

Code Coverage Results

Azure DevOps has started rollout of changes to disable communication over TLS 1.0 and TLS 1.1. This change is permanent and if your tools are dependent on TLS 1.0/1.1 for communication with Azure DevOps, please take necessary actions to enable TLS1.2, as detailed [in the blog](#).

#20220606.2 updated the pipeline

Run new

AlexSandro19.testing_project_delivery_system

This run is being retained as one of 3 recent runs by pipeline.

View retention leases

Summary Tests Code Coverage

Manually run by Alexandru Sandrovshii

View change

Repository and version
 AlexSandro19/testing_project_delivery_system
 unit-testing 372bc4c2

Time started and elapsed
 Today at 11:33 PM
 47s

Related
 0 work items
 2 published

Tests and coverage
 100% passed
 41.84% covered

Jobs

Name	Status	Duration
Job	Success	40s

alex15Sr / Testing Final Project / Pipelines / AlexSandro19.testing_proje... / 20220606.2

Azure DevOps has started rollout of changes to disable communication over TLS 1.0 and TLS 1.1. This change is permanent and if your tools are dependent on TLS 1.0/1.1 for communication with Azure DevOps, please take necessary actions to enable TLS1.2, as detailed [in the blog](#).

AlexSandro19.testing_project_delivery_system

This run is being retained as one of 3 recent runs by pipeline. [View retention leases](#)

Summary Tests Code Coverage

Summary

1 Run(s) Completed (1 Passed, 0 Failed)

193 Total tests +193		100% Pass percentage 100%	13s 233ms Run duration +13s 233ms	0 Tests not reported
-----------------------------------	--	--	--	--------------------------------

Bug ▾ Link

Filter by test or run name

Test run ▾ Column Options

Tags ▾ Test file ▾ Owner ▾ Aborted (+1) ▾

project/build/results?buildId=133&view=ms.vss-test-web.build-test-results-tab

Azure DevOps has started rollout of changes to disable communication over TLS 1.0 and TLS 1.1. This change is permanent and if your tools are dependent on TLS 1.0/1.1 for communication with Azure DevOps, please take necessary actions to enable TLS1.2, as detailed in the blog.

Jobs in run #2022060...

AlexSandro19.testing_project_deliveri

Jobs

Job	40s
Initialize job	4s
Checkout AlexSandr...	3s
Install Node.js	<1s
install dependencies	10s
run tests	14s
PublishTestResults	2s
PublishCodeCoverag...	2s
CopyFiles	1s
PublishBuildArtifacts	1s
Post-job: Checkout ...	<1s
Finalize Job	<1s

Job

View raw log

```
1 Pool: Azure Pipelines
2 Image: ubuntu-latest
3 Agent: Hosted Agent
4 Started: Today at 11:34 PM
5 Duration: 40s
6
7 Job preparation parameters
8 fr 5 queue time variables used
9 2 artifacts produced
10 100% tests passed
```

No new notifications (Off)

Azure DevOps has started rollout of changes to disable communication over TLS 1.0 and TLS 1.1. This change is permanent and if your tools are dependent on TLS 1.0/1.1 for communication with Azure DevOps, please take necessary actions to enable TLS1.2, as detailed in the blog.

#20220606.2 updated the pipeline

AlexSandro19.testing_project_delivery_system

Run new

This run is being retained as one of 3 recent runs by pipeline.

View retention leases

Summary Tests Code Coverage

Assemblies:	4
Classes:	15
Files:	15
Covered lines:	1241
Uncovered lines:	1725
Coverable lines:	2966
Total lines:	2970
Line coverage:	41.8% (1241 of 2966)
Covered branches:	199
Total branches:	353
Branch coverage:	56.3% (199 of 353)

How code coverage helped

```
utility > JS utility.functions.js > getDateInSqlFormat
You, 2 hours ago | 1 author (You)
1  const { execute } = require("../database/mysql.connector.js");
2
3  const getDateInSqlFormat = (date = null) => {
4    // console.log("date: ", date)
5    // console.log("typeof date: ", typeof date)
6    if (date instanceof Date && !isNaN(date)) {
7      const year = date.getFullYear()
8      const month = ((date.getMonth() + 1) >= 10) ? `${date.getMonth() + 1}` : `0${date.getMonth() + 1}`
9      const day = (date.getDate() >= 10) ? `${date.getDate()}` : `0${date.getDate()}`
10     const hours = (date.getHours() >= 10) ? `${date.getHours()}` : `0${date.getHours()}`
11     const minutes = (date.getMinutes() >= 10) ? `${date.getMinutes()}` : `0${date.getMinutes()}`
12     const seconds = (date.getSeconds() >= 10) ? `${date.getSeconds()}` : `0${date.getSeconds()}`
13     // Date format that Mysql expects to receive: YYYY-MM-DD hh:mm:ss
14     const res = `${year}-${month}-${day} ${hours}:${minutes}:${seconds}`
15     // console.log("getDateInSqlFormat: ", res)
16     return res
17   } else {
18     return null
19   }
20 }
21
22 module.exports = {
23   getDateInSqlFormat
24 }
25
```

PROBLEMS OUTPUT GITLENS TERMINAL DEBUG CONSOLE

```
PASS tests/utility/utility.generator.test.js
PASS tests/utility/utility.functions.test.js
PASS tests/utility/utility.calculations.test.js
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	74.07	82.35	63.63	74.07	
utility.calculations.js	55.88	100	33.33	55.88	24-37,45-58,71-84,87-89
utility.functions.js	100	44.44	100	100	8-12
utility.generators.js	93.65	92.85	100	93.65	39-42

Test Suites: 3 passed, 3 total

We have a function that converts the Date object into a string in the “YYYY-MM-DD hh:mm:ss” that MySQL expects to receive.

Before implementing the code coverage, we were testing if a passed Date object returns the string in the correct format and what is the output with invalid input values.

```
PASS tests/utility/utility.functions.test.js
Run tests for getDateInSqlFormat function
✓ Pass a valid Date object and expect a string representation of it if format 'YYYY-MM-DD hh:mm:ss' (9 ms)
✓ Pass a valid Date object with month smaller than 10 and expect a string representation of it (1 ms)
✓ Pass an invalid Date object and expect null (1 ms)
✓ Pass a invalid object and expect null (1 ms)
✓ Pass a invalid string and expect null (1 ms)
✓ Pass a invalid number and expect null
✓ Pass a invalid null vallue and expect null
✓ Pass a nothing and expect null (1 ms)
```


However, in the code we have some logic that appends a 0 to the value if it is smaller than 10. This portion of code wasn't tested and code coverage showed that by specifying the % and 'Uncovered Lines #s': 8-12.

After adding additional tests that cover those use cases, we get 100% coverage.

```

PASS tests/utility/utility.functions.test.js
Run tests for getDateInSqlFormat function
  ✓ Pass a valid Date object and expect a string representation of it in format 'YYYY-MM-DD hh:mm:ss' (5 ms)
  ✓ Pass a valid Date object with month and day smaller than 10 and expect a string representation of it in format 'YYYY-MM-DD hh:mm:ss'
  ✓ Pass a valid Date object with month and day bigger than 10 and expect a string representation of it in format 'YYYY-MM-DD hh:mm:ss'
  ✓ Pass a valid Date object with hour, minutes and seconds smaller than 10 and expect a string representation of it in format 'YYYY-MM-DD hh:mm:ss' (1 ms)
  ✓ Pass a valid Date object with hour, minutes and seconds bigger than 10 and expect a string representation of it in format 'YYYY-MM-DD hh:mm:ss'
  ✓ Pass an invalid Date object and expect null (1 ms)
  ✓ Pass a invalid object and expect null
  ✓ Pass a invalid string and expect null (1 ms)
  ✓ Pass a invalid number and expect null
  ✓ Pass a invalid null vallue and expect null

```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	74.07	97.43	63.63	74.07	
utility.calculations.js	55.88	100	33.33	55.88	24-37,45-58,71-84,87-89
utility.functions.js	100	100	100	100	
utility.generators.js	93.65	92.85	100	93.65	39-42

Test Suites: 3 passed, 3 total
 Tests: 46 passed, 46 total
 Snapshots: 0 total
 Time: 3.175 s