Password Test Case SDET - Automated Test Code (Generated by Haiku)

Python Test Code (auto-generated from Test Lead requirements):

Here is the generated test code document for password validation:  
  
# Test Cases for Password Validation  
  
## Overview  
  
This document covers test cases to validate the password policy which requires passwords to:  
  
- Be at least 8 characters long  
- Include at least one numeric digit   
- Include at least one special character  
  
## Requirements  
  
- Passwords must be a minimum of 8 characters  
- Passwords must contain at least one number   
- Passwords must contain at least one special character  
  
## Test Cases  
  
```python  
import unittest  
  
class TestPasswordValidation(unittest.TestCase):  
  
 def test\_password\_too\_short(self):  
 password = "abc1$"   
 self.assertFalse(is\_valid\_password(password))  
  
 def test\_password\_no\_number(self):  
 password = "abcd@xyz"  
 self.assertFalse(is\_valid\_password(password))  
  
 def test\_password\_no\_special\_char(self):  
 password = "abcd1234"  
 self.assertFalse(is\_valid\_password(password))  
  
 def test\_password\_valid(self):  
 password = "abc1@def"  
 self.assertTrue(is\_valid\_password(password))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()  
```  
  
## Additional Test Cases  
  
```python   
def test\_long\_valid\_password(self):  
 password = "MyPass123!"  
 self.assertTrue(is\_valid\_password(password))  
  
def test\_password\_too\_short(self):  
 password = "abc1$"  
 self.assertFalse(is\_valid\_password(password))  
  
def test\_password\_no\_number\_or\_special(self):  
 password = "abcdefgh"   
 self.assertFalse(is\_valid\_password(password))  
  
def test\_password\_missing\_special\_char(self):  
 password = "abcd1234"  
 self.assertFalse(is\_valid\_password(password))  
  
def test\_complex\_valid\_password(self):  
 password = "Ab1$xyz9"   
 self.assertTrue(is\_valid\_password(password))  
```