Software Testing Laboratory
(CS6474)
Assignment 05 :TCASES TOOL

Tapas Manna
223CS3152
Master of Technology
223cs3152@nitrkl.ac.in

Department of Computer Science Engineering
NIT, Rourkela

February 08, 2024

Contents

1 Annotations-Test

1.1 Specification

System: Examples Usage: ADD or Create different geometrical shape We can add shape by using addShape function by passing parameter to 1) Type: CIRCLE RECTANGLE,LINE etc 2) Size: In cm 3) Colour: RED ,GREEN etc Function: addShape

1.2 XML Code

```
<!--
    System: Examples
    Usage: ADD or Create different geometrical shape
    We can add shape by using addShape function by passing parameter to
    1) Type: CIRCLE RECTANGLE,LINE etc
    2) Size: In cm
    3) Colour: RED ,GREEN etc
    Function: addShape
-->

<System name="Examples">

    <Function name="addShape">
        <!-- Test case annotations -->
        <Has name="pageType" value="Page"/>
        <Has name="pageName" value="page"/>
        <Has name="pageValue" value="new Page()"/>

        <Input>

            <Var name="Type">
                <!-- Variable binding annotations -->
                <Has name="varType" value="Shape"/>
                <Has name="varName" value="shape"/>
                <Has name="varEval" value="new Shape"/>

                <Value name="SQUARE"/>
                <Value name="CIRCLE"/>
                <Value name="LINE" property="1D"/>
            </Var>

            <Var name="Size">
                <!-- Variable binding annotations -->
                <Has name="varType" value="int"/>
                <Has name="varName" value="size"/>
                <Has name="varApply" value="setSize"/>

                <Value name="1"/>
                <Value name="10"/>
                <Value name="100" property="Large"/>
            </Var>

            <Var name="Color">
                <!-- Variable binding annotations -->
                <Has name="varType" value="String"/>
                <Has name="varName" value="color"/>
                <Has name="varApply" value="setColor"/>
```

```
                <Value name="red"/>
                <Value name="green"/>
                <Value name="blue"/>
            </Var>
        </Input>
    </Function>
</System>
```

1.3 Test cases screenshot

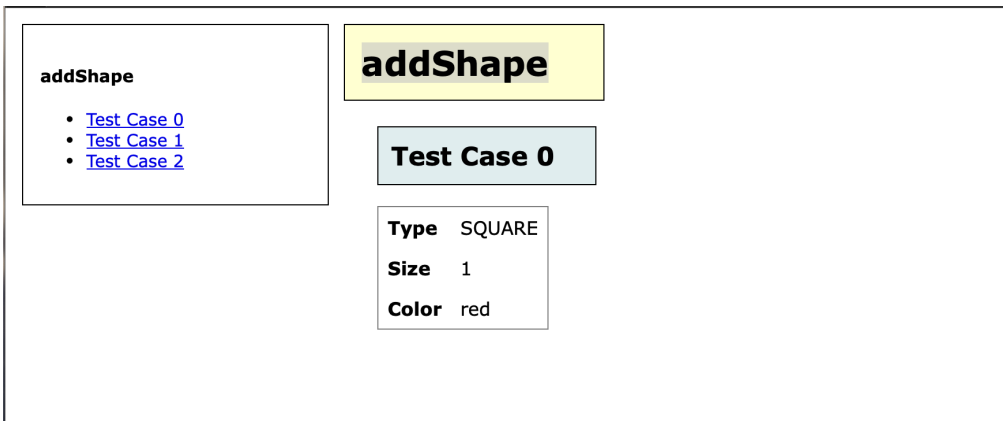

Figure 1: TCases Html View

2 Find-Input

2.1 Specification

Usage: find pattern file

Locates one or more instances of a given pattern in a text file.

All lines in the file that contain the pattern are written to standard output. A line containing the pattern is written only once, regardless of the number of times the pattern occurs in it.

The pattern is any sequence of characters whose length does not exceed the maximum length of a line in the file. To include a blank in the pattern, the entire pattern must be enclosed in quotes ("). To include a quotation mark in the pattern, two quotes in a row ("") must be used.

2.2 XML Code

```
<System name="Examples">
  <Function name="find">
    <!--
         Usage: find pattern file

         Locates one or more instances of a given pattern in a text file.

         All lines in the file that contain the pattern are written to standard
    output. A
         line containing the pattern is written only once, regardless of the
    number of
         times the pattern occurs in it.

         The pattern is any sequence of characters whose length does not exceed
    the
         maximum length of a line in the file. To include a blank in the pattern,
    the
         entire pattern must be enclosed in quotes ("). To include a quotation
    mark in the
         pattern, two quotes in a row ("") must be used.
      -->
    <Input type="arg">
      <VarSet name="pattern" when="fileExists">
        <Var name="size">
          <Value name="empty" property="empty"/>
          <Value name="singleChar" property="singleChar"/>
          <Value name="manyChars"/>
        </Var>
        <Var name="quoted">
          <Value name="yes" property="quoted"/>
          <Value name="no" whenNot="empty"/>
          <Value name="unterminated" failure="true"/>
        </Var>
        <Var name="blanks" whenNot="empty">
          <Value name="none"/>
          <Value name="one" when="quoted, singleChar"/>
          <Value name="many">
            <When>
              <AllOf property="quoted">
                <Not property="singleChar"/>
              </AllOf>
            </When>
```

```
          </Value>
        </Var>
        <Var name="embeddedQuotes" whenNot="empty, singleChar">
          <Value name="none"/>
          <Value name="one"/>
          <Value name="many" once="true"/>
        </Var>
      </VarSet>

      <Var name="fileName">
        <Value name="defined" property="fileName"/>
        <Value name="missing" failure="true"/>
      </Var>
    </Input>

    <Input type="env">
      <VarSet name="file" when="fileName">
        <Var name="exists">
          <Value name="yes" property="fileExists"/>
          <Value name="no" failure="true"/>
        </Var>
        <VarSet name="contents" when="fileExists" whenNot="empty">
          <Var name="linesLongerThanPattern">
            <Value name="one" property="matchable" once="true"/>
            <Value name="many" property="matchable"/>
            <Value name="none" failure="true"/>
          </Var>
          <Var name="patterns" when="matchable" whenNot="empty">
            <Value name="none" once="true"/>
            <Value name="one" property="match"/>
            <Value name="many" property="match, many"/>
          </Var>
          <Var name="patternsInLine" when="match">
            <Value name="one"/>
            <Value name="many" once="true" when="many"/>
          </Var>
        </VarSet>
      </VarSet>
    </Input>

  </Function>
</System>
```

2.3 Test cases screenshot

3 Tcases-Input

3.1 Specification

System: Tcases

Function: run

Specification: This function is responsible for running test cases with various options.

Inputs: - defaultTupleSize: Specifies the default tuple size for test cases. - Yes: Indicates that a default tuple size is defined. - No: Indicates that no default tuple size is defined. - isNumber: Indicates whether the default tuple size is a number or not. Failure occurs if it's not a number.

- outFile: Specifies the output file option. - Yes: Indicates that an output file is defined. - No: Indicates that no output file is defined. - TransformOutputUndefined: Failure occurs if transformed

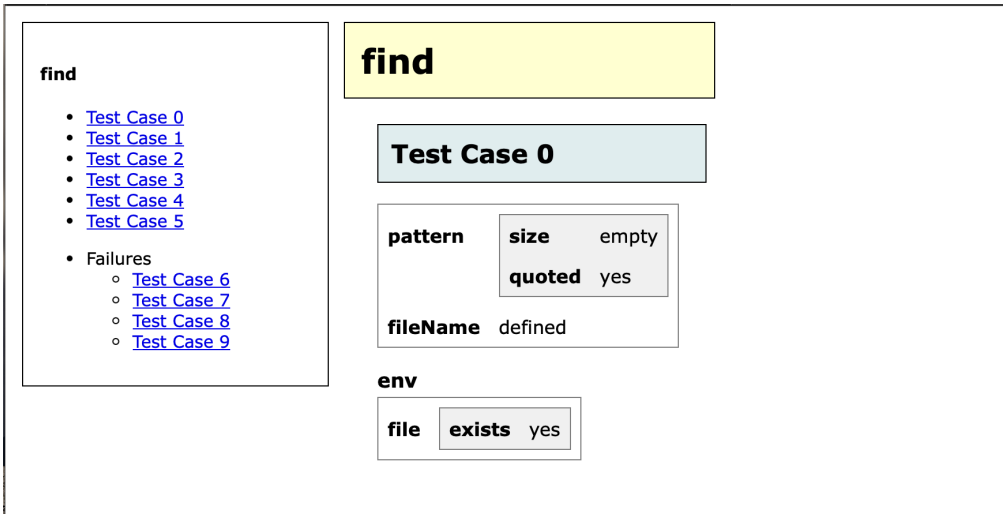

Figure 2: TCases Html View

output is undefined and the test file exists.

- genFile: Specifies the generated file option. - Yes: Indicates that a generated file is defined. - No: Indicates that no generated file is defined. - path: Specifies the path of the generated file. - isAbsolute: Indicates whether the path is absolute or not. - exists: Indicates whether the generated file exists or not.

- isJUnit: Specifies whether the output format is JUnit or not. - Yes: Indicates the output format is JUnit. It also marks transformed output if transform is not specified. - No: Indicates the output format is not JUnit if transform is specified. - NotAllowed: Failure occurs if transform is specified and isJUnit is not allowed.

- newTests: Specifies whether new tests are defined or not. - Yes: Indicates new tests are defined when the test file exists. - No: Indicates no new tests are defined.

- outDir: Specifies the output directory option. - Yes: Indicates that an output directory is defined. - No: Indicates that no output directory is defined. - path: Specifies the path of the output directory. - exists: Indicates whether the output directory exists or not. - isDirectory: Indicates whether the path is a directory or not. Failure occurs if it's not a directory.

- transformParams: Specifies the transformation parameters option. - count: Specifies the count of transformation parameters. - assignsValue: Indicates whether the transformation parameter assigns a value or not. Failure occurs if it doesn't assign a value. - nameDefined: Indicates whether the name of the transformation parameter is defined or not. Failure occurs if it's not defined. - valueDefined: Indicates whether the value of the transformation parameter is defined or not.

- seed: Specifies the seed option for generating random values. - Yes: Indicates that a seed is defined for generating random values. - No: Indicates that no seed is defined. - isNumber: Indicates whether the seed is a number or not. Failure occurs if it's not a number.

- testFile: Specifies the test file option. - Yes: Indicates that a test file is defined. - No: Indicates that no test file is defined. - path: Specifies the path of the test file. - isAbsolute: Indicates whether the path is absolute or not. - exists: Indicates whether the test file exists or not.

- transform: Specifies the transformation option. - Yes: Indicates that a transformation is defined. - No: Indicates that no transformation is defined. - path: Specifies the path of the transformation. - isAbsolute: Indicates whether the path is absolute or not. - exists: Indicates whether the transformation exists or not.

- inFile: Specifies the input definition file option. - Yes: Indicates that an input definition file is defined. - No: Indicates that no input definition file is defined. - path: Specifies the path of the input definition file. - isAbsolute: Indicates whether the path is absolute or not. - exists: Indicates whether the input definition file exists or not.

–¿

3.2 XML Code

```
<!--
    System: Tcases

    Function: run

    Specification:
    This function is responsible for running test cases with various options.

    Inputs:
    - defaultTupleSize: Specifies the default tuple size for test cases.
        - Yes: Indicates that a default tuple size is defined.
        - No: Indicates that no default tuple size is defined.
        - isNumber: Indicates whether the default tuple size is a number or not.
    Failure occurs if it's not a number.

    - outFile: Specifies the output file option.
        - Yes: Indicates that an output file is defined.
        - No: Indicates that no output file is defined.
        - TransformOutputUndefined: Failure occurs if transformed output is
    undefined and the test file exists.

    - genFile: Specifies the generated file option.
        - Yes: Indicates that a generated file is defined.
        - No: Indicates that no generated file is defined.
        - path: Specifies the path of the generated file.
        - isAbsolute: Indicates whether the path is absolute or not.
        - exists: Indicates whether the generated file exists or not.

    - isJUnit: Specifies whether the output format is JUnit or not.
        - Yes: Indicates the output format is JUnit. It also marks transformed
    output if transform is not specified.
        - No: Indicates the output format is not JUnit if transform is specified.
        - NotAllowed: Failure occurs if transform is specified and isJUnit is not
    allowed.

    - newTests: Specifies whether new tests are defined or not.
        - Yes: Indicates new tests are defined when the test file exists.
        - No: Indicates no new tests are defined.

    - outDir: Specifies the output directory option.
        - Yes: Indicates that an output directory is defined.
        - No: Indicates that no output directory is defined.
        - path: Specifies the path of the output directory.
        - exists: Indicates whether the output directory exists or not.
        - isDirectory: Indicates whether the path is a directory or not. Failure
    occurs if it's not a directory.

    - transformParams: Specifies the transformation parameters option.
        - count: Specifies the count of transformation parameters.
```

```
        - assignsValue: Indicates whether the transformation parameter assigns a value or not. Failure occurs if it doesn't assign a value.
        - nameDefined: Indicates whether the name of the transformation parameter is defined or not. Failure occurs if it's not defined.
        - valueDefined: Indicates whether the value of the transformation parameter is defined or not.

    - seed: Specifies the seed option for generating random values.
        - Yes: Indicates that a seed is defined for generating random values.
        - No: Indicates that no seed is defined.
        - isNumber: Indicates whether the seed is a number or not. Failure occurs if it's not a number.

    - testFile: Specifies the test file option.
        - Yes: Indicates that a test file is defined.
        - No: Indicates that no test file is defined.
        - path: Specifies the path of the test file.
        - isAbsolute: Indicates whether the path is absolute or not.
        - exists: Indicates whether the test file exists or not.

    - transform: Specifies the transformation option.
        - Yes: Indicates that a transformation is defined.
        - No: Indicates that no transformation is defined.
        - path: Specifies the path of the transformation.
        - isAbsolute: Indicates whether the path is absolute or not.
        - exists: Indicates whether the transformation exists or not.

    - inFile: Specifies the input definition file option.
        - Yes: Indicates that an input definition file is defined.
        - No: Indicates that no input definition file is defined.
        - path: Specifies the path of the input definition file.
        - isAbsolute: Indicates whether the path is absolute or not.
        - exists: Indicates whether the input definition file exists or not.

-->

<System name="Tcases">
  <Function name="run">
    <Input>

      <!-- Option: -c -->
      <VarSet name="defaultTupleSize">
        <Var name="defined">
          <Value name="Yes" property="defaultTupleSize"/>
          <Value name="No"/>
        </Var>

        <Var name="isNumber" when="defaultTupleSize">
          <Value name="Yes"/>
          <Value name="No" failure="true"/>
        </Var>
      </VarSet>

      <!-- Option: -f -->
      <VarSet name="outFile">
        <Var name="defined">
          <Value name="Yes" property="outFile"/>
          <Value name="No"/>
          <Value name="TransformOutputUndefined" failure="true" when="
```

```
    transformedOut, testFileExists"/>
      </Var>

      <VarSet name="path" when="outFile">
        <Var name="isAbsolute">
          <Value name="Yes"/>
          <Value name="No"/>
        </Var>
        <Var name="exists">
          <Value name="Yes" property="outFileExists"/>
          <Value name="No"/>
        </Var>
      </VarSet>
    </VarSet>

    <!-- Option: -g -->
    <VarSet name="genFile">
      <Var name="defined">
        <Value name="Yes" property="genFile"/>
        <Value name="No"/>
      </Var>

      <VarSet name="path" when="genFile">
        <Var name="isAbsolute">
          <Value name="Yes"/>
          <Value name="No"/>
        </Var>
        <Var name="exists">
          <Value name="Yes"/>
          <Value name="No" failure="true"/>
        </Var>
      </VarSet>

      <Var name="default" whenNot="genFile">
        <Value name="ForInputExists" when="inFile"/>
        <Value name="ForInputNone" when="inFile"/>
        <Value name="Standard" whenNot="inFile"/>
      </Var>
    </VarSet>

    <!-- Option: -J -->
    <Var name="isJUnit">
      <Value name="Yes" property="isJUnit, transformedOut" whenNot="transform"/>
      <Value name="No" when="transform"/>
      <Value name="NotAllowed" when="transform" failure="true"/>
    </Var>

    <!-- Option: -n -->
    <VarSet name="newTests">
      <Var name="defined" when="testFileExists">
        <Value name="Yes"/>
        <Value name="No"/>
      </Var>
    </VarSet>

    <!-- Option: -o -->
    <VarSet name="outDir">
      <Var name="defined">
        <Value name="Yes" property="outDir"/>
```

```
          <Value name="No"/>
        </Var>

        <VarSet name="path" when="outDir">
          <Var name="exists">
            <Value name="Yes" property="outDirExists"/>
            <Value name="No"/>
          </Var>
          <Var name="isDirectory" when="outDirExists">
            <Value name="Yes"/>
            <Value name="No" failure="true"/>
          </Var>
        </VarSet>
      </VarSet>

      <!-- Option: -p -->
      <VarSet name="transformParams">
        <When>
          <AnyOf property="transform, isJUnit"/>
        </When>
        <Var name="count">
          <Value name="One" property="params"/>
          <Value name="Many" property="params"/>
          <Value name="None"/>
        </Var>
        <Var name="assignsValue" when="params">
          <Value name="Yes"/>
          <Value name="No" failure="true"/>
        </Var>
        <Var name="nameDefined" when="params">
          <Value name="Yes"/>
          <Value name="No" failure="true"/>
        </Var>
        <Var name="valueDefined" when="params">
          <Value name="Yes"/>
          <Value name="No"/>
        </Var>
      </VarSet>

      <!-- Option: -r -->
      <VarSet name="seed">
        <Var name="defined">
          <Value name="Yes" property="random"/>
          <Value name="No"/>
        </Var>

        <Var name="isNumber" when="random">
          <Value name="Yes"/>
          <Value name="No" failure="true"/>
        </Var>
      </VarSet>

      <!-- Option: -t -->
      <VarSet name="testFile">
        <Var name="defined">
          <Value name="Yes" property="testFile"/>
          <Value name="No"/>
        </Var>

```

```
        <VarSet name="path" when="testFile">
          <Var name="isAbsolute">
            <Value name="Yes"/>
            <Value name="No"/>
          </Var>
          <Var name="exists">
            <Value name="Yes" property="testFileExists"/>
            <Value name="No"/>
          </Var>
        </VarSet>

        <VarSet name="default" whenNot="testFile">
          <Var name="exists">
            <Value name="Yes" property="testFileExists"/>
            <Value name="No"/>
          </Var>
        </VarSet>
      </VarSet>

      <!-- Option: -x -->
      <VarSet name="transform">
        <Var name="defined">
          <Value name="Yes" property="transform, transformedOut"/>
          <Value name="No"/>
        </Var>
        <VarSet name="path" when="transform">
          <Var name="isAbsolute">
            <Value name="Yes"/>
            <Value name="No"/>
          </Var>
          <Var name="exists">
            <Value name="Yes"/>
            <Value name="No" failure="true"/>
          </Var>
        </VarSet>
      </VarSet>

      <!-- Input definition file -->
      <VarSet name="inFile">
        <Var name="defined">
          <Value name="Yes" property="inFile"/>
          <Value name="No"/>
        </Var>
        <VarSet name="path" when="inFile">
          <Var name="isAbsolute">
            <Value name="Yes"/>
            <Value name="No"/>
          </Var>
          <Var name="exists">
            <Value name="asDefined"/>
            <Value name="withInputXml"/>
            <Value name="withXml"/>
            <Value name="No" failure="true"/>
          </Var>
        </VarSet>
      </VarSet>

    </Input>
  </Function>
```

```
</System>
```

3.3 Test cases screenshot

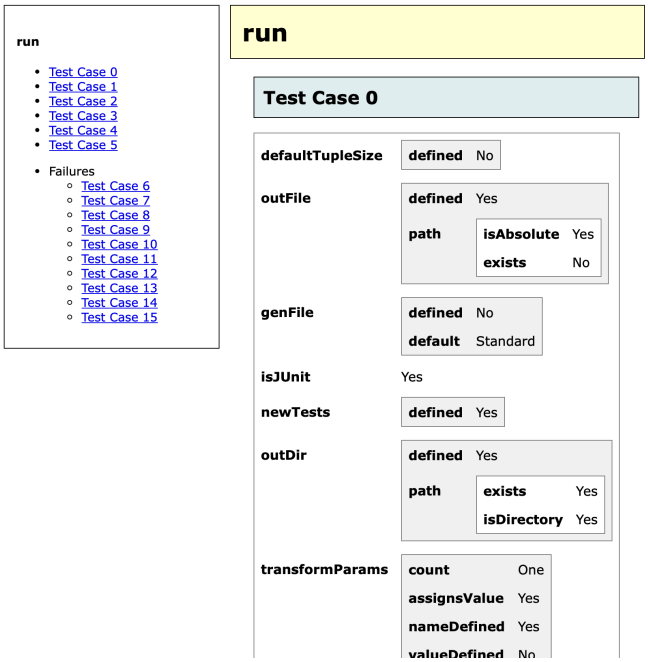

Figure 3: TCases Html View

4 Ice-Cream Input

4.1 Specification

Usage: Checking different Scoop level toppings of ice-cream We can add different levels of scoops in different condition of cone functions for example 1) If cone is not there then we will not add any scoop of ice-cream. 2) If cone is there and customer want Plain ice-cream then we can add maximum of 1 scoop and 1 topping in ice-cream. 3) If cone is there and customer want Plenty ice-cream then we can add min of 1 and maximum of 2 scoop and 2 topping in ice-cream. 4) If cone is there and customer want Grande ice-cream then we can add maximum of 4 scoop and 3 maximum and 1 minimum topping in ice-cream. 5) If cone is there and customer want Too-much ice-cream then we can add maximum of 3 scoop and 4 minimum topping in ice-cream. 6) If cone is there and customer want Too-much ice-cream then we can add maximum of 4 scoop and 5 minimum topping in ice-cream.

4.2 XML Code

```
<System name="Ice-Cream">
    <Function name="Cones">
    <!--
    Usage: Checking different Scoop level & toppings of ice-cream
    We can add different levels of scoops in different condition of cone functions
     for example
```

```
    1) If cone is not there then we will not add any scoop of ice-cream.
    2) If cone is there and customer want Plain ice-cream then we can add maximum of 1 scoop and 1 topping in ice-cream.
    3) If cone is there and customer want Plenty ice-cream then we can add min of 1 and maximum of 2 scoop and 2 topping in ice-cream.
    4) If cone is there and customer want Grande ice-cream then we can add maximum of 4 scoop and 3 maximum and 1 minimum topping in ice-cream.
    5) If cone is there and customer want Too-much ice-cream then we can add maximum of 3 scoop and 4 minimum topping in ice-cream.
    6) If cone is there and customer want Too-much ice-cream then we can add maximum of 4 scoop and 5 minimum topping in ice-cream.
-->

        <Input>
            <Var name="Cone">
                <Value name="Empty" failure="true">
                    <When>
                        <LessThan property="scoop" max="1"/>
                    </When>
                </Value>
                <Value name="Plain">
                    <When>
                        <AllOf>
                            <Equals property="scoop" count="1"/>
                            <NotMoreThan property="topping" max="1"/>
                        </AllOf>
                    </When>
                </Value>
                <Value name="Plenty">
                    <When>
                        <AllOf>
                            <Between property="scoop" min="1" max="2"/>
                            <NotMoreThan property="topping" max="2"/>
                        </AllOf>
                    </When>
                </Value>
                <Value name="Grande">
                    <When>
                        <AllOf>
                            <Between property="scoop" exclusiveMin="0" exclusiveMax="4"/>
                            <Between property="topping" min="1" max="3"/>
                        </AllOf>
                    </When>
                </Value>
                <Value name="Too-Much" failure="true">
                    <When>
                        <AnyOf>
                            <MoreThan property="scoop" min="3"/>
                            <NotLessThan property="topping" min="4"/>
                        </AnyOf>
                    </When>
                </Value>
            </Var>

            <VarSet name="Flavors">
                <Var name="Vanilla">
                    <Value name="Yes" property="scoop"/>
                    <Value name="No"/>
```

```
                </Var>
                <Var name="Chocolate">
                    <Value name="Yes" property="scoop"/>
                    <Value name="No"/>
                </Var>
                <Var name="Strawberry">
                    <Value name="Yes" property="scoop"/>
                    <Value name="No"/>
                </Var>
                <Var name="Pistachio">
                    <Value name="Yes" property="scoop"/>
                    <Value name="No"/>
                </Var>
                <Var name="Lemon">
                    <Value name="Yes" property="scoop"/>
                    <Value name="No"/>
                </Var>
                <Var name="Coffee">
                    <Value name="Yes" property="scoop"/>
                    <Value name="No"/>
                </Var>
            </VarSet>

            <VarSet name="Toppings" when="scoop">
                <Var name="Sprinkles">
                    <Value name="Yes" property="topping"/>
                    <Value name="No"/>
                </Var>
                <Var name="Pecans">
                    <Value name="Yes" property="topping"/>
                    <Value name="No"/>
                </Var>
                <Var name="Oreos">
                    <Value name="Yes" property="topping"/>
                    <Value name="No"/>
                </Var>
                <Var name="Cherries">
                    <Value name="Yes" property="topping"/>
                    <Value name="No"/>
                </Var>
                <Var name="MMs">
                    <Value name="Yes" property="topping"/>
                    <Value name="No"/>
                </Var>
                <Var name="Peppermint">
                    <Value name="Yes" property="topping"/>
                    <Value name="No"/>
                </Var>
            </VarSet>
        </Input>
    </Function>
</System>
```

4.3 Test cases screenshot

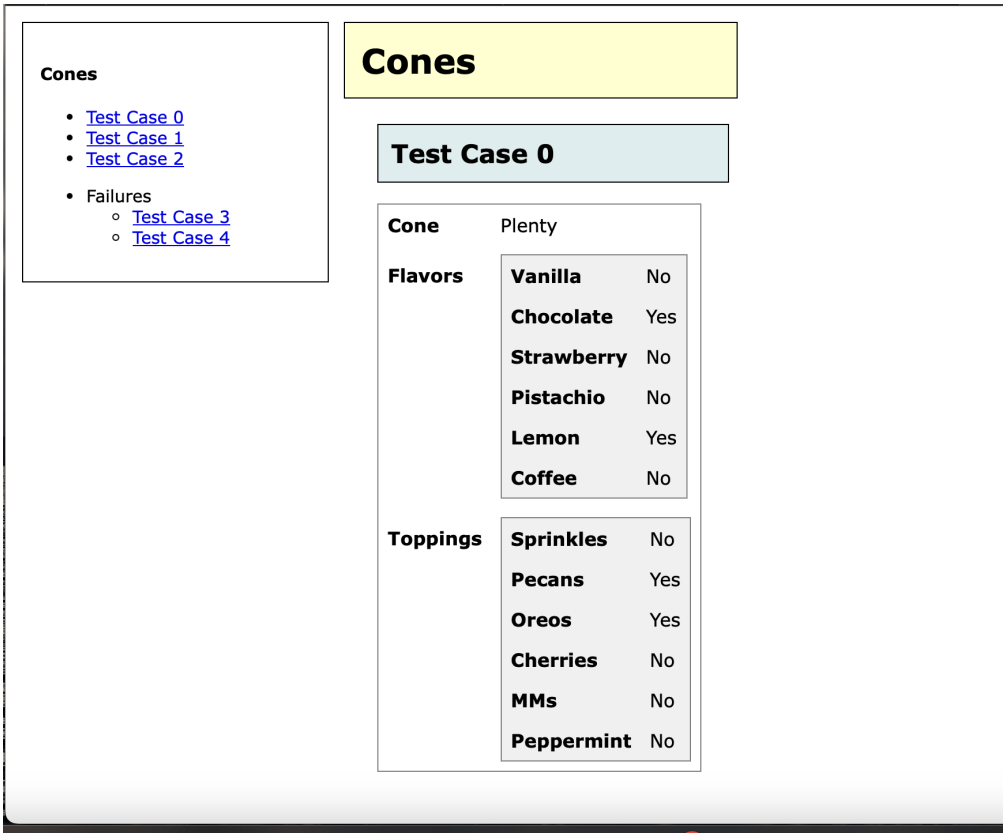

Figure 4: TCases Html View