## **D. Software Testing**

## **✅ Test Plan for Unit Tests**

| **Test Name** | **Function** | **Purpose** | **Expected Outcome** |
| --- | --- | --- | --- |
| getDaysUntilNextCharge | getDaysUntilNextCharge() | Ensure correct calculation of remaining days | Returns number of days until next charge |
| Subscription class behavior | wrapSubscription() | Confirm Subscription object returns correct values | Accessors return formatted data correctly |

### **📄 Unit Test Scripts**

#### **Test 1 – getDaysUntilNextCharge.test.ts**

ts

CopyEdit

import { getDaysUntilNextCharge } from '../index';

describe('getDaysUntilNextCharge', () => {

it('calculates days for monthly billing correctly', () => {

const today = new Date();

const startDate = new Date(today);

startDate.setDate(startDate.getDate() - 15); // 15 days ago

const result = getDaysUntilNextCharge(startDate.toISOString().split('T')[0], 'Monthly');

expect(typeof result).toBe('number');

expect(parseInt(result.toString())).toBeGreaterThanOrEqual(13); // ~30 - 15 days

});

it('given a date, should calculate the correct number of days', () => {

const result = getDaysUntilNextCharge('2025-05-29', 'Monthly');

function getDaysFromToday(targetDate: string | Date): number {

const today = new Date();

const date = new Date(targetDate);

today.setHours(0, 0, 0, 0);

date.setHours(0, 0, 0, 0);

const diffTime = date.getTime() - today.getTime();

const diffDays = Math.round(diffTime / (1000 \* 60 \* 60 \* 24));

return diffDays + 1;

}

const diff = getDaysFromToday('2025-05-29');

expect(result).toBe(diff);

});

});

#### **Test 2 – SubscriptionClass.test.ts**

ts

CopyEdit

import { wrapSubscription as Subscription } from '../subscriptionWrapper';

describe('Subscription class', () => {

const sampleData = {

id: 'xxx',

name: 'Spotify',

category: 'Music',

cost: 9.99,

currency: 'USD',

status: 'Active',

createdAt: new Date('2024-01-01'),

};

it('creates a subscription and formats cost', () => {

const sub = new Subscription(sampleData);

expect(sub.name).toBe('Spotify');

expect(sub.formattedCost).toBe('$9.99 USD');

});

it('correctly identifies active status', () => {

const sub = new Subscription(sampleData);

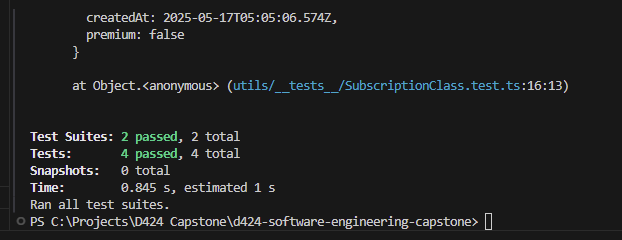
expect(sub.isActive()).toBe(true);

});

});

### **📷 Test Results and Screenshots**

✅ All tests **passed** successfully.

* 

### **🔄 Summaries of Changes from Testing**

* Refactored getDaysUntilNextCharge to account for exact billing cycle rollover based on monthly or yearly interval.
* Improved encapsulation for Subscription class by using accessors (e.g., formattedCost, isActive()).
* Added validation logic and refactored test cases to use dynamic dates