**Валидация, Верификация и Тестови план**

*SHA-512*

**Ревизионен лист**

|  |  |  |
| --- | --- | --- |
| **Релийз No.** | **Дата** | **Описание** |
| Rev. 0 | 4/26/2020 | Валентин Георгиев Александров |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Валидация, Верификация и Тестови план**

**СЪДЪРЖАНИЕ**

Page #

1.0 Обща информация 1-1

1.1 Цел 1-1

1.2 Обхват 1-1

1.3 Общ преглед на системата 1-1

1.5 Съкращения 1-1

2.0 Оценка на тестването 2-1

2.1 Матрица за проследяване на изискванията (Requirements Traceability Matrix) 2-1

2.2 Критерии за оценка на тест 2-1

3.0 Описание на тестовите случаи 3-1

3.1 Функционални тестове 3-1

3.1.х [Test Identifier] 3-1

 *Тестови / функционални връзки* 3-1

 *Средства за контрол* 3-1

 *Входни данни* 3-1

 *Изходни данни* 3-1

 *Процедура за тестване* 3-1

3.2 Модулни (Unit) тестове 3-1

3.2.х [Test Identifier] 3-1

 *Средства за контрол* 3-1

 *Входни данни* 3-2

 *Изходни данни* 3-2

 *Покритие на теста* 3-2

# Обща информация

## 1.1 Цел

*Целта е да верифицираме, че софтуера поддържа всички функционалности и че има документация.*

*Искаме да валидираме, че функционалностите отговарят на желаното поведение и че софтуера минава през необходимите изисквания.*

## 1.2 Обхват

*Обхвата на Валидиране обхваща ръчна проверка дали софтуера работи спрямо изискваните функционалности, както и проверка на съществуващите тестове.*

## 1.3 Общ преглед на системата

*Системата представлява програма, която имплементира SHA-512 алгоритъма и предоставя възможността за използването на този алгоритъм като се чете от конзолата, от файл или чрез задаване на даден вход в кода.*

## 1.5 Съкращения

*SHA-512 -* Secure Hash Algorithm(Сигурен Хеширащ Алгоритъм)

# ОЦенка на тестването

## 2.1 Матрица за проследяване на изискванията (Requirements Traceability Matrix)

*Подгответе функция / тестова матрица, която изброява всички функции на приложението на една ос и ги препраща към всички тестове, включени в плана за тестване*

|  |  |  |
| --- | --- | --- |
| *Функционалност* | *Описание на теста* | *Статус* |
| *Да се хешира зададено съобщение* | *Теста проверява дали подаденото съобщение ще бъде хеширано успешно.* | *Успешно минал* |
| *Да се хешира съдържанието на файл* | *Теста проверява дали прочетеното съдържание на файла, по зададената пътека ще бъде хеширано правилно.* | *Успешно минал* |
| *Да се хешира съобщение въведено от клавиатурата.* | *Теста проверява дали предварително прочетено съобщение от клавиатурата ще бъде хеширано успешно.* | *Успешно минал* |

*.*

## 2.2 Критерии за оценка на тест

*Опишете специфичните критерии, на които трябва да отговаря всеки сегмент на системата / подсистемата. Такива критерии са описани от потребителя на системата / подсистемата и обикновено представляват комбинация от функционални и нефункционални изисквания, като например обработка на данни в определен период от време, изготвяне на отчет или отговаряне на онлайн заявка в рамките на определен период от време.*

|  |  |  |
| --- | --- | --- |
| *Тест* | *Критерии* | *Статус* |
| *Дали подаденото съобщение е хеширано правилно.* | *Проверява дали зададеното съобщение е правилно хеширано.* | *Успешно минал.* |
| *Дали прочетеното съдържание на файла е успешно хеширано.* | *Файлът е прочетен успешно без хвърляне на изключения.*  *Съдържанието е подадено на хеширащатата функция. Тя успява успешно да го хешира. Четенето от файла става в нормално време.* | *Успешно минал.* |
| *Дали предварително въведено съобщение от клавиатурата е успешно въведено.* | *Подава се предварително прочетено от клавиатурата съобщение и се проверява дали е правилно прочетено.* | *Успешно минал.* |

# Описание на тестовите случаи

## 3.1 Функционални тестове

### 3.1.1 [FkstHashConsole]

Трябва да се провери дали дали метода на класа връща хешираната версия на входа от клавиатурата.

### *Тестови / функционални връзки*

*Проследява кой тест/функция изпълнява от Матрица за проследяване на изискванията.*

### *Средства за контрол*

*Автоматични.*

### *Входни данни*

civic

### *Изходни данни*

38679e1931f619f09dcb06eca3bb6908885c73a03756732ddac0782cad9742d4d18cd990cfe47da2d89e1fc53532af9583056d71b2d6dac63c6f5716c5f3063

### *Процедура за тестване*

npm run test  
  
Може и ръчно да се провери, като се стартира приложението.

### 3.1.2 [FkstHashFile]

Трябва да прочете съдържанието на файла, за който е подадена пътека до него и да хешира това съдържание посредством sha-512.

### *Тестови / функционални връзки*

*Проследява кой тест/функция изпълнява от Матрица за проследяване на изискванията.*

### *Средства за контрол*

*Посочете дали тестът трябва да се контролира с ръчни, полуавтоматични или автоматични средства.*

### *Входни данни*

C:\\Users\\VALKA\\Desktop\\VVPS\_PROJECT\\repositories\\sha-project\\textFile.txt

### *Изходни данни*

c5493409580caa9130c4a1e291e98c5514d75d92fa24a22cea60540eda964976278fe901313f4fcdba97523da40455c4282e31b8b4b0268e8436be7dcbf1021d

### *Процедура за тестване*

npm run test

### 3.1.3 [FkstHashString]

Трябва да бъде зададено дадено съобщение и след това метода на класа, ще го хешира.

### *Тестови / функционални връзки*

*Проследява кой тест/функция изпълнява от Матрица за проследяване на изискванията.*

### *Средства за контрол*

*Посочете дали тестът трябва да се контролира с ръчни, полуавтоматични или автоматични средства.*

### *Входни данни*

civic

### *Изходни данни*

### *38679e1931f619f09dcb06eca3bb6908885c73a03756732ddac0782cad9742d4d18cd990cfe47da2d89e1fc53532af9583056d71b2d6dac63c6f5716c5f3063*

### *Процедура за тестване*

npm run test

## 3.2 Модулни (Unit) тестове

*Описват се всички модулни тестове, които ще се проведат. Всеки тест трябва да бъде в отделен раздел, 3.2.1 – 3.2.x.*

### 3.2.1 [hash]

*При даден вход, трябва да въден хешираната версия на входа.*

### *Средства за контрол*

*Автоматично.*

### *Входни данни*

0110100001101111011011100110010001100001100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000101000

### *Изходни данни*

[

'0110100001101111011011100110010001100001100000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

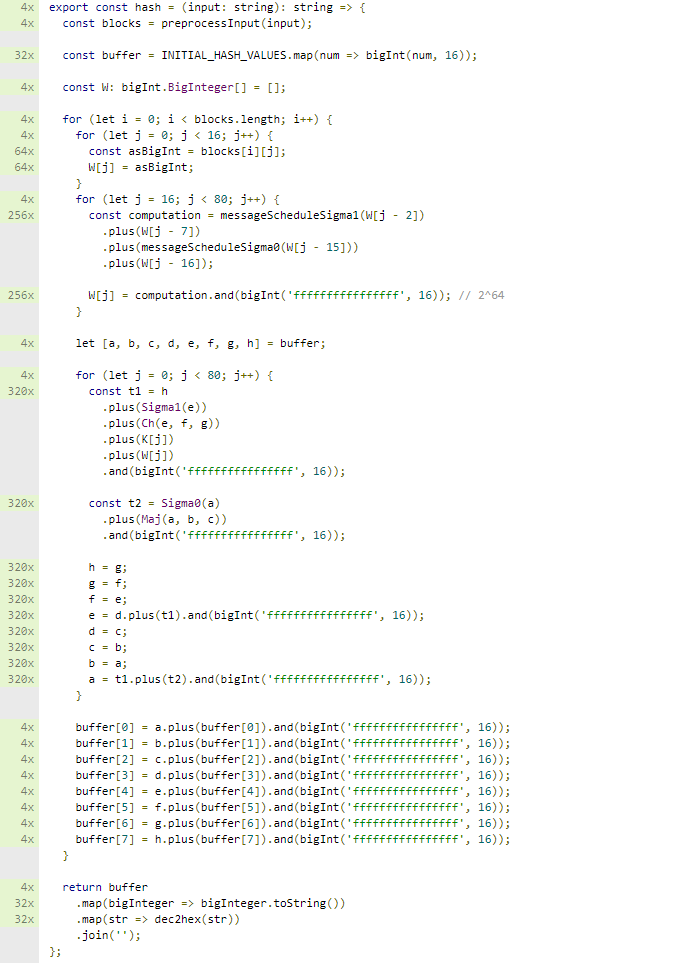
'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000101000',

]

### *Покритие на теста*



### 3.2.2 [padMessage]

*Трябва съобщението, представено като съвкупност от бинарни числа, да го преобрази до*

*Формат, с дължина кратна на 1024, като вземе всички бинарни числа, добави 1 единица, добави необходими брой нули и накрая добави константа изразена, чрез бинарно число.*

### *Средства за контрол*

*автоматично*

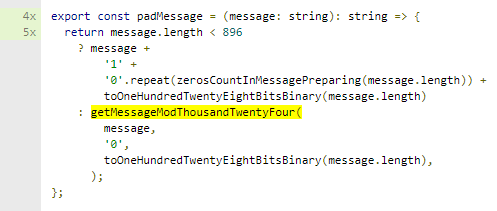
### *Входни данни*

0110100001101111011011100110010001100001

### *Изходни данни*

0110100001101111011011100110010001100001100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000101000

* *Покритие на теста*



### 3.2.3 [separateBlockByWords]

*Трябва да раздели блок на думи по 64 бита.*

### *Средства за контрол*

*Автоматично*

### *Входни данни*

0110100001101111011011100110010001100001100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000101000

### *Изходни данни*

[

'0110100001101111011011100110010001100001100000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000000000',

'0000000000000000000000000000000000000000000000000000000000101000',

]

*Покритие на теста*

