

For this section of the implementation of HAMPR, I began by looking at all the tests that were output to the console using **npm test**. I then proceeded to target all the tests for '**handleRequestMachine**' and began from the top. As I started with the first case, I singled it out and began tracing the request input parameters and the output parameters so that I know where I start and where I end. With this information outlined, the process became trivial as I would follow the path check in the database or cache and make appropriate changes to either while returning the appropriate response.

```
PASS test/api.test.ts
PASS test/simulation.test.ts
```

- Console

```
console.log
```

(index)	Resource	Run 1 Units	Run 1 %	Run 2 Units	Run 2 %	Run 3 Units	Run 3 %	Run 4 Units	Run 4 %
0	'IdentityProviderClient'	256	'0.02%	256	'0.02%	256	'0.01%	256	'0.02%
1	'SmartMachineClient'	32256	'1.89%	32256	'1.91%	32256	'1.87%	32256	'1.89%
2	'MachineStateTable'	1648400	'96.68%	1648400	'97.69%	1648400	'95.48%	1648400	'96.77%
3	'DataCache'	22556	'1.32%	22556	'1.34%	22556	'1.31%	22556	'1.32%

```
at Object.<anonymous> (test/simulation.test.ts:160:13)
```

```
console.log
```

(index)	Run	Cache Hits	Cache Misses	Hit Rate
0	1	3774	2130	'63.92%
1	2	3839	2080	'64.86%
2	3	3694	2166	'63.04%
3	4	3779	2110	'64.17%

```
at Object.<anonymous> (test/simulation.test.ts:161:13)
```

```
console.log
```

(index)	Run	Cache Hits	DB Accesses	Hit/Access Ratio
0	1	3774	6721	'0.5615'
1	2	3839	6721	'0.5712'
2	3	3694	6721	'0.5496'
3	4	3779	6721	'0.5623'

```
at Object.<anonymous> (test/simulation.test.ts:162:13)
```

```
Test Suites: 2 passed, 2 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        1.106 s
```