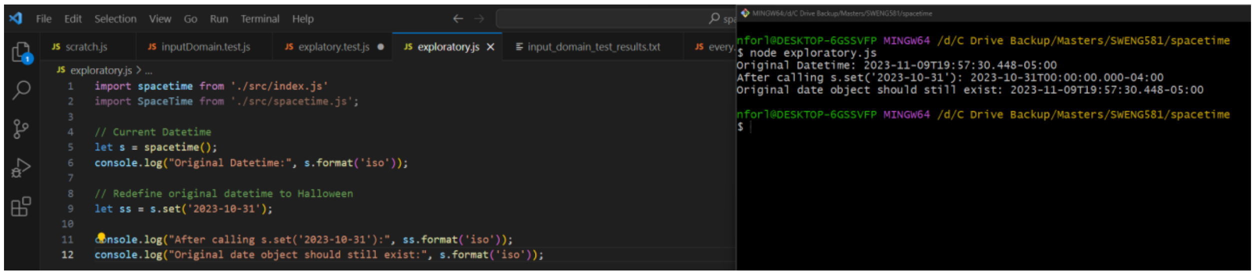
Test Tour	Guidebook Tour
Object to be tested / Rough Guidance	In accordance with the definition of Guidebook Tour concept, the goal of this session is to verify that the functions execute as defined in the API docs
Tester	Whole Team
Further Testing Opportunities?	Could also be tested using invalid inputs

Protocol

1. Test: Verify that the .set() function changes the date to the input parameter and a clone is returned
    a. Status: ✓
    b. Documentation/Comments: Passed

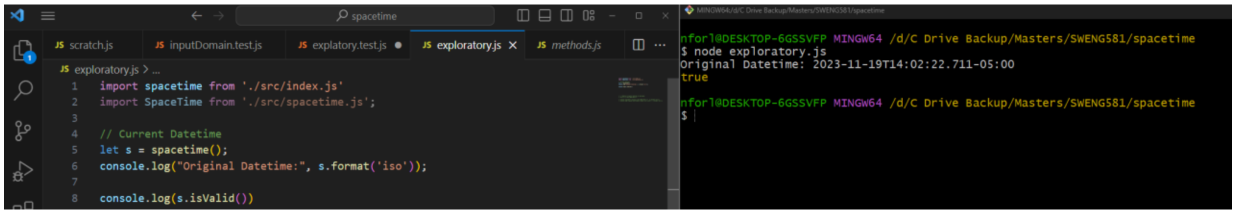


File Edit Selection View Go Run Terminal Help
scratch.js
inputDomain.test.js
explatory.test.js
exploratory.js
input_domain_test_results.txt
every
exploratory.js > ...
MINGW64:/d/C Drive Backup/Masters/SWENG581/spacetime

```
import spacetime from './src/index.js'
import SpaceTime from './src/spacetime.js';

// Current Datetime
let s = spacetime();
console.log("Original Datetime:", s.format('iso'));

// Redefine original datetime to Halloween
let ss = s.set('2023-10-31');

console.log("After calling s.set('2023-10-31'):", ss.format('iso'));
console.log("Original date object should still exist:", s.format('iso'));
```

```
nforl@DESKTOP-6GSSVFP MINGW64 /d/C Drive Backup/Masters/SWENG581/spacetime
$ node exploratory.js
Original Datetime: 2023-11-09T19:57:30.448-05:00
After calling s.set('2023-10-31'): 2023-10-31T00:00:00.000-04:00
Original date object should still exist: 2023-11-09T19:57:30.448-05:00

nforl@DESKTOP-6GSSVFP MINGW64 /d/C Drive Backup/Masters/SWENG581/spacetime
$
```

2. Test: Verify that the .isValid() function confirms a valid date and time according to the rules of the Gregorian calendar and the JavaScript Date object – Return clone
    a. Status: ✓
    b. Documentation/Comments: Passed – However, the docs say the return type is a clone. The correct return type should be Boolean

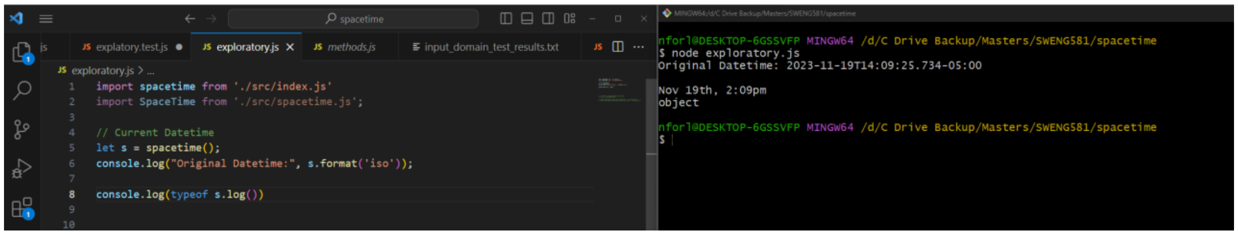


```
import spacetime from './src/index.js'
import SpaceTime from './src/spacetime.js';

// Current Datetime
let s = spacetime();
console.log("Original Datetime:", s.format('iso'));

console.log(s.isValid())
```

```
nforl@DESKTOP-6GSSVFP MINGW64 /d/C Drive Backup/Masters/SWENG581/spacetime
$ node exploratory.js
Original Datetime: 2023-11-19T14:02:22.711-05:00
true

nforl@DESKTOP-6GSSVFP MINGW64 /d/C Drive Backup/Masters/SWENG581/spacetime
$
```

3. Test: Verify that the .log() function will pretty-print the date to the console, for nicer debugging – Return clone
    a. Status: ✓

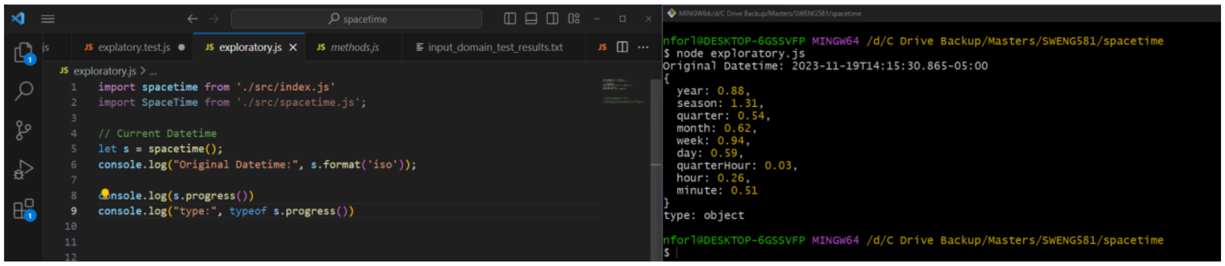
b. Documentation/Comments: Passed



```
import spacetime from './src/index.js'
import SpaceTime from './src/spacetime.js';

// Current Datetime
let s = spacetime();
console.log("Original Datetime:", s.format('iso'));

console.log(typeof s.log())
```

```
nforl@DESKTOP-6GSSVFP MINGW64 /d/C Drive Backup/Masters/SWENG581/spacetime
$ node exploratory.js
Original Datetime: 2023-11-19T14:09:25.734-05:00

Nov 19th, 2:09pm
object

nforl@DESKTOP-6GSSVFP MINGW64 /d/C Drive Backup/Masters/SWENG581/spacetime
$
```

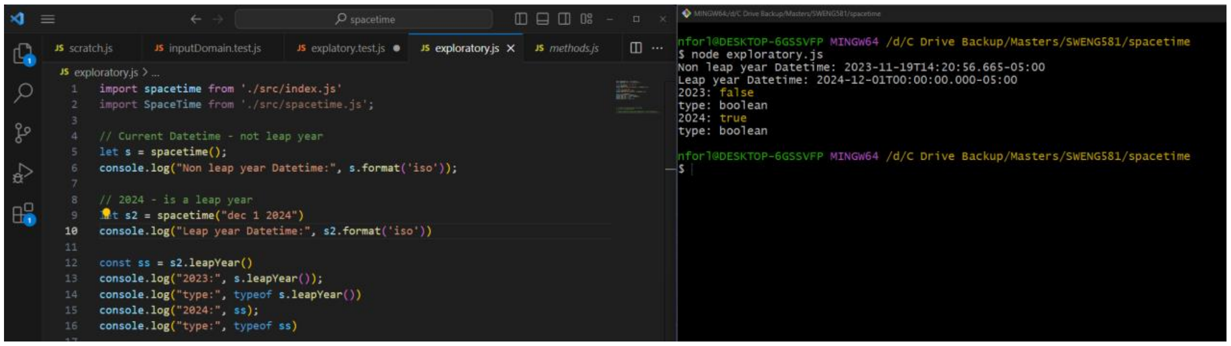
4. Test: Verify that the .progress() function correctly calculates the progress of a given moment within a specific time frame, such as a day, week, month, or year. The function takes a moment in time and returns a value between 0 and 1, indicating how far that moment is in relation to the start and end of the specified time frame – Return Object
   a. Status: ✓
   b. Documentation/Comments: Passed



```
import spacetime from './src/index.js'
import SpaceTime from './src/spacetime.js';

// Current Datetime
let s = spacetime();
console.log("Original Datetime:", s.format('iso'));

console.log(s.progress())
console.log("type:", typeof s.progress())
```

```
nforl@DESKTOP-6GSSVFP MINGW64 /d/C Drive Backup/Masters/SWENG581/spacetime
$ node exploratory.js
Original Datetime: 2023-11-19T14:15:30.865-05:00
{
  year: 0.88,
  season: 1.31,
  quarter: 0.54,
  month: 0.62,
  week: 0.94,
  day: 0.59,
  quarterHour: 0.03,
  hour: 0.26,
  minute: 0.51
}
type: object

nforl@DESKTOP-6GSSVFP MINGW64 /d/C Drive Backup/Masters/SWENG581/spacetime
$
```

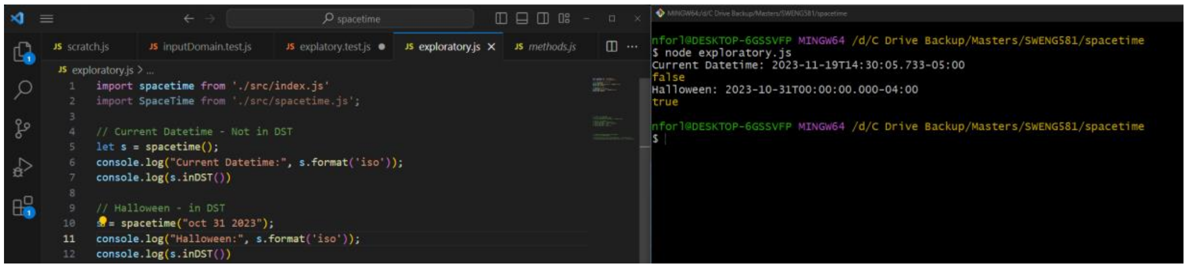
5. Test: Verify that the .leapYear() function correctly returns if the current year is a leap year or not – Returns Boolean
   a. Status: ✓
   b. Documentation/Comments:



```
import spacetime from './src/index.js'
import SpaceTime from './src/spacetime.js';

// Current Datetime - not leap year
let s = spacetime();
console.log("Non leap year Datetime:", s.format('iso'));

// 2024 - is a leap year
let s2 = spacetime("dec 1 2024")
console.log("Leap year Datetime:", s2.format('iso'))

const ss = s2.leapYear()
console.log("2023:", s.leapYear());
console.log("type:", typeof s.leapYear())
console.log("2024:", ss);
console.log("type:", typeof ss)
```

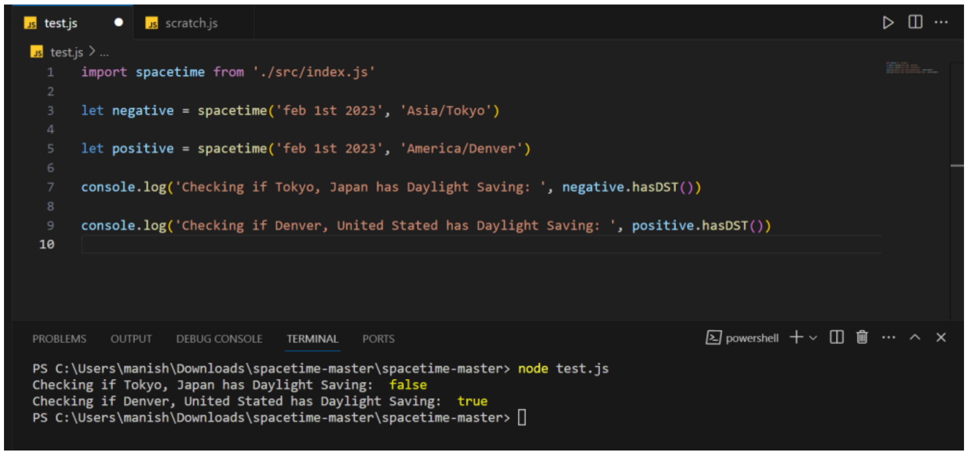
```
nforl@DESKTOP-6GSSVFP MINGW64 /d/C Drive Backup/Masters/SWENG581/spacetime
$ node exploratory.js
Non leap year Datetime: 2023-11-19T14:20:56.665-05:00
Leap year Datetime: 2024-12-01T00:00:00.000-05:00
2023: false
type: boolean
2024: true
type: boolean

nforl@DESKTOP-6GSSVFP MINGW64 /d/C Drive Backup/Masters/SWENG581/spacetime
$
```

6. Test: Verify that the .inDST() function correctly returns if the current selected time zone is in daylight savings time – Returns Boolean
   a. Status: ✓
   b. Documentation/Comments:



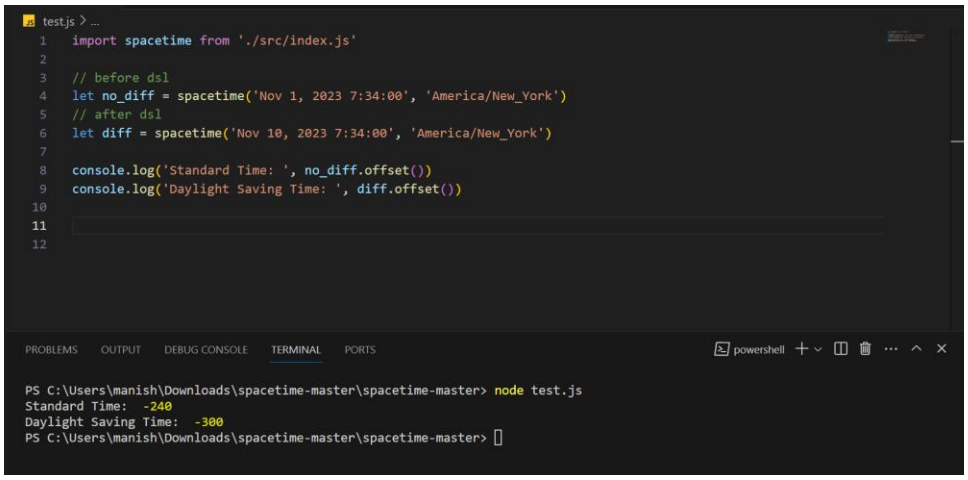
```
import spacetime from './src/index.js'
import SpaceTime from './src/spacetime.js';

// Current Datetime - Not in DST
let s = spacetime();
console.log("Current Datetime:", s.format('iso'));
console.log(s.inDST())

// Halloween - in DST
s = spacetime("oct 31 2023");
console.log("Halloween:", s.format('iso'));
console.log(s.inDST())
```

```
nforl@DESKTOP-6GSSVFP MINGW64 /d/C Drive Backup/Masters/SWENG581/spacetime
$ node exploratory.js
Current Datetime: 2023-11-19T14:30:05.733-05:00
false
Halloween: 2023-10-31T00:00:00.000-04:00
true

nforl@DESKTOP-6GSSVFP MINGW64 /d/C Drive Backup/Masters/SWENG581/spacetime
$
```

7. Test: Verify that the .hasDST() function validated if the timezone ever uses daylight-savings and return boolean
   a. Status: ✓
   b. Documentation/Comments: Passed
   c. 

```
import spacetime from './src/index.js'

let negative = spacetime('feb 1st 2023', 'Asia/Tokyo')

let positive = spacetime('feb 1st 2023', 'America/Denver')

console.log('Checking if Tokyo, Japan has Daylight Saving: ', negative.hasDST())

console.log('Checking if Denver, United Stated has Daylight Saving: ', positive.hasDST())
```

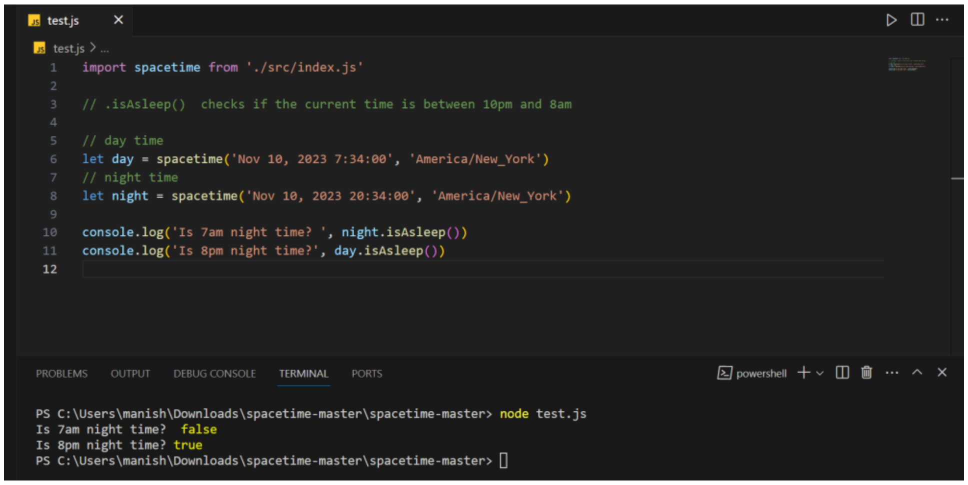
```
PS C:\Users\manish\Downloads\spacetime-master\spacetime-master> node test.js
Checking if Tokyo, Japan has Daylight Saving:  false
Checking if Denver, United Stated has Daylight Saving:  true
PS C:\Users\manish\Downloads\spacetime-master\spacetime-master>
```

8. Test: Verify that the .offset() function correctly calculates the current time difference from Coordinated Universal Time (UTC) while accounting for Daylight Saving Time (DST) - Returns number.
   a. Status: ✓
   b. Documentation/Comments: Passed
   c. 

```
import spacetime from './src/index.js'

// before dsl
let no_diff = spacetime('Nov 1, 2023 7:34:00', 'America/New_York')
// after dsl
let diff = spacetime('Nov 10, 2023 7:34:00', 'America/New_York')

console.log('Standard Time: ', no_diff.offset())
console.log('Daylight Saving Time: ', diff.offset())
```

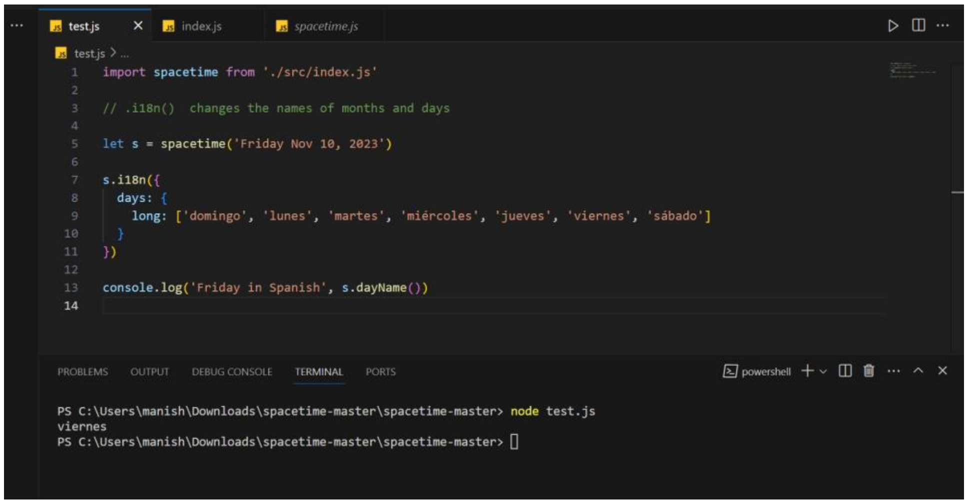
```
PS C:\Users\manish\Downloads\spacetime-master\spacetime-master> node test.js
Standard Time:  -240
Daylight Saving Time:  -300
PS C:\Users\manish\Downloads\spacetime-master\spacetime-master>
```

9. Test: Verify that .isAsleep() correctly states if a user is asleep, or in this case, the time is between 10pm and 8am – Returns boolean.
   a. Status: ✓
   b. Documentation/Comments: Passed

c. 

```
test.js ×
test.js > ...
import spacetime from './src/index.js'

// .isAsleep()  checks if the current time is between 10pm and 8am

// day time
let day = spacetime('Nov 10, 2023 7:34:00', 'America/New_York')
// night time
let night = spacetime('Nov 10, 2023 20:34:00', 'America/New_York')

console.log('Is 7am night time? ', night.isAsleep())
console.log('Is 8pm night time?', day.isAsleep())
```

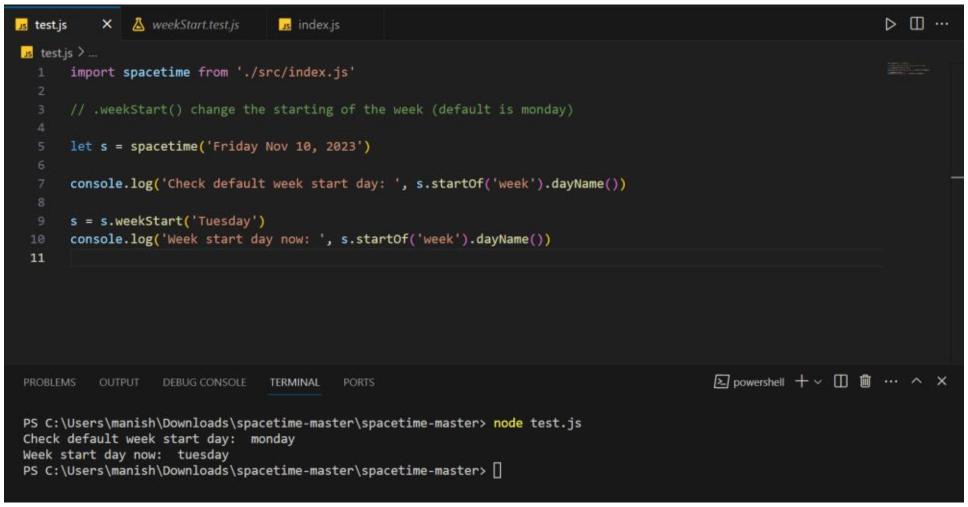
```
PS C:\Users\manish\Downloads\spacetime-master\spacetime-master> node test.js
Is 7am night time?  false
Is 8pm night time? true
PS C:\Users\manish\Downloads\spacetime-master\spacetime-master>
```

10. Test: Verify that the .i18n() internalization function correctly allows a user to switch between different sets of names based on the desired language or regional conventions. Example: In English, ‘November’ and ‘December’ are ‘Novembre’ and ‘Vendredi’ in French. - Returns clone
    a. Status: ✓
    b. Documentation/Comments: Passed
    c. 

```
test.js ×   index.js   spacetime.js
test.js > ...
import spacetime from './src/index.js'

// .i18n()  changes the names of months and days

let s = spacetime('Friday Nov 10, 2023')

s.i18n({
  days: {
    long: ['domingo', 'lunes', 'martes', 'miércoles', 'jueves', 'viernes', 'sábado']
  }
})

console.log('Friday in Spanish', s.dayName())
```

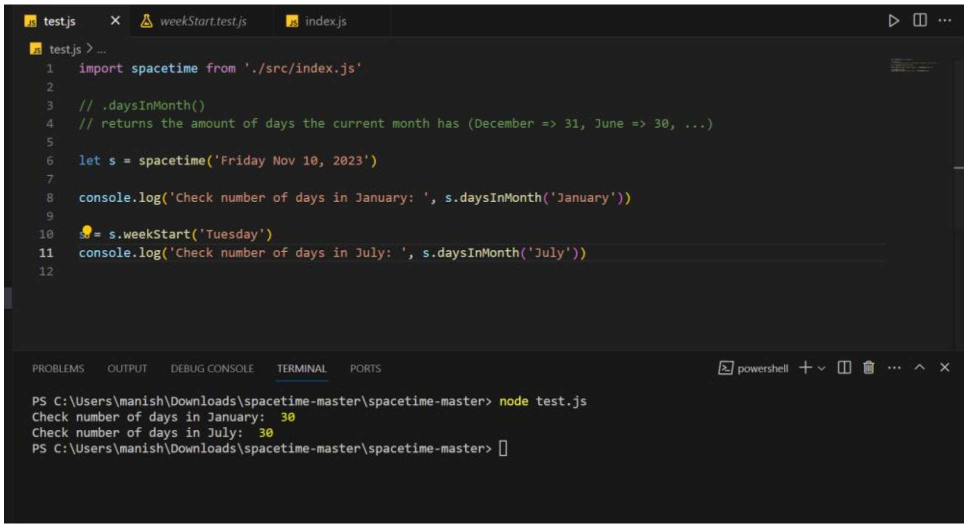
```
PS C:\Users\manish\Downloads\spacetime-master\spacetime-master> node test.js
viernes
PS C:\Users\manish\Downloads\spacetime-master\spacetime-master>
```

11. Test: Verify that the .weekStart() function can be changed to whichever day is desired. The default starting day of the week for this application is Monday – Return clone
    a. Status: ✓
    b. Documentation/Comments: Passed

c. 

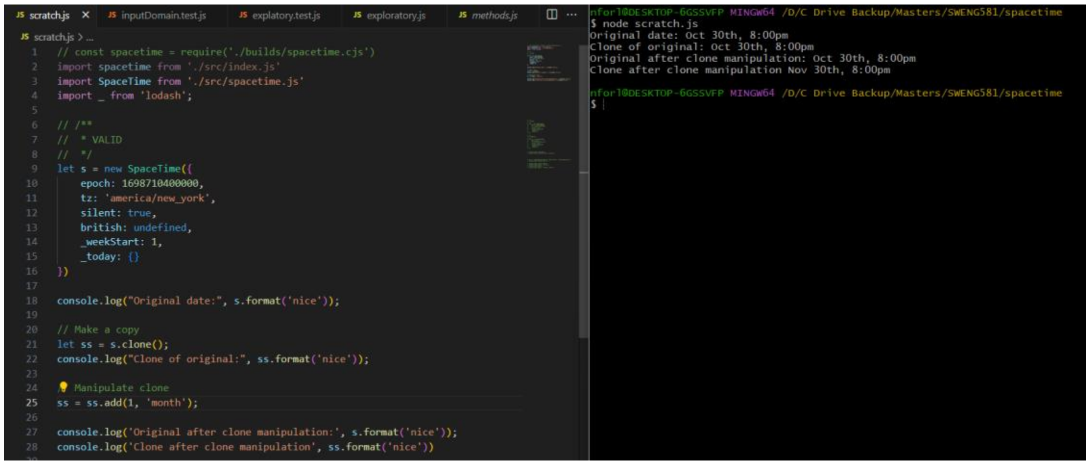
```
import spacetime from './src/index.js'

// .weekStart() change the starting of the week (default is monday)

let s = spacetime('Friday Nov 10, 2023')

console.log('Check default week start day: ', s.startOf('week').dayName())

s = s.weekStart('Tuesday')
console.log('Week start day now: ', s.startOf('week').dayName())
```

```
PS C:\Users\manish\Downloads\spacetime-master\spacetime-master> node test.js
Check default week start day:  monday
Week start day now:  tuesday
PS C:\Users\manish\Downloads\spacetime-master\spacetime-master> 
```

12. Test: Verify that the .daysInMonth() function correctly returns the number of days witin the current month (December has 31 days; June has 30, etc.) - Returns number
    a. Status: ✓
    b. Documentation/Comments: Passed
    c. 

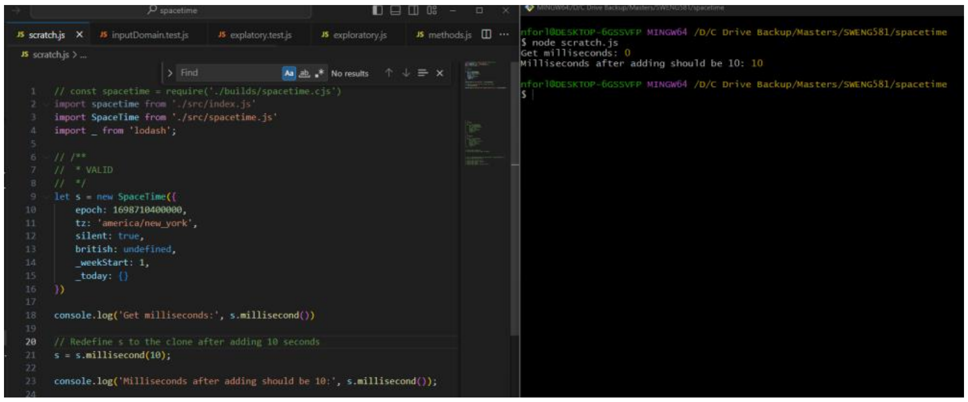
```
import spacetime from './src/index.js'

// .daysInMonth()
// returns the amount of days the current month has (December => 31, June => 30, ...)

let s = spacetime('Friday Nov 10, 2023')

console.log('Check number of days in January: ', s.daysInMonth('January'))

s = s.weekStart('Tuesday')
console.log('Check number of days in July: ', s.daysInMonth('July'))
```

```
PS C:\Users\manish\Downloads\spacetime-master\spacetime-master> node test.js
Check number of days in January:  30
Check number of days in July:  30
PS C:\Users\manish\Downloads\spacetime-master\spacetime-master> 
```

13. Test: Verify that .clone() will make a copy of the object and does not reference the original – Returns clone
    a. Status: ✓
    b. Documentation/Comments: Passed
    c. 

```
// const spacetime = require('./builds/spacetime.cjs')
import spacetime from './src/index.js'
import SpaceTime from './src/spacetime.js'
import _ from 'lodash';

// /**
//  * VALID
//  */
let s = new SpaceTime({
    epoch: 1698710400000,
    tz: 'america/new_york',
    silent: true,
    british: undefined,
    _weekStart: 1,
    _today: {}
})

console.log("Original date:", s.format('nice'));

// Make a copy
let ss = s.clone();
console.log("Clone of original:", ss.format('nice'));

// Manipulate clone
ss = ss.add(1, 'month');

console.log('Original after clone manipulation:', s.format('nice'));
console.log('Clone after clone manipulation', ss.format('nice'))
```

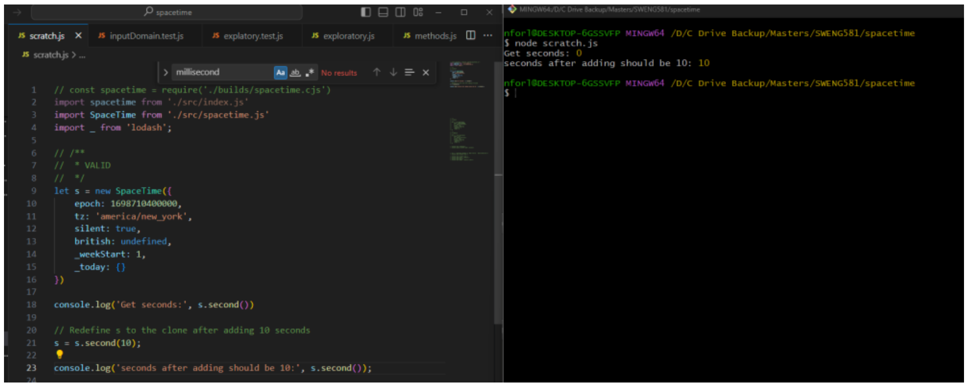
```
nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ node scratch.js
Original date: Oct 30th, 8:00pm
Clone of original: Oct 30th, 8:00pm
Original after clone manipulation: Oct 30th, 8:00pm
Clone after clone manipulation Nov 30th, 8:00pm

nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$
```

14. Test: Verify that .millisecond() will set or return the current number of milliseconds (0-999) - Returns a clone if setting and number if getting
    a. Status: ✓
    b. Documentation/Comments: Passed
    c. 

```
// const spacetime = require('./builds/spacetime.cjs')
import spacetime from './src/index.js'
import SpaceTime from './src/spacetime.js'
import _ from 'lodash';

// /**
//  * VALID
//  */
let s = new SpaceTime({
    epoch: 1698710400000,
    tz: 'america/new_york',
    silent: true,
    british: undefined,
    _weekStart: 1,
    _today: {}
})

console.log('Get milliseconds:', s.millisecond())

// Redefine s to the clone after adding 10 seconds
s = s.millisecond(10);

console.log('Milliseconds after adding should be 10:', s.millisecond());
```

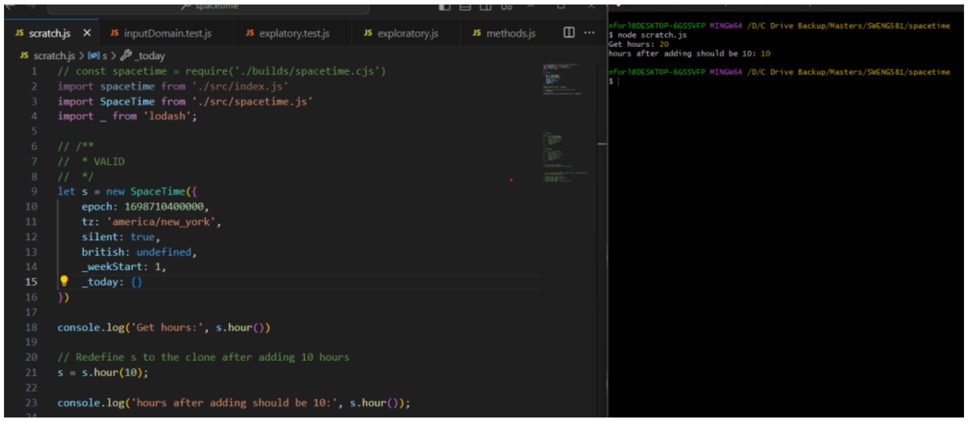
```
nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ node scratch.js
Get milliseconds: 0
Milliseconds after adding should be 10: 10

nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$
```

15. Test: Verify that .second() will set or return the current number of second (0-59) - Returns a clone if setting and number if getting
    a. Status: ✓
    b. Documentation/Comments: Passed

c. 

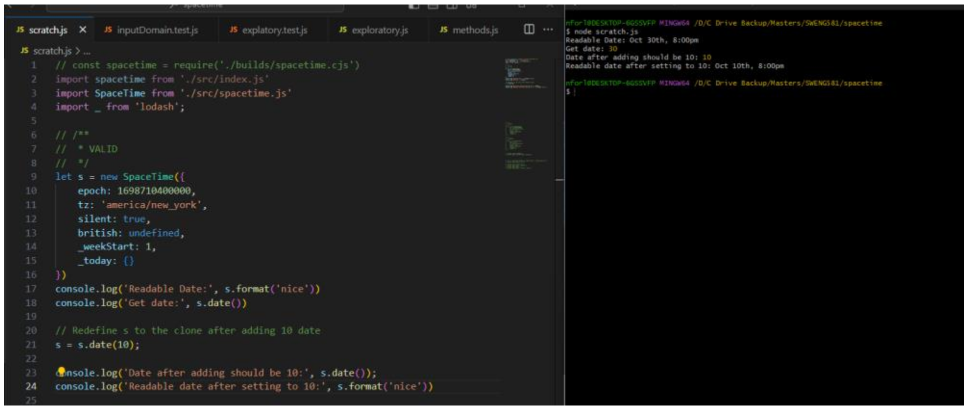
```
// const spacetime = require('./builds/spacetime.cjs')
import spacetime from './src/index.js'
import SpaceTime from './src/spacetime.js'
import _ from 'lodash';

// /**
//  * VALID
//  */
let s = new SpaceTime({
    epoch: 1698710400000,
    tz: 'america/new_york',
    silent: true,
    british: undefined,
    _weekStart: 1,
    _today: {}
})

console.log('Get seconds:', s.second())

// Redefine s to the clone after adding 10 seconds
s = s.second(10);

console.log('seconds after adding should be 10:', s.second());
```

```
nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ node scratch.js
Get seconds: 0
seconds after adding should be 10: 10

nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$
```

16. Test: Verify that .hour() will set or return the current number of hour(0-23) - Returns a clone if setting and number if getting
    a. Status: ✓
    b. Documentation/Comments: Passed
    c. 

```
// const spacetime = require('./builds/spacetime.cjs')
import spacetime from './src/index.js'
import SpaceTime from './src/spacetime.js'
import _ from 'lodash';

// /**
//  * VALID
//  */
let s = new SpaceTime({
    epoch: 1698710400000,
    tz: 'america/new_york',
    silent: true,
    british: undefined,
    _weekStart: 1,
    _today: {}
})

console.log('Get hours:', s.hour())

// Redefine s to the clone after adding 10 hours
s = s.hour(10);

console.log('hours after adding should be 10:', s.hour());
```

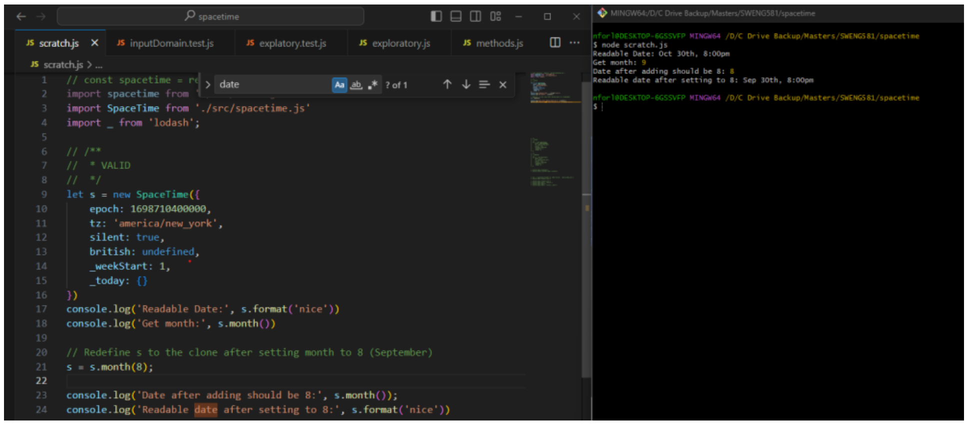
```
nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ node scratch.js
Get hours: 20
hours after adding should be 10: 10

nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$
```

17. Test: Verify that .date() will set or return the current date (1-31) - Returns clone if setting and number if getting
    a. Status: ✓
    b. Documentation/Comments: Passed
    c. 

```
// const spacetime = require('./builds/spacetime.cjs')
import spacetime from './src/index.js'
import SpaceTime from './src/spacetime.js'
import _ from 'lodash';

// /**
//  * VALID
//  */
let s = new SpaceTime({
    epoch: 1698710400000,
    tz: 'america/new_york',
    silent: true,
    british: undefined,
    _weekStart: 1,
    _today: {}
})
console.log('Readable Date:', s.format('nice'))
console.log('Get date:', s.date())

// Redefine s to the clone after adding 10 date
s = s.date(10);

console.log('Date after adding should be 10:', s.date());
console.log('Readable date after setting to 10:', s.format('nice'))
```

```
nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ node scratch.js
Readable Date: Oct 30th, 8:00pm
Get date: 30
Date after adding should be 10: 10
Readable date after setting to 10: Oct 10th, 8:00pm

nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$
```

18. Test: Verify that .month() will set or return the zero-based month-number(0-11) - Returns clone if getting and number if getting
    a. Status: ✓
    b. Documentation/Comments: Passed

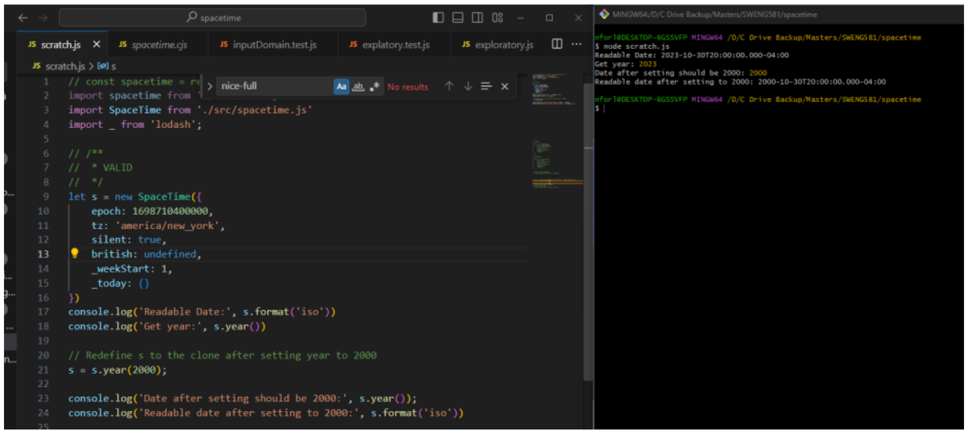


```
// const spacetime = r
import spacetime from
import SpaceTime from './src/spacetime.js'
import _ from 'lodash';

// /**
//  * VALID
//  */
let s = new SpaceTime({
    epoch: 1698710400000,
    tz: 'america/new_york',
    silent: true,
    british: undefined,
    _weekStart: 1,
    _today: {}
})
console.log('Readable Date:', s.format('nice'))
console.log('Get month:', s.month())

// Redefine s to the clone after setting month to 8 (September)
s = s.month(8);

console.log('Date after adding should be 8:', s.month());
console.log('Readable date after setting to 8:', s.format('nice'))
```

```
MINGW64:/D/C Drive Backup/Masters/SWENG581/spacetime
nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ node scratch.js
Readable Date: Oct 30th, 8:00pm
Get month: 9
Date after adding should be 8: 8
Readable date after setting to 8: Sep 30th, 8:00pm

nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$
```

c.

19. Test: Verify that .year() will set or return the 4-digit year as an integer – Returns clone if setting and return number if getting
    a. Status: ✓
    b. Documentation/Comments: Passed

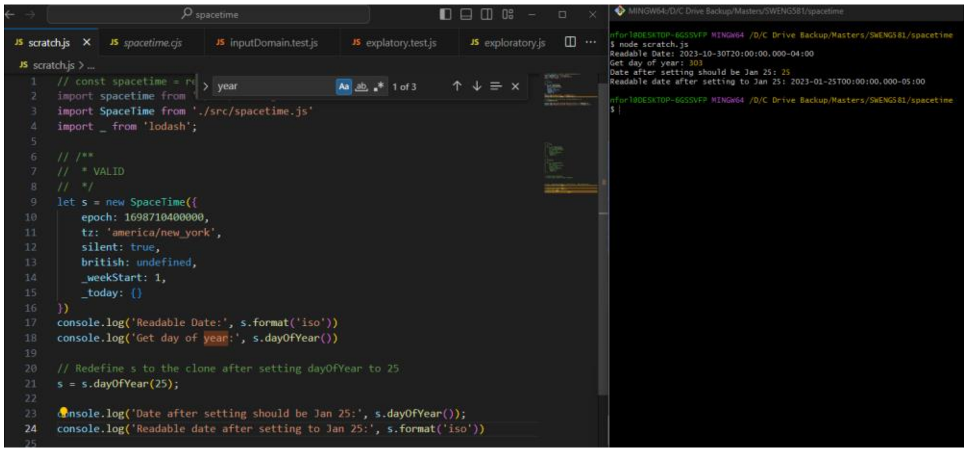


```
// const spacetime = r
import spacetime from
import SpaceTime from './src/spacetime.js'
import _ from 'lodash';

// /**
//  * VALID
//  */
let s = new SpaceTime({
    epoch: 1698710400000,
    tz: 'america/new_york',
    silent: true,
    british: undefined,
    _weekStart: 1,
    _today: {}
})
console.log('Readable Date:', s.format('iso'))
console.log('Get year:', s.year())

// Redefine s to the clone after setting year to 2000
s = s.year(2000);

console.log('Date after setting should be 2000:', s.year());
console.log('Readable date after setting to 2000:', s.format('iso'))
```

```
MINGW64:/D/C Drive Backup/Masters/SWENG581/spacetime
nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ node scratch.js
Readable Date: 2023-10-30T20:00:00.000-04:00
Get year: 2023
Date after setting should be 2000: 2000
Readable date after setting to 2000: 2000-10-30T20:00:00.000-04:00

nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$
```

c.

20. Test: Verify that .dayOfYear() will set or return the day of the year (1-366) - Return clone if setting and number if getting
    a. Status:
    b. Documentation/Comments: Passed



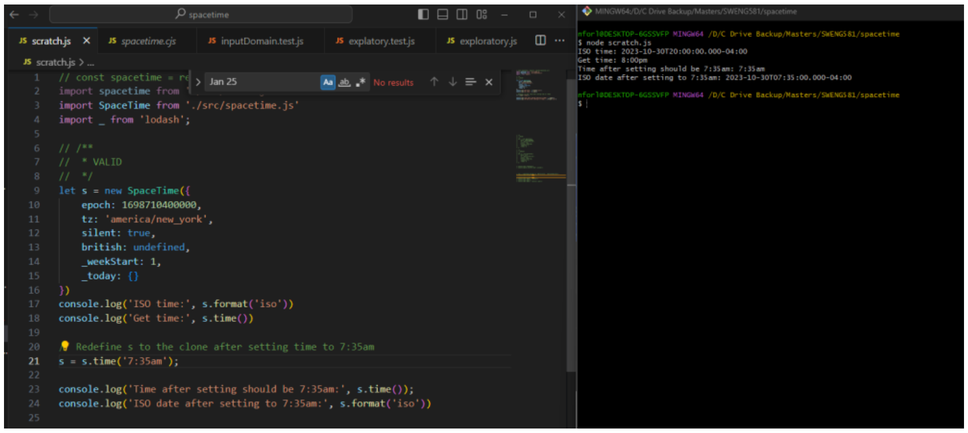
```
// const spacetime = r
import spacetime from
import SpaceTime from './src/spacetime.js'
import _ from 'lodash';

// /**
//  * VALID
//  */
let s = new SpaceTime({
    epoch: 1698710400000,
    tz: 'america/new_york',
    silent: true,
    british: undefined,
    _weekStart: 1,
    _today: {}
})
console.log('Readable Date:', s.format('iso'))
console.log('Get day of year:', s.dayOfYear())

// Redefine s to the clone after setting dayOfYear to 25
s = s.dayOfYear(25);

console.log('Date after setting should be Jan 25:', s.dayOfYear());
console.log('Readable date after setting to Jan 25:', s.format('iso'))
```

```
MINGW64:/D/C Drive Backup/Masters/SWENG581/spacetime
nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ node scratch.js
Readable Date: 2023-10-30T20:00:00.000-04:00
Get day of year: 303
Date after setting should be Jan 25: 25
Readable date after setting to Jan 25: 2023-01-25T00:00:00.000-05:00

nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$
```

c.

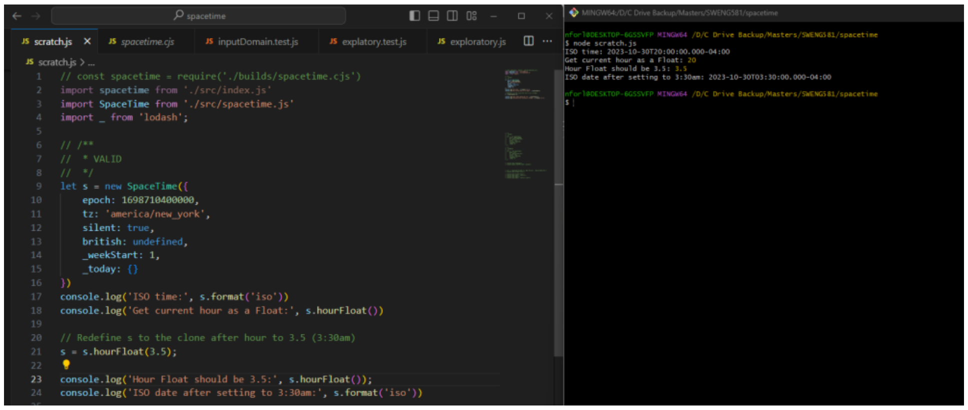
21. Test: Verify that .time() will set or return the time – Return clone if setting and number if getting
    a. Status: ✓

b. Documentation/Comments: Passed* - The documentation says it should return a number but it should be returning a string. The function is working correctly, but the documentation needs to be updated.

c. 

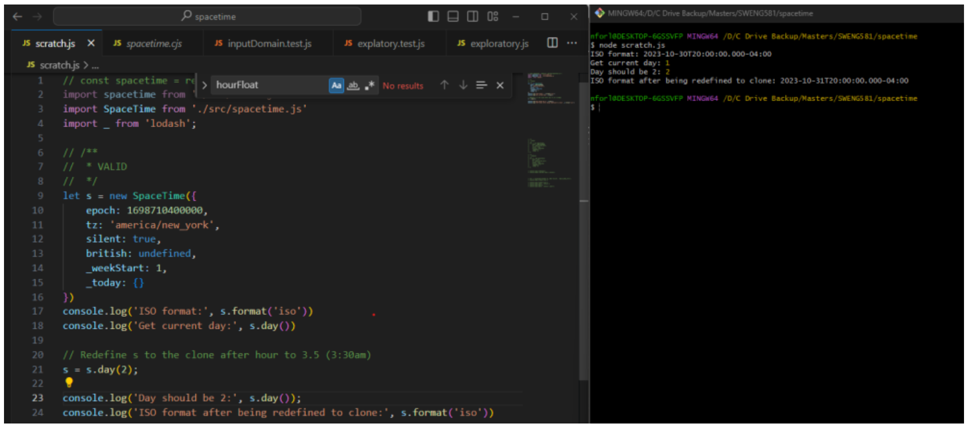
```
// const spacetime = re
import spacetime from
import SpaceTime from './src/spacetime.js'
import _ from 'lodash';

// /**
//  * VALID
//  */
let s = new SpaceTime({
    epoch: 1698710400000,
    tz: 'america/new_york',
    silent: true,
    british: undefined,
    _weekStart: 1,
    _today: {}
})
console.log('ISO time:', s.format('iso'))
console.log('Get time:', s.time())

 Redefine s to the clone after setting time to 7:35am
s = s.time('7:35am');

console.log('Time after setting should be 7:35am:', s.time());
console.log('ISO date after setting to 7:35am:', s.format('iso'))
```

```
nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ node scratch.js
ISO time: 2023-10-30T20:00:00.000-04:00
Get time: 8:00pm
Time after setting should be 7:35am: 7:35am
ISO date after setting to 7:35am: 2023-10-30T07:35:00.000-04:00

nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ |
```

22. Test: Verify that .hourFloat() will set or return the hour + minute in decimal form – Returns clone if setting and number if getting
    a. Status: ✓
    b. Documentation/Comments: Passed
    c. 

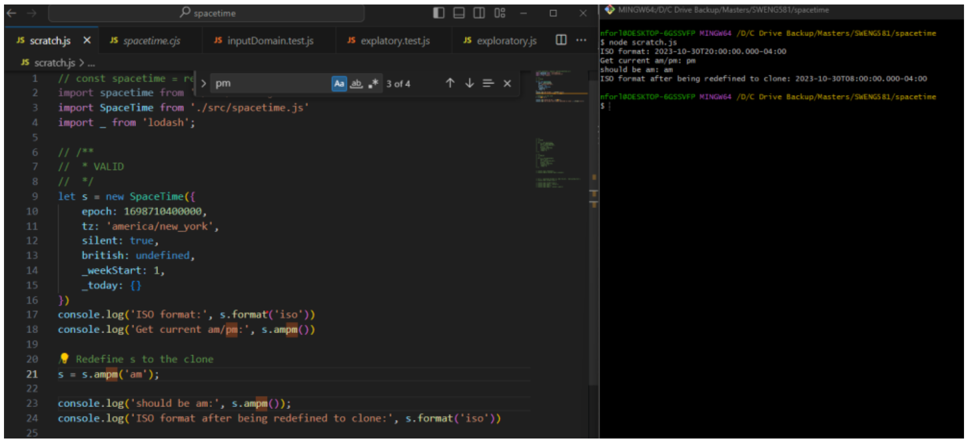
```
// const spacetime = require('./builds/spacetime.cjs')
import spacetime from './src/index.js'
import SpaceTime from './src/spacetime.js'
import _ from 'lodash';

// /**
//  * VALID
//  */
let s = new SpaceTime({
    epoch: 1698710400000,
    tz: 'america/new_york',
    silent: true,
    british: undefined,
    _weekStart: 1,
    _today: {}
})
console.log('ISO time:', s.format('iso'))
console.log('Get current hour as a Float:', s.hourFloat())

// Redefine s to the clone after hour to 3.5 (3:30am)
s = s.hourFloat(3.5);

console.log('Hour Float should be 3.5:', s.hourFloat());
console.log('ISO date after setting to 3:30am:', s.format('iso'))
```

```
nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ node scratch.js
ISO time: 2023-10-30T20:00:00.000-04:00
Get current hour as a Float: 20
Hour Float should be 3.5: 3.5
ISO date after setting to 3:30am: 2023-10-30T03:30:00.000-04:00

nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ |
```

23. Test: Verify that .day() will set or return the day of the week as an integer, starting on Sunday (day-0) - Will return a clone if setting or a number if getting
    a. Status: ✓
    b. Documentation/Comments: Passed
    c. 

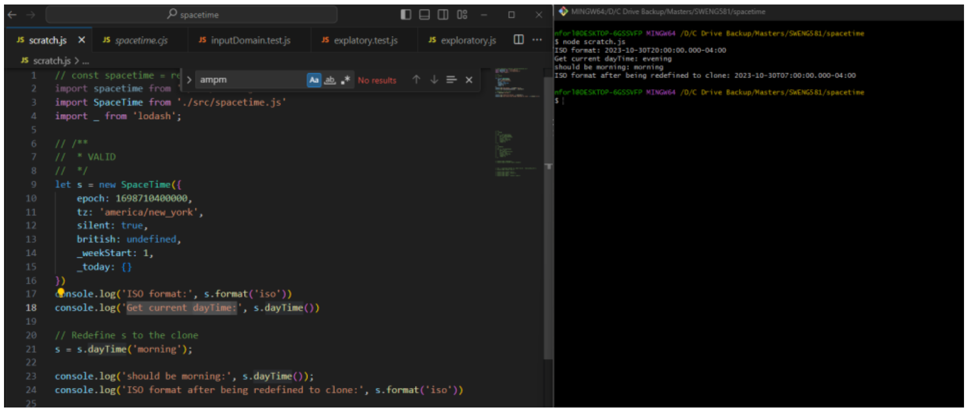
```
// const spacetime = re
import spacetime from
import SpaceTime from './src/spacetime.js'
import _ from 'lodash';

// /**
//  * VALID
//  */
let s = new SpaceTime({
    epoch: 1698710400000,
    tz: 'america/new_york',
    silent: true,
    british: undefined,
    _weekStart: 1,
    _today: {}
})
console.log('ISO format:', s.format('iso'))
console.log('Get current day:', s.day())

// Redefine s to the clone after hour to 3.5 (3:30am)
s = s.day(2);

console.log('Day should be 2:', s.day());
console.log('ISO format after being redefined to clone:', s.format('iso'))
```

```
nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ node scratch.js
ISO format: 2023-10-30T20:00:00.000-04:00
Get current day: 1
Day should be 2: 2
ISO format after being redefined to clone: 2023-10-31T20:00:00.000-04:00

nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ |
```

24. Test: Verify that .ampm() will set or return whether the time is am or pm – Returns clone if setting and string if getting.
    a. Status: ✓
    b. Documentation/Comments: Passed
    c. 

```js
// const spacetime = re
import spacetime from
import SpaceTime from './src/spacetime.js'
import _ from 'lodash';

// /**
//  * VALID
//  */
let s = new SpaceTime({
    epoch: 1698710400000,
    tz: 'america/new_york',
    silent: true,
    british: undefined,
    _weekStart: 1,
    _today: {}
})
console.log('ISO format:', s.format('iso'))
console.log('Get current am/pm:', s.ampm())

// Redefine s to the clone
s = s.ampm('am');

console.log('should be am:', s.ampm());
console.log('ISO format after being redefined to clone:', s.format('iso'))
```

```
nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ node scratch.js
ISO format: 2023-10-30T20:00:00.000-04:00
Get current am/pm: pm
should be am: am
ISO format after being redefined to clone: 2023-10-30T08:00:00.000-04:00

nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$
```

25. Test: Verify that .dayTime() will return or set the general time-of-day – Returns clone if setting and string if getting
    a. Status: ✓
    b. Documentation/Comments: Passed
    c. 

```js
// const spacetime = re
import spacetime from
import SpaceTime from './src/spacetime.js'
import _ from 'lodash';

// /**
//  * VALID
//  */
let s = new SpaceTime({
    epoch: 1698710400000,
    tz: 'america/new_york',
    silent: true,
    british: undefined,
    _weekStart: 1,
    _today: {}
})
console.log('ISO format:', s.format('iso'))
console.log('Get current dayTime:', s.dayTime())

// Redefine s to the clone
s = s.dayTime('morning');

console.log('should be morning:', s.dayTime());
console.log('ISO format after being redefined to clone:', s.format('iso'))
```

```
nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$ node scratch.js
ISO format: 2023-10-30T20:00:00.000-04:00
Get current dayTime: evening
should be morning: morning
ISO format after being redefined to clone: 2023-10-30T07:00:00.000-04:00

nforl@DESKTOP-6GSSVFP MINGW64 /D/C Drive Backup/Masters/SWENG581/spacetime
$
```

Nr	What done?	Status	Comment
1	Verify that the .set() function changes the date to the input parameter and a clone is returned	✓	
2	Verify that the .isValid() function confirms a valid date and time according to the rules of the Gregorian calendar and the JavaScript Date object – Return clone	✓	The docs say the return type is a clone. The correct return type should be Boolean
3	Verify that the .log() function will pretty-print the date to the console, for nicer debugging – Return clone	✓	
4	Verify that the .progress() function correctly calculates the progress of a given moment within a specific time frame, such as a day, week, month, or year. The function takes a moment in time and returns a value between 0 and 1, indicating how far that moment is in relation to the start and end of the specified time frame – Return Object	✓	
5	Verify that the .leapYear() function correctly returns if the current year is a leap year or not – Returns Boolean	✓	
6	Verify that the .inDST() function corectly returns if the current selected time zone is in daylight savings time – Returns Boolean	✓	
7	Verify that the .hasDST() function validated if the timezone ever uses daylight-savings and return boolean	✓	
8	Verify that the .offset() function correctly calculates the current time difference from Coordinated Universal Time (UTC) while accounting for Daylight Saving Time (DST) - Returns number.	✓	
9	Verify that .isAsleep() correctly states if a user is asleep, or in this case, the time is between 10pm and 8am – Returns boolean.	✓	
10	Verify that the .i18n() internalization function correctly allows a user to switch between different sets of names based on the desired language or regional conventions. Returns clone	✓	
11	Verify that the .weekStart() function can be changed to whichever day is desired. The default starting day of the week for this application is Monday – Return clone	✓	
12	Verify that the .daysInMonth() function correctly returns the number of days witin the current month (December has 31 days; June has 30, etc.) - Returns number	✓	

13	Verify that .clone() will make a copy of the object and does not reference the original – Returns clone		
14	Verify that .millisecond() will set or return the current number of milliseconds (0-999) - Returns a clone if setting and number if getting	✓	
15	Verify that .second() will set or return the current number of second (0-59) - Returns a clone if setting and number if getting	✓	
16	Verify that .hour() will set or return the current number of hour(0-23) - Returns a clone if setting and number if getting	✓	
17	Verify that .date() will set or return the current date (1-31) - Returns clone if setting and number if getting	✓	
18	Verify that .month() will set or return the zero-based month-number(0-11) - Returns clone if getting and number if getting	✓	
19	Verify that .year() will set or return the 4-digit year as an integer – Returns clone if setting and return number if getting	✓	
20	Verify that .dayOfYear() will set or return the day of the year (1-366) - Return clone if setting and number if getting	✓	
21	Verify that .time() will set or return the time – Return clone if setting and number if getting	✓	The documentation says it should return a number but it should be returning a string. The function is working correctly, but the documentation needs to be updated
22	Verify that .hourFloat() will set or return the hour + minute in decimal form – Returns clone if setting and number if getting	✓	
23	Verify that .day() will set or return the day of the week as an integer, starting on Sunday (day-0) - Will return a clone if setting or a number if getting	✓	
24	Verify that .ampm() will set or return whether the time is am or pm – Returns clone if setting and string if getting	✓	

25	Verify that .dayTime() will return or set the general time-of-day – Returns clone if setting and string if getting	✓	