.signIn.handle_wait_for_sync', lambda *a,**kw: 'Account refresh complete')
    monkeypatch.setattr('oathtool.generate_otp', lambda token: '123456')
    # Simulate MFA
    email = 'user@example.com'
    password = 'correct-password'
    result = sign_in(email=email, password=password, driver=driver, mfa_method='SOFT_TOKEN', mfa_token='dummy-secret', wa
    assert result is None or result == 'Account refresh complete'
```

---Extra Details---
Filename: signIn.py
Description: Tests the successful sign-in flow with correct email/password and SOFT_TOKEN MFA using oathtool. Checks that si
Score: 100%
Alignment: 100%
Validation Notes: Simulates correct credentials with soft token MFA; monkeypatching is used to skip Selenium and external calls.