Test Case Generator

Review - 3

List of Contributors

Name	Initials	Organization	Email
Ayush Vipul Mehrotra(19BCE2393)	AVM	VIT	ayushvipul.mehrotra2019@vitstudent.ac.in
Harshvardhan Singh Gahlaut (19BCE2372)	HSG	VIT	harshvardhan.singh2019a@vitstudent.ac.in
Dharmik Jayeshbhai Govani (19BCE0091)	DJG	VIT	dharmik.jayeshbhai2019@vitstudent.ac.in

Executive Summary

This document represents the final Report for the Test Case generator project by this team of VITians. The document begins with an Introduction section that describes the purpose of the document and what is considered to be in the scope of this document as well as what is outside the scope of this document.

The next section is a Technical Specification of the Project. This section includes the overall features of the project along with other requirements and Attributes defining the technical aspects of the project

The Design Approach and details section outlies the approach and methods used during the design of the software. Furthermore, it discusses the code that was used and the also outlays the constraints alternatives and tradeoffs of the software

The Schedule for the project along with the task distribution and breakdown of the work structure has been mentioned. This is followed by a Demonstration of the software where you will find links to the GitHub repository and the software webpage.

Finally, we end off by discussing the outcome of the Software Development and the final software itself.

Our Test Case Generator Deals with the basic use of a test case at a time when the test cases required are pertaining different variants of random integers, arrays, trees, graphs and strings. A basic design of choosing a data type and giving the required constraints and clicking a button and voila you have a usable test case which you can save and log as a .txt file in case of a big project.

Table of Contents

List of Figures

List of Tables

1 Definitions, Acronyms, and Abbreviations

Table of Definitions, Acronyms, and Abbreviations

Definition, Acronym, or Abbreviation	Description
SRS	Software Requirements Specification.
SMTP	Simple Mail Transport Protocol
POP3	Post Office Protocol 3

2 References

Table of References

References	Description
Software Development Plan	The Software Development Plan from the Electronic Stamp project was referenced.
Software Requirement Specification	The specifications and requirements were mentioned
Software Design Specifications	The design and data flow specifications have been mentioned
Prototype GUI	From This document the design of the application has been mentioned
Test Report	The results of the Testing phase have been mentioned

3 Introduction

3.1 Objective

For this software, our Objective as a team is to provide test cases for specific requirement of different data types. Our approach is going to be more towards a domestic/ small scale testing rather than industrial testing so as to act as a helpful tool for the up-and-coming coders as well as many others to use this software so as to simulate their basic testing needs when it comes to the matter of data types.

3.2 Motivation

Now days a lot of people irrespective of their age, are trying to learn more about the internal operations of a computer especially, during this pandemic situation. From beginners to high level testers, generating test cases to test your software is very tiresome, as for experienced coders, it takes up too much of their time whereas, novice coders might not be aware of the correct parameters for testing their code.
Test generation is the process of creating a set of test data or test cases, for testing the adequacy of new or revised software applications. Test generation is seen to be a complex problem and though a lot of solutions have come forth, most of them are limited to toy programs. Test generation is one aspect of software testing. Since testing is labor-intensive, accounting for nearly 1/3rd of the cost of system development, the problem of generating quality test data quickly, efficiently and, accurately, is seen to be important.
A test data generator follows the following steps-
• Program control flow graph construction.
• Path selection.
• Generating test data.
The basis of the generator is simple. The path selector identifies the path. Once a set of test paths is determined, the test generator derives input data for every path that results in the execution of the selected path.
Essentially, our objective is to find a data set that will traverse the path chosen by the path selector. The solution will ideally be a system of equations which will describe the nature of input data so as to traverse the path.

3.3 Background

Testing is a vital part of software engineering. However, when it comes to small scale coding and learning how to code a lot of people find it difficult to create and run test cases on their software especially when it comes to data types. Test case generators generally are pretty basic and aim to test the code in a very vast dimension. Our test case generator goes a bit further and allows our coders to use many ways such a basic copy paste or it gives them a method to create and integrate a software system with a testing feature via a .txt file which to a novice coder also gives insight into the field of data sharing.

4 Project Descriptions and Goals

This project is a basic data type test case generator ranging between the following

- Random Integers:
  This allows the user to produce random integers within a given range of numbers as per their choice.
- Strings:
  This allows the user to produce a number of strings with a specific number of characters and the user can choose if they would want the strings to be made out of specific characters.
- Arrays:
  The user has a choice to create a single array or a matrix of array to simulate inputting an entire array or a list of arrays
- Trees:
  Unweighted or weighted trees which can be made as per their requirements.
- Graphs
  Just as Trees graphs can also be made by requirement as a weighted directed, unweighted undirected or a weighted undirected graph.

Our goal is to create a test case generator for the masses and younger generation to use with a diverse range of specifications that can be catered to every data type use.

5 Technical Specification

5.1 Product Perspective

The Test Case Software that is to be developed by VIT is not a complete testing system itself. The Test Case Generator Software and its requirements are only pertaining to the functionality needed to generate the required data types for specific cases.

Since the Test Case Software is not an Interactive Development Environment (IDE) by itself, the software is being developed as a web extension of a previously implemented IDE.

It is to be developed as an Open-Source Website where the user will input what kind of data type they require and the constraints of their code.

5.2 Product Functions

The follow is a table of the requirements that the system SHALL meet. The list of requirements was produced from the initial project documentation provided by the requirements expert.

Table of Shall Requirements

ID	Origin	Shall Requirement
10	*Project Description Document*	The User SHALL be able to choose the data type from the available choices.
11	*Project Description Document*	The User SHALL be able to input constraints to the requested data type according to the constraints of the code.
12	*Project Description Document*	The User Shall be able to make changes to the constraints at any given time.
13	*Project Description Document*	The User SHALL be able to copy and paste the generated test cases in their respective places.
14	*Project Description Document*	The user SHALL be able to download the generated test cases in the form of a .txt file.

5.3 Specific Requirements

5.3.1 System Features

5.3.1.1 SELECTING DATA TYPE

5.3.1.1.1 Introduction

The test case generator shall allow the user to select the datatype for which the test cases are required.

5.3.1.1.2 Functional Requirements

Purpose: Choosing the appropriate datatype

Input: Clicking on the data type which is to be used

Processing: If there are subparts to the data type they will be selected.

Output: The constraint page of that particular data type is made visible to the user

5.3.1.1.3 Stimulus Response

A) User Selects datatype:

User Actions	System Actions
(1) Click on required data type	
	(2) Slides the title revealing the sub types in the data type
(3) Click on Subtype	
	(4) System directs the user to the constraints window

5.3.1.2 INPUTING CONSTRAINTS

5.3.1.2.1 Introduction

In this the user inputs the constraints according to their requirements

5.3.1.2.2 Functional Requirements

Purpose: giving constraints to the code

Input: maximum value, number of test cases, minimum value, etc.

Processing: System verifies the validity of the constraints returns error if wrong

Output: Calculates and generates random data type test cases according to the given constraints

5.3.1.2.3 Stimulus Response

User Actions	System Actions
(1) input constraints	
	(2) Check if constraints are valid.
	(3) Return error if exists
	(4) Generate random test case according to constraints if not.

	(5) Display Output

5.3.1.3 DOWNLOADING TEST CASES

5.3.1.3.1 Introduction

This feature allows the user to download the generated test cases as a .txt file.

5.3.1.3.2 Functional Requirements

Purpose: For user to keep for further reference.

Input: Click download button.

Processing: If Download button is clicked generates a .txt file for user

Output: .txt file with the test cases for user to save is generated

5.3.1.3.3 Stimulus Response

User Actions	**System Actions**
(1) Click on download button	
	(2) System takes the output and saves it in a .txt file
	(3) System asks user to give name to the file
(4) enters name for file	
	(5) Saves file on computer

5.4 Performance Requirements

The following tables list the performance requirements of the Test Case Generator.

Table of Performance Requirements

Performance Requirement	**Description**
Test case Capacity	The test case capacity of the test case generator will be configurable to the extent that the hardware permits.
Software Runtime Errors	The Test Case Generator will handle the runtime errors consistently and as gracefully as possible.

5.5 Software System Attributes

5.5.1 Reliability

Reliability in the Test Case Generator will be ensured by thorough unit, milestone, and release testing. Comprehensive test scenarios and acceptance criteria will be established to reflect the necessary level reliability required of the Test Case Generator. The all delivered source code will

be thoroughly tested using the established test scenarios until the acceptance criteria are satisfied by the Test Case Generator

5.5.2 Portability

The Test Case generator software will also be written in JavaScript and gain the portability provided by that language.

It is safe to say that the implementation of the Test Case Generator will be able to be ported to other system platforms that accept JavaScript applications with little to no changes required. It is not safe to say that the Test Case Generator will execute properly on the other system platforms with little or no change. Significant changes to the Test Case Generator will not be required to ensure proper execution on other system platforms as long as they have a viable version of chrome.

5.6 Hardware and Software Specifications

To access our software the user must have the following available to them:

- A computer with a working browser that can access GitHub.
- An Operating System which supports the latest versions of all JavaScript Enabled Browsers.

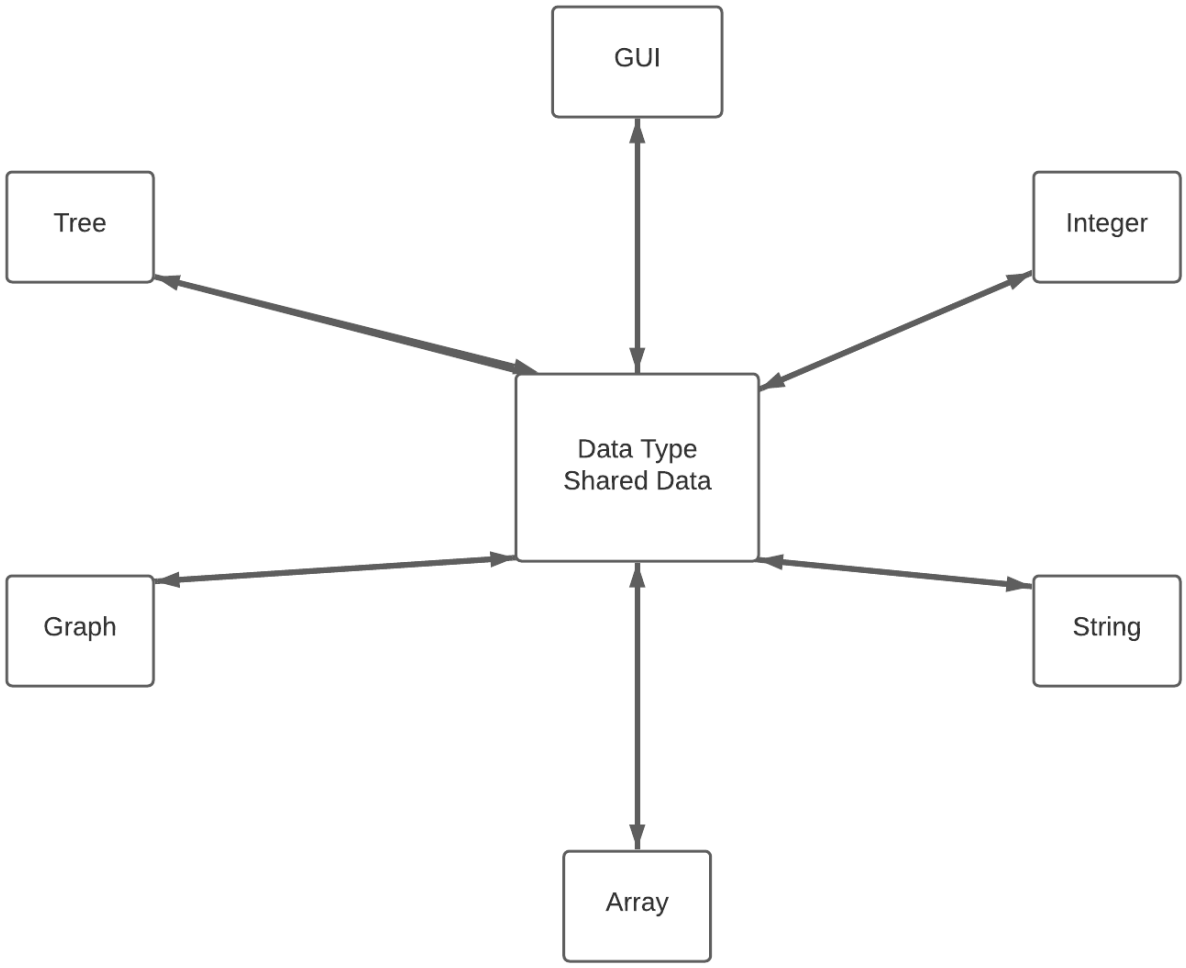
6 Design Approach and Details (as applicable)

6.1 Design Approach/ Materials and Methods

6.1.1 ARCHITECTURE MODEL: REPOSITORY MODEL

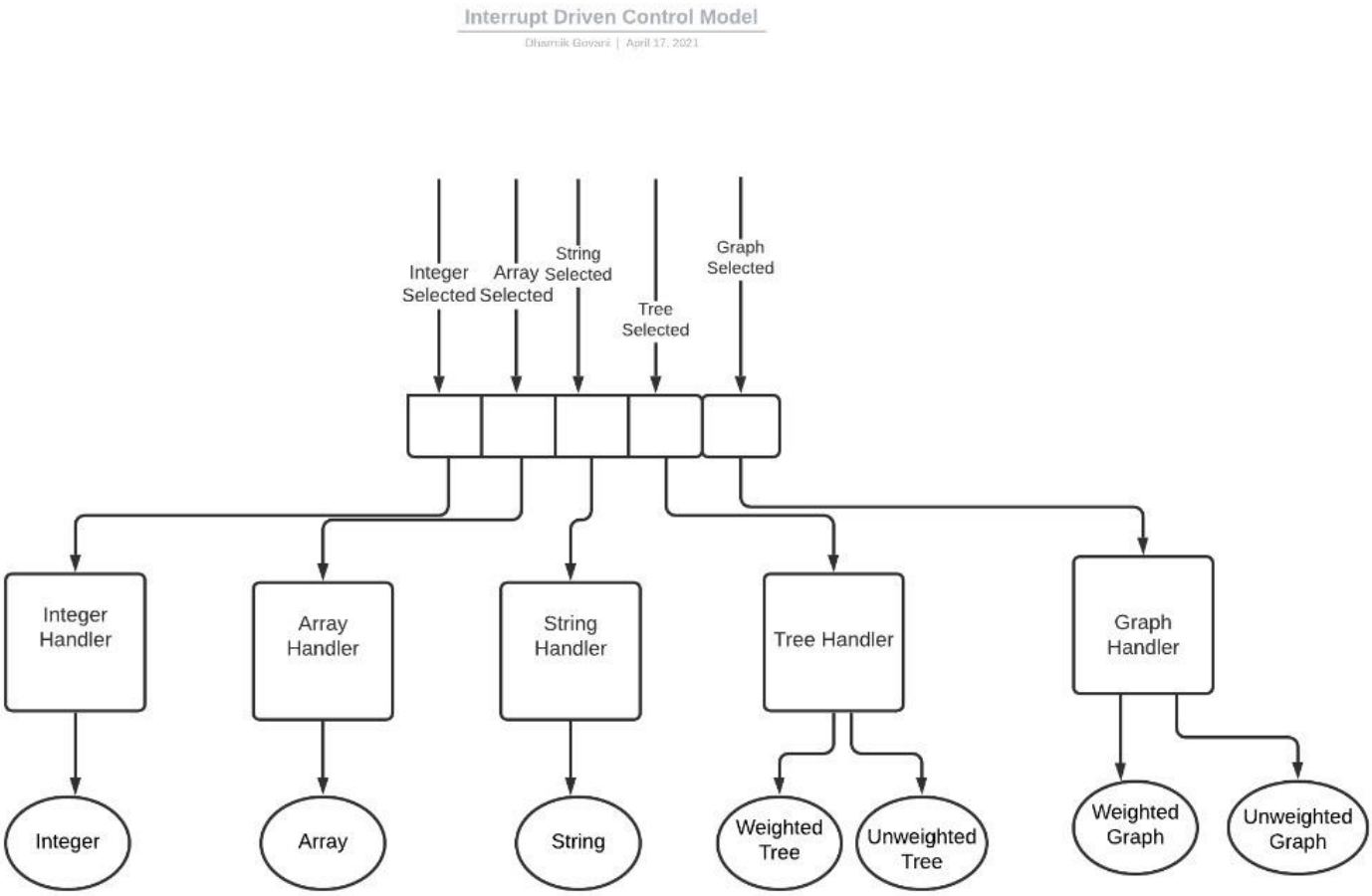
We chose the Repository model as it was best suited for our project Our project called for a shared platform for the data to share between the different modules and for data to be exchanged between the GUI and the modules in turn connecting the user with the modules.

Figure 1 Repository Model



6.1.2 CONTROL MODEL: INTERRUPT DRIVEN MODEL

We chose the Interrupt driven model as in our design according to the interrupt the access was given to different modules.



6.1.3 Module Decomposition

The E-Mail Client Software has been decomposed into the following modules.

- Integer Generator Module: This module collects data from the user to be used to generate a random integer according to the given constraints.
- Array Generator Module: This module collects data from the user to be used to generate a random array according to the given constraints.
- String Generator Module: This module collects data from the user to be used to generate a random string according to the given constraints.
- Tree Generator Module: This module collects data from the user to be used to generate a random tree according to the given constraints.
- Graph Generator Module: This module collects data from the user to be used to generate a random graph according to the given constraints.
- Data type selection Module: This module selects the type of data type whose test cases are required.

6.1.4 Concurrent Process Decomposition

The Test case generator Project consists of two major components, the user and the system. This team shall design the system and the user.

A complete view of the project suggests that there are two processes, the user process and the system process. The system process communicates with the system to generate required data types. The user process communicates with the user to get the constraints and type of data type required. These two processes run concurrently and only exchange information when the system process requires data to generate test cases from the user.

6.1.5 Data Decomposition

The following are the two major data components, the user Input and the System Output.

User Input: This is a database that contains the following data items;
Data type,

- Data Type: Gives the type of data type
- Subtype: Gives subtype if it exists
- Constraints: Gives constraints based on data type

The Electronic Stamp: This is a data structure that is attached with the outgoing email. It contains the following data.

- Output: Terminal that gives us the output based on the given constraints
- Download Button: Downloads the test cases as a .txt file

6.1.6 Dependency Description

6.1.6.1 Inter-module Dependencies

6.1.6.2 Dependent Modules

The following modules are dependent on one another for their functioning.

- Integer Generator Module: This module collects data from the Data type selection Module to be used to generate a random integer according to the given constraints.
- Array Generator Module: This module collects data from the Data type selection Module to be used to generate a random array according to the given constraints.
- String Generator Module: This module collects data from the Data type selection Module to be used to generate a random string according to the given constraints.
- Tree Generator Module: This module collects data from the Data type selection Module to be used to generate a random tree according to the given constraints.
- Graph Generator Module: This module collects data from the Data type selection Module to be used to generate a random graph according to the given constraints.
- Data type selection Module: This module selects the type of data type whose test cases are required.

6.1.7 Inter-process Dependencies

As described earlier the two main processes are the user process and the system process. The system process depends on the user process for obtaining data from the user. This is the only dependency between the two processes. Please reference Appendix A for a full class diagram.

6.1.8 Data Dependencies

The following Data Flow Diagram shows the data dependencies between the various entities and modules.

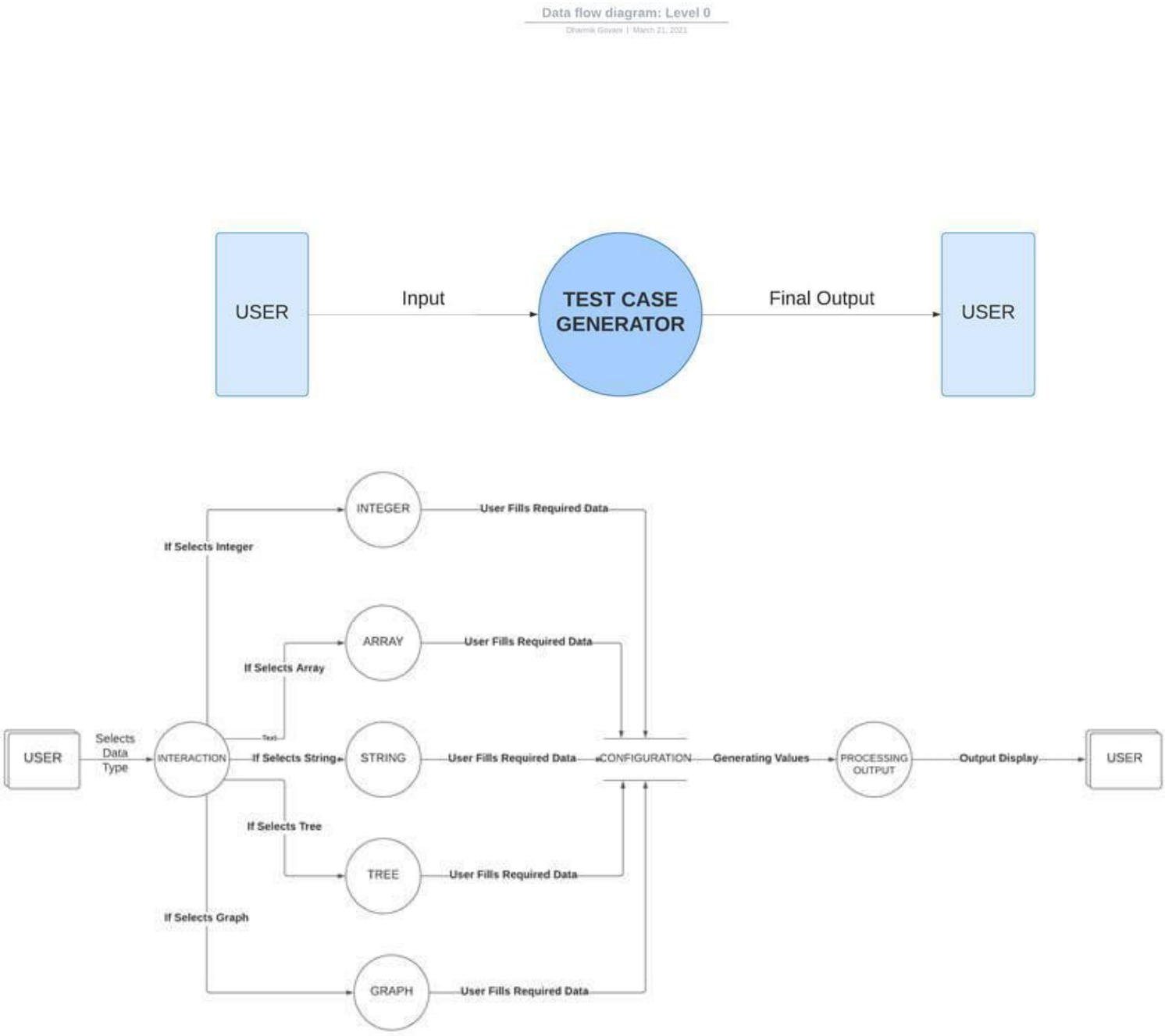
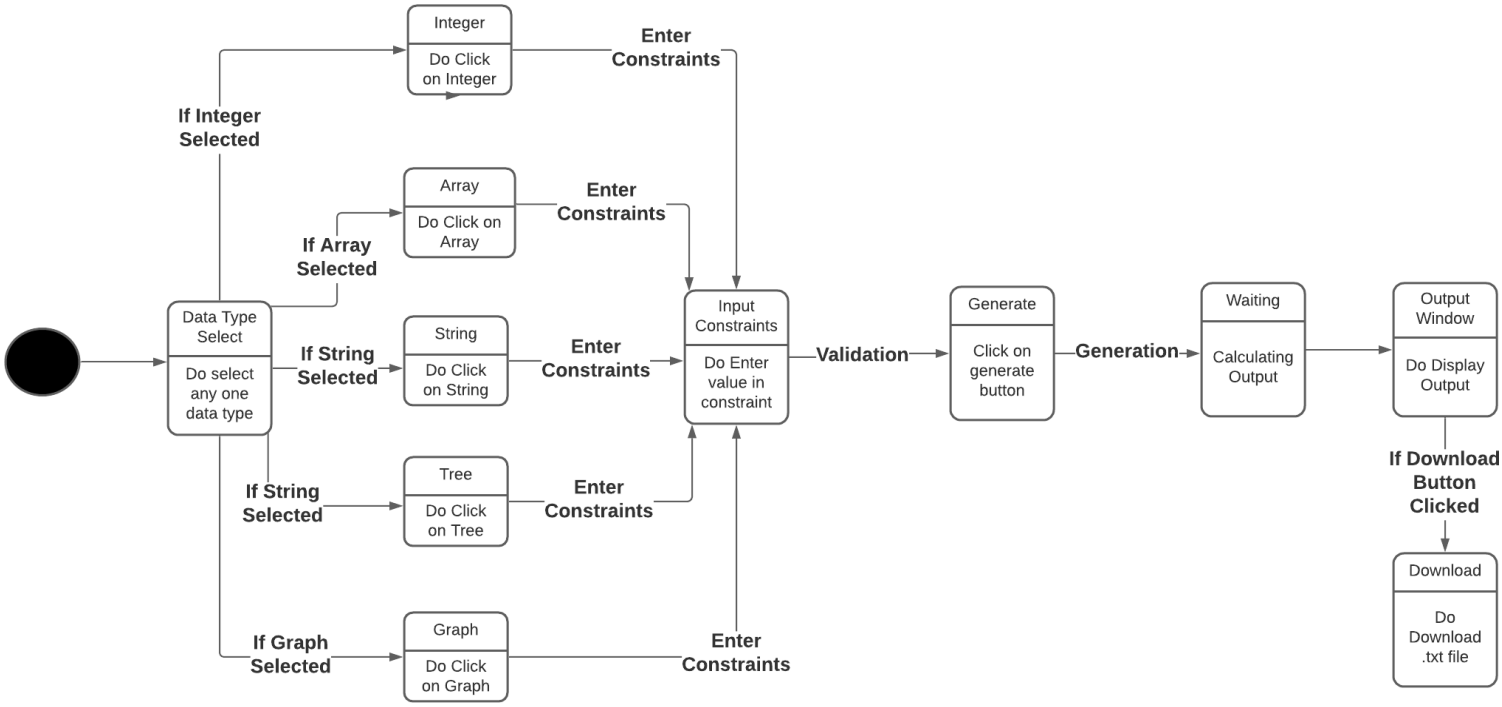


Figure 2, Data Flow Diagram

6.1.9 State Transitions

The state transition diagram shows the state transitions when a user chooses a type of data type to be generated as test cases, inputs the constraints and everything required for the generation of said test cases followed by an option to either copy and forget or save the test cases.

Figure 3, State Transition Diagram



6.1.10 Interface Description

6.1.10.1 Module Interface

6.1.10.1.1 Data Type Selection Module Description

6.1.10.1.1.1 User Interface Design

Figure 2, Data type selection module UI

6.1.10.1.1.2 Description

The Data Type selection module is the first page displayed to the user having links in the form of buttons to the different available modules and also shortcuts to weighted and unweighted options

6.1.10.1.2 Integer Module Description

6.1.10.1.2.1 User Interface Design

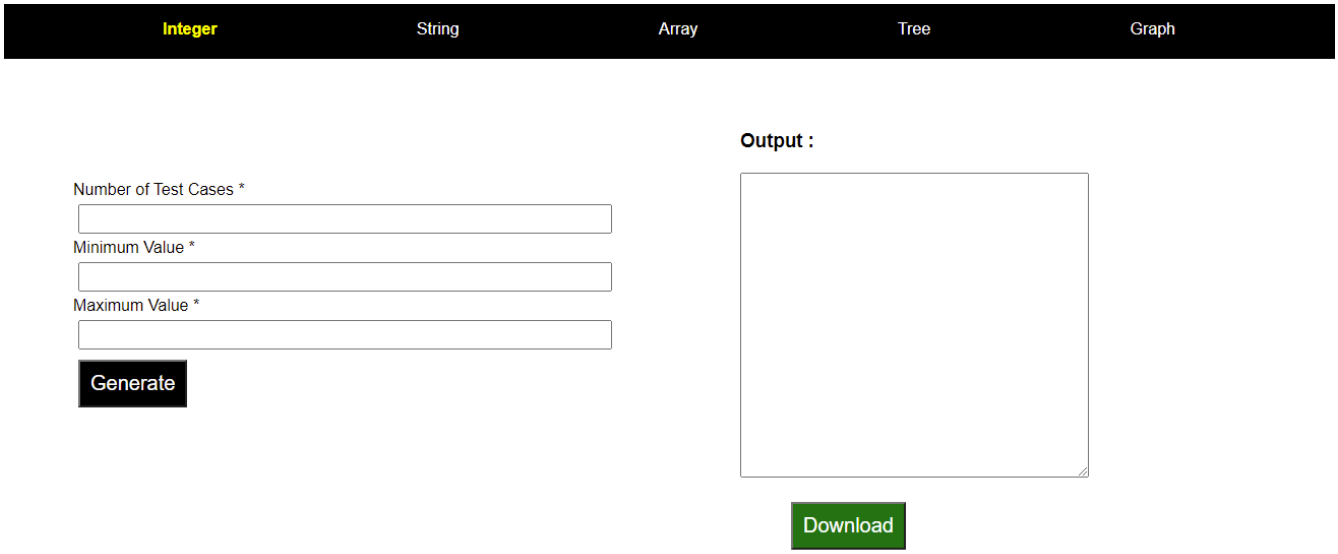


Figure 3, Integer Module UI

6.1.10.1.2.2 Description

The following figure the different constraints required by the Integer module. It also shows the download and output window.

6.1.10.1.3 Array Module Description

6.1.10.1.3.1 User Interface Design

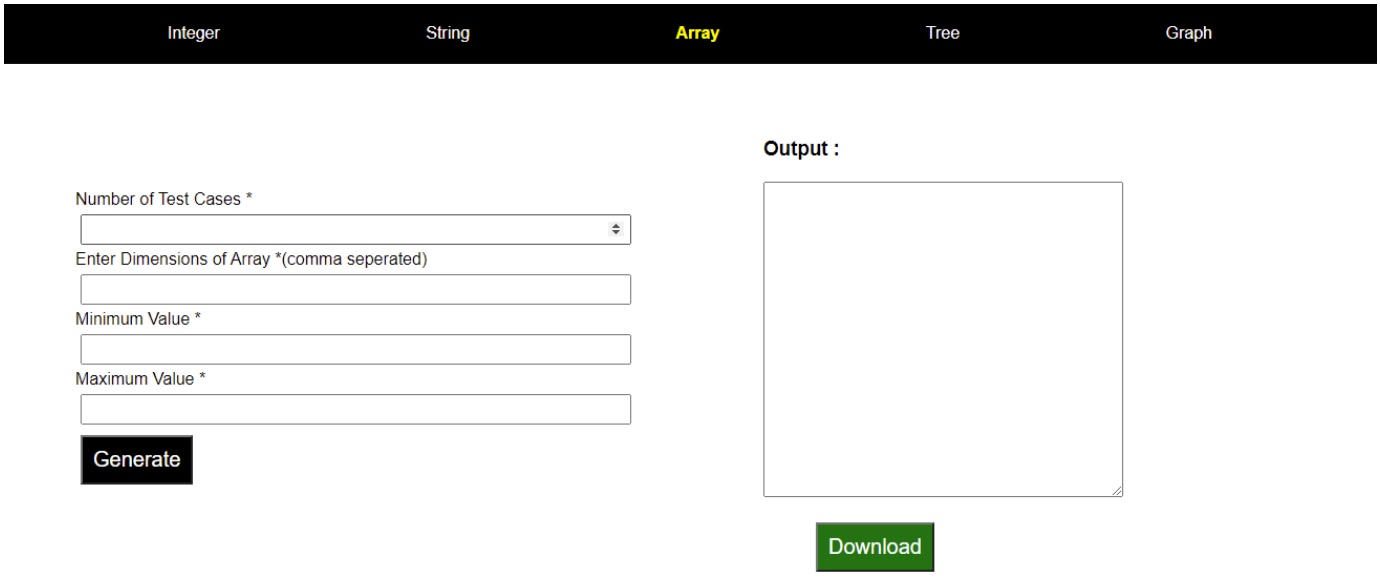


Figure 4, Array Module UI

6.1.10.1.3.2 Description

The following figure the different constraints required by the Array module. It also shows the download and output window.

6.1.10.1.4 String Module Description

6.1.10.1.4.1 User Interface Design

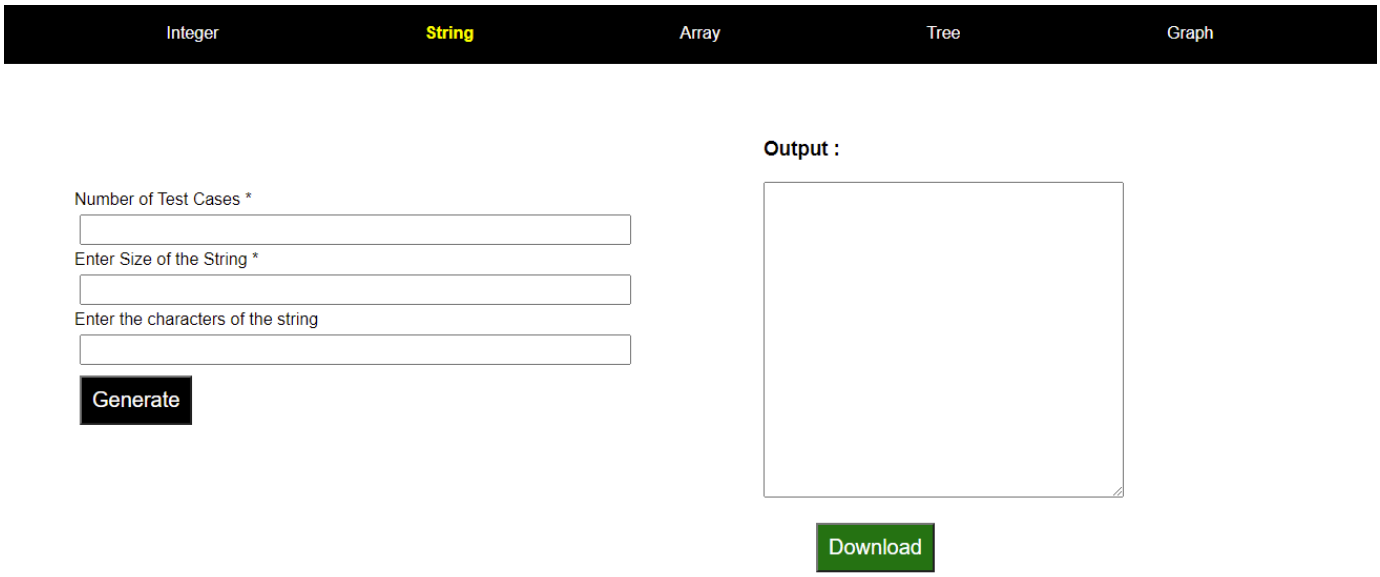


Figure 5, String Module UI

6.1.10.1.4.2 Description

The following figure the different constraints required by the String module. It also shows the download and output window.

6.1.10.1.5 Graph Module Description

6.1.10.1.5.1 User Interface Design

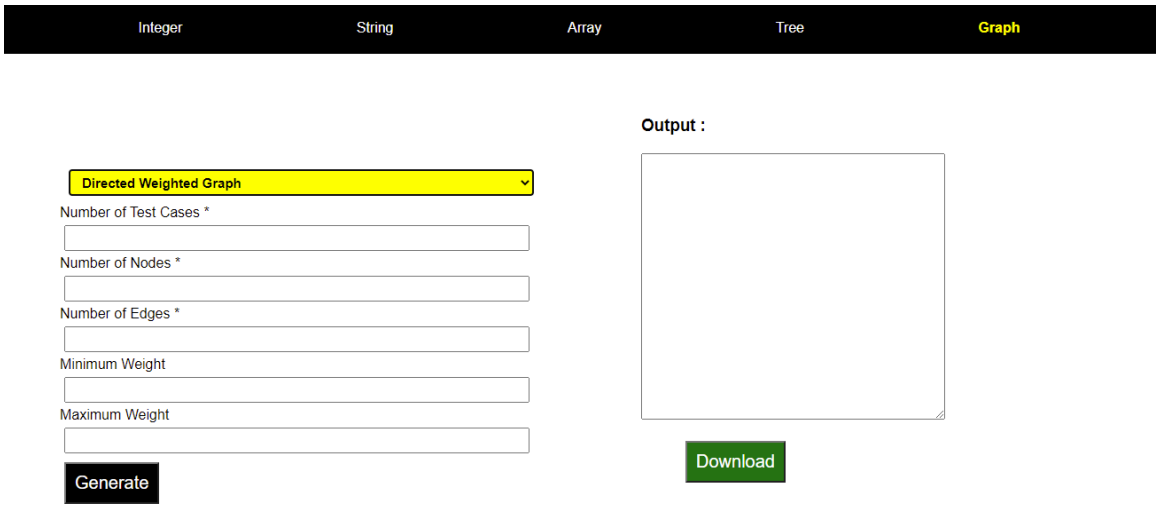


Figure 6, Weighted Graph UI

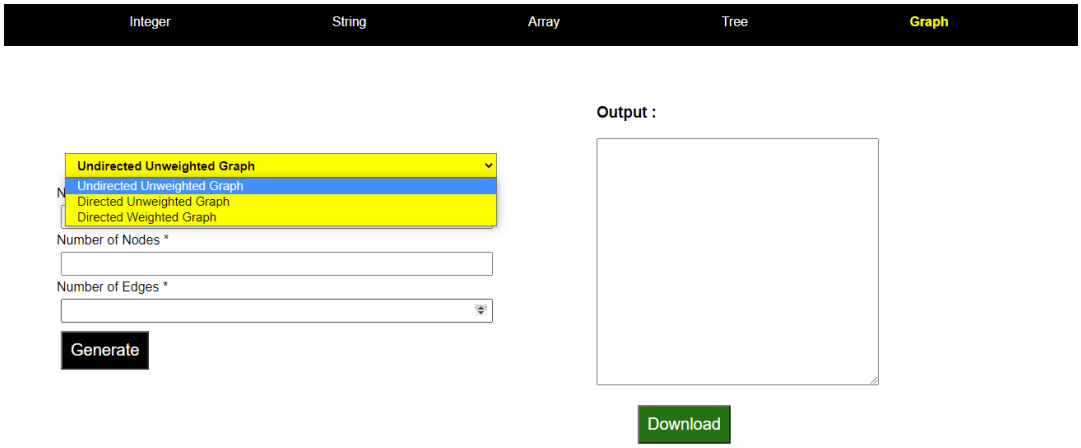


Figure 7, Unweighted Graph UI

6.1.10.1.5.2 Description

The following figure the different constraints required by the Graph module depending on whether it is a weighted or unweighted graph. It also shows the download and output window.

6.1.10.1.6 Tree Module Description

6.1.10.1.6.1 User Interface Design

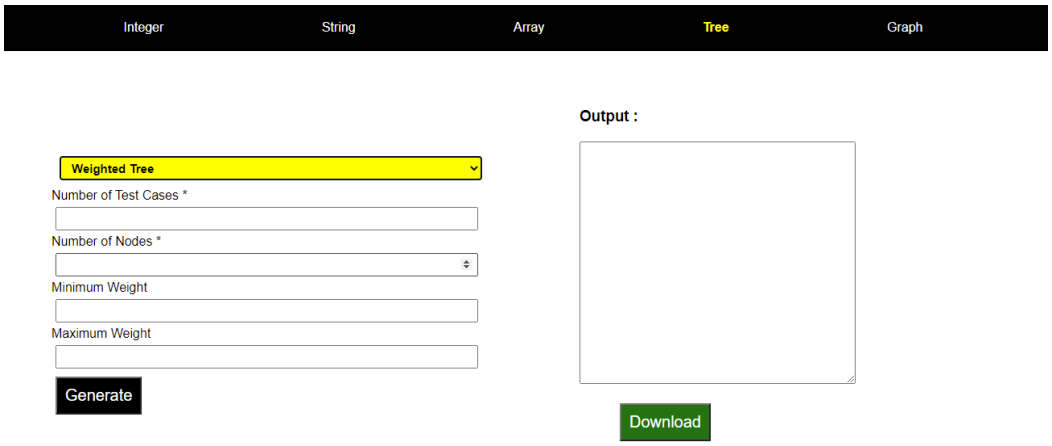


Figure 8, Weighted Tree Module UI

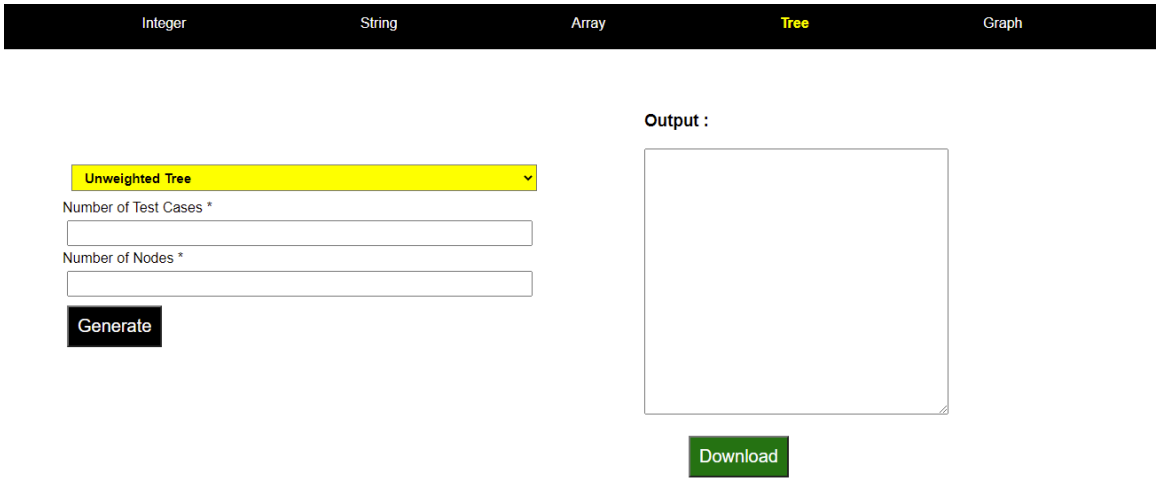


Figure 9, Unweighted Tree Module UI

6.1.10.1.6.2 Description

The following figure the different constraints required by the Tree module depending on whether it is a weighted or unweighted tree. It also shows the download and output window.

6.1.11 Process Interface

6.1.11.1 Datatype selection Process Description

The primary objective of this module is to obtain The type of data type and guide the user to that particular page.

6.1.11.2 Integer Process Description

The Integer Module Interacts with the user to take in constraints such as no of test cases, maximum value and minimum value to convey it to the system. The system then uses the constraints to give out test cases and displays them on the output window. Thereafter the user has a choice to either copy it using ctrl c or download it in a .txt file.

6.1.11.3 Array Process Description

The Array Module Interacts with the user to take in constraints such as no of test cases, maximum value and minimum value to convey it to the system. The system then uses the constraints to give out test cases and displays them on the output window. Thereafter the user has a choice to either copy it using ctrl c or download it in a .txt file.

6.1.11.4 String Process Description

The String Module Interacts with the user to take in constraints such as no of test cases, maximum value and minimum value to convey it to the system. The system then uses the constraints to give out test cases and displays them on the output window. Thereafter the user has a choice to either copy it using ctrl c or download it in a .txt file.

6.1.11.5 Graph Process Description

The Graph Module Interacts with the user to take in constraints such as no of test cases, maximum value and minimum value depending on whether it is a weighted or unweighted graph to convey it to the system. The system then uses the constraints to give out test cases and displays them on the output window. Thereafter the user has a choice to either copy it using ctrl c or download it in a .txt file.

6.1.11.6 Tree Process Description

The Tree Module Interacts with the user to take in constraints such as no of test cases, maximum value and minimum value depending on whether it is a weighted or unweighted graph to convey it to the system. The system then uses the constraints to give out test cases and displays them on the output window. Thereafter the user has a choice to either copy it using ctrl c or download it in a .txt file.

6.1.12 Detailed Design

6.1.12.1 Module Detailed Design

6.1.12.1.1 Data Type Detailed Design

6.1.12.1.1.1 Design

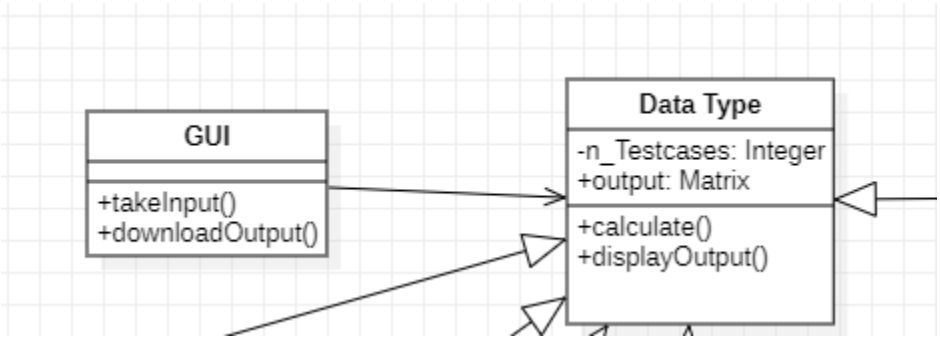


Figure 10, Data type Class Diagram

6.1.12.1.1.2 Design Description

The Data type class deserves special attention because it is the central class to all of the other Modules. As shown in the UML in the figure above, the GUI class is immediately called on application startup, instantiates the one and only data type class used in the system.

The Data Type class has a containment relationship with two other important Classes; testcases and output. These are shown as attributes in the Data Type UML class definition in the figure above.

One particular public method exposed on data type that needs explanation is calculate(). This method is used to calculate and produce testcase which are displayed using displayOutput(),

6.1.12.1.2 Integer Module Detailed Design

6.1.12.1.2.1 Design

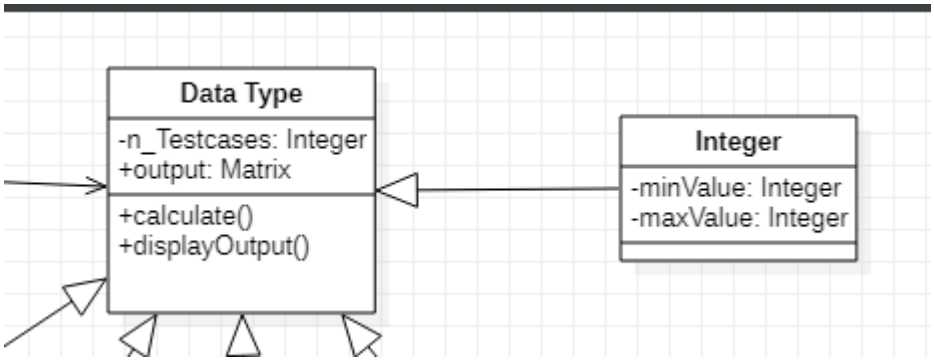


Figure 11, Integer Module Class Diagram

6.1.12.1.2.2 Design Description

The data type constructor will instantiate one instance of the calculate class. The Integer class is one of the instances and has two attributes which ask for a maximum and minimum value for integer namely minValue and maxValue

6.1.12.1.3 Array Module Detailed Design

6.1.12.1.3.1 Design

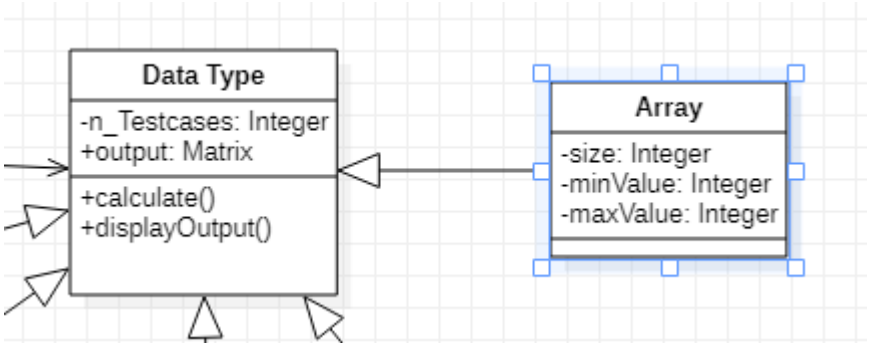


Figure 12, Array Module Class Diagram

6.1.12.1.3.2 Design Description

The data type constructor will instantiate one instance of the calculate class. The Array class is one of the instances and has two attributes which ask for a maximum and minimum value for integer namely minValue and maxValue.It also has an attribute which asks for the sixe of the array namely size

6.1.12.1.4 String Module Detailed Design

6.1.12.1.4.1 Design

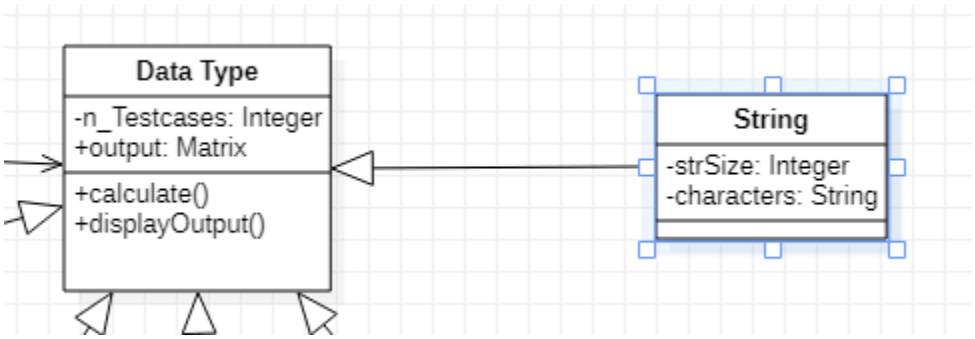


Figure 13, String Module Class Diagram

6.1.12.1.4.2 Design Description

The data type constructor will instantiate one instance of the calculate class. The String class is one of the instances and has an attributes which ask for the characters namely characters also has an attribute which asks for the size of the string namely size

6.1.12.1.5 Graph Module Detailed Design

6.1.12.1.5.1 Design

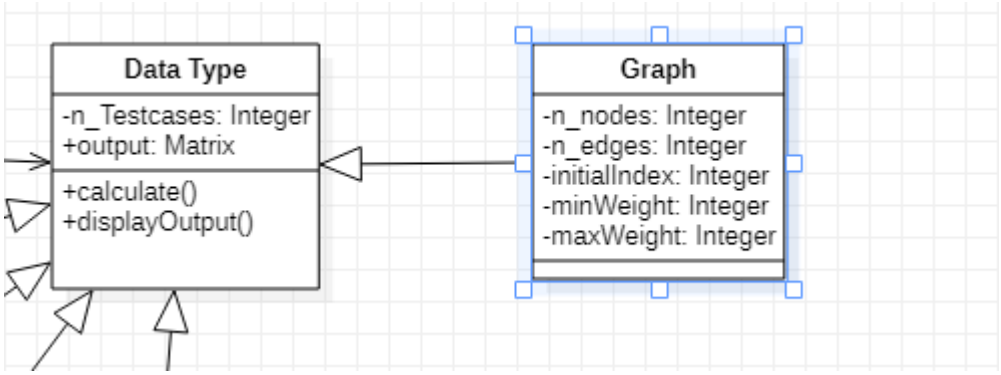


Figure 14, Graph Module Class Diagram

6.1.12.1.5.2 Design Description

The data type constructor will instantiate one instance of the calculate class. The graph class is one of the instances and has two attributes which ask for a maximum and minimum weight for graph namely minWeight and maxWeight .It also has an attribute which asks for the number of nodes namely n_nodes and number of edges namely n_edges. Also it has an attribute to give the initial index namely Initial Index.
.

6.1.12.1.6 Tree Module Detailed Design

6.1.12.1.6.1 Design

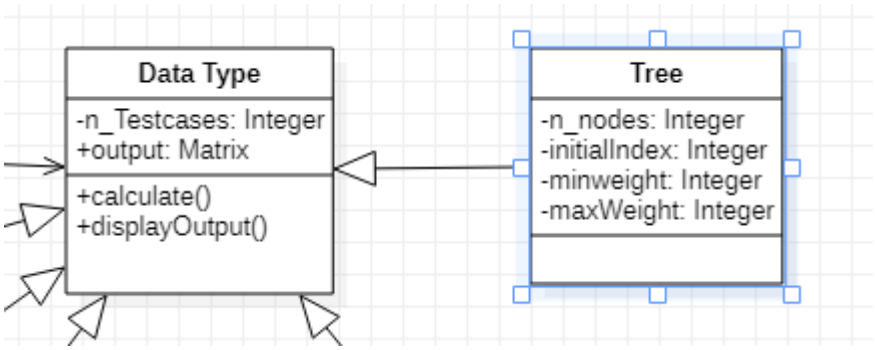


Figure 15, Tree Module Class Diagram

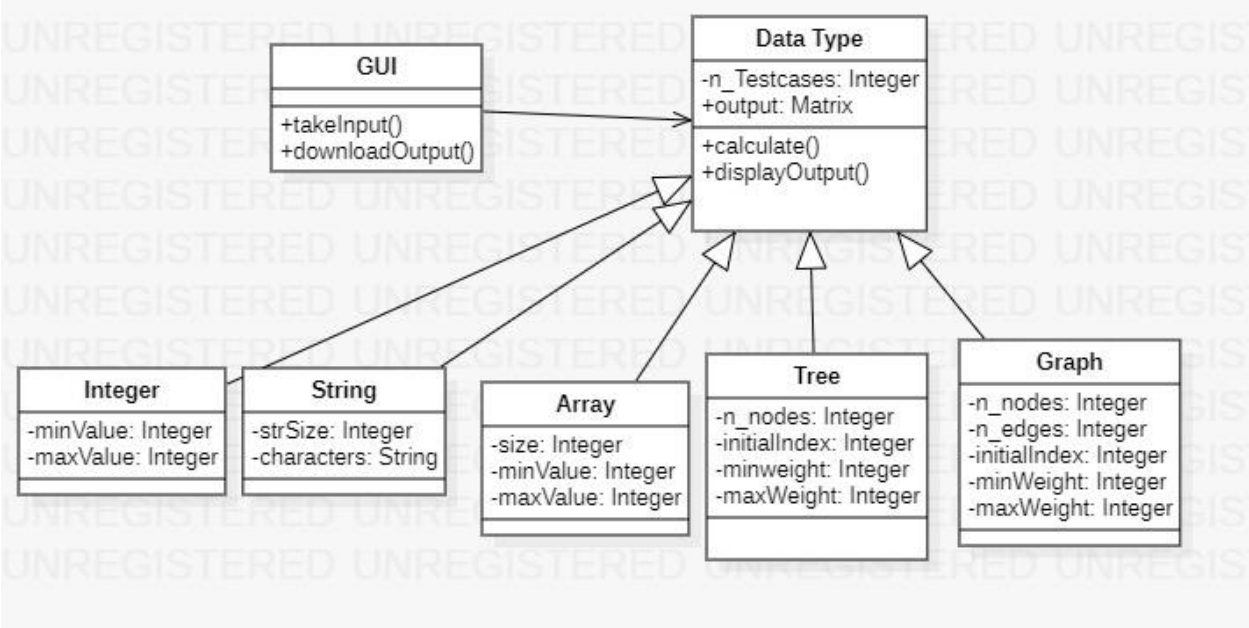
6.1.12.1.6.2 Design Description

The data type constructor will instantiate one instance of the calculate class. The Tree class is one of the instances and has two attributes which ask for a maximum and minimum weight for graph namely minWeight and maxWeight .It also has an attribute which asks for the number of nodes namely n_nodes and number of edges namely n_edges. Also it has an attribute to give the initial index namely Initial Index.

6.1.13 Other UML Diagrams

6.1.13.1 Class Diagram

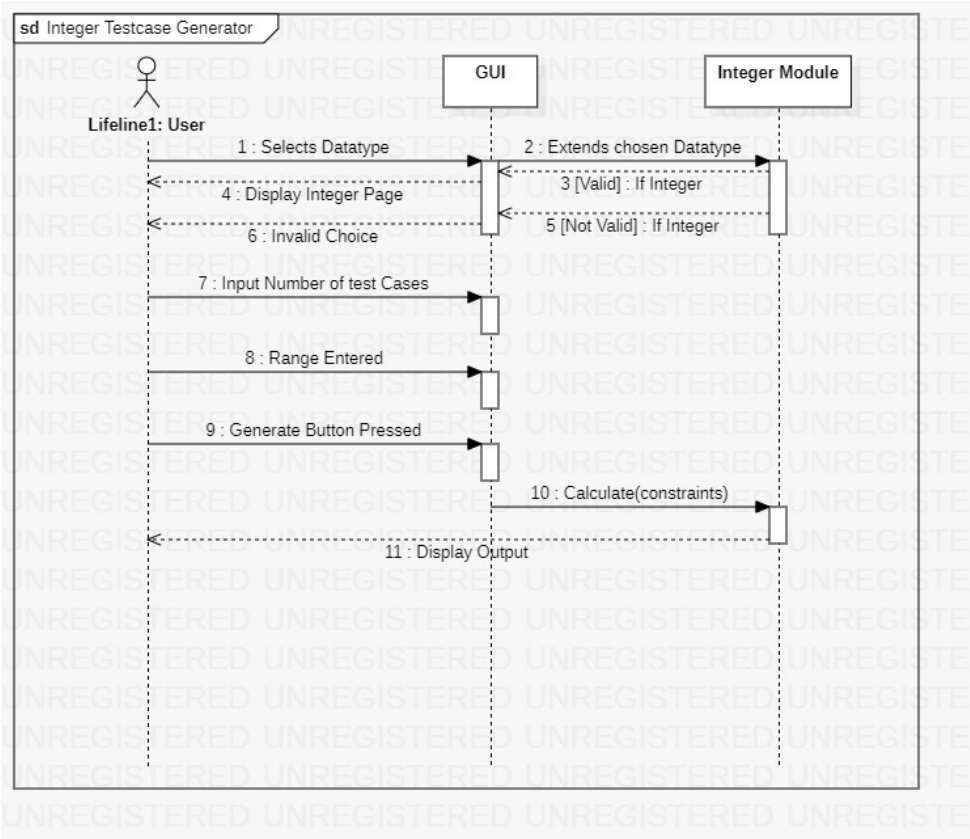
Figure 16, Class Diagram



6.1.13.2 Sequence and Collaboration Diagrams

Integer Sequence Diagram

Figure 17, Integer Sequence Diagram



Integer Collaboration Diagram

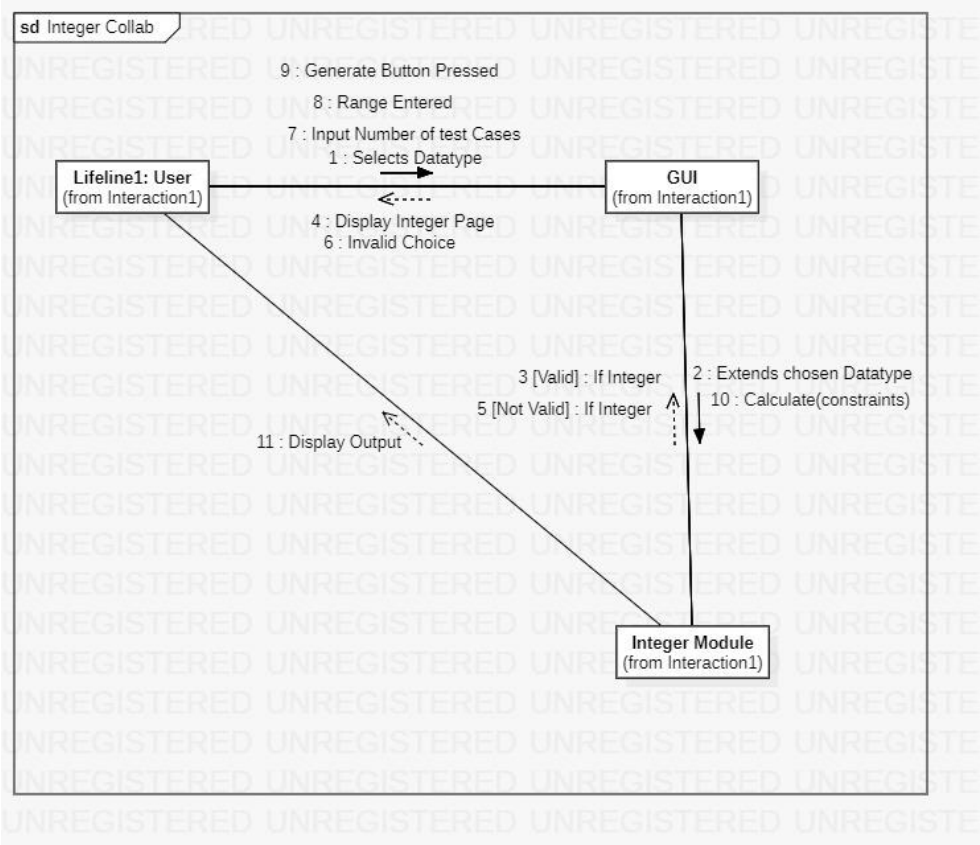
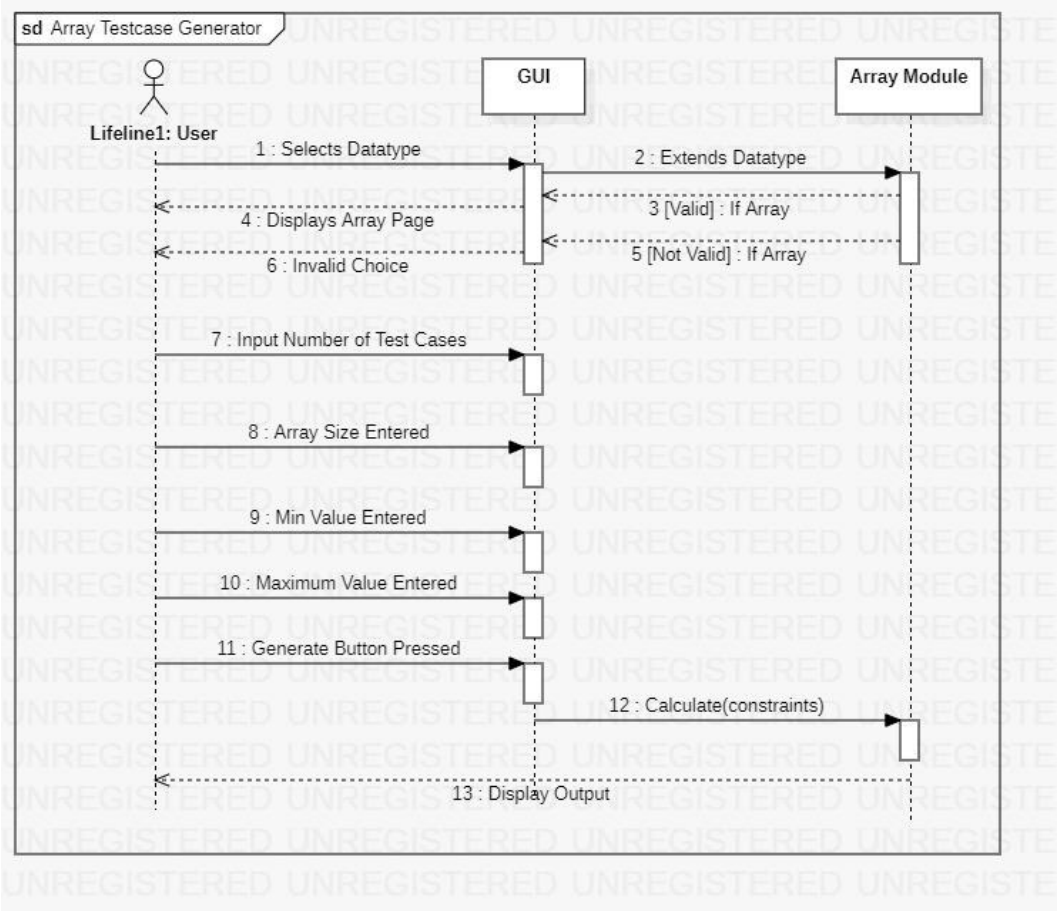


Figure 18, Integer Collaboration Diagram

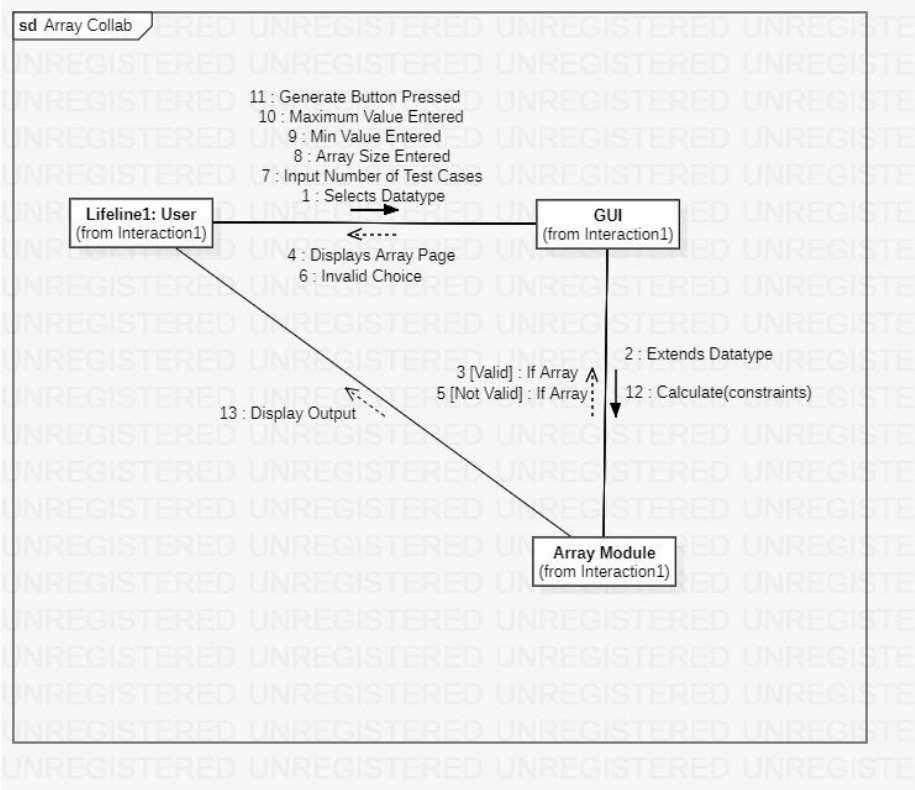
Array Sequence Diagram

Figure 19, Array Sequence Diagram



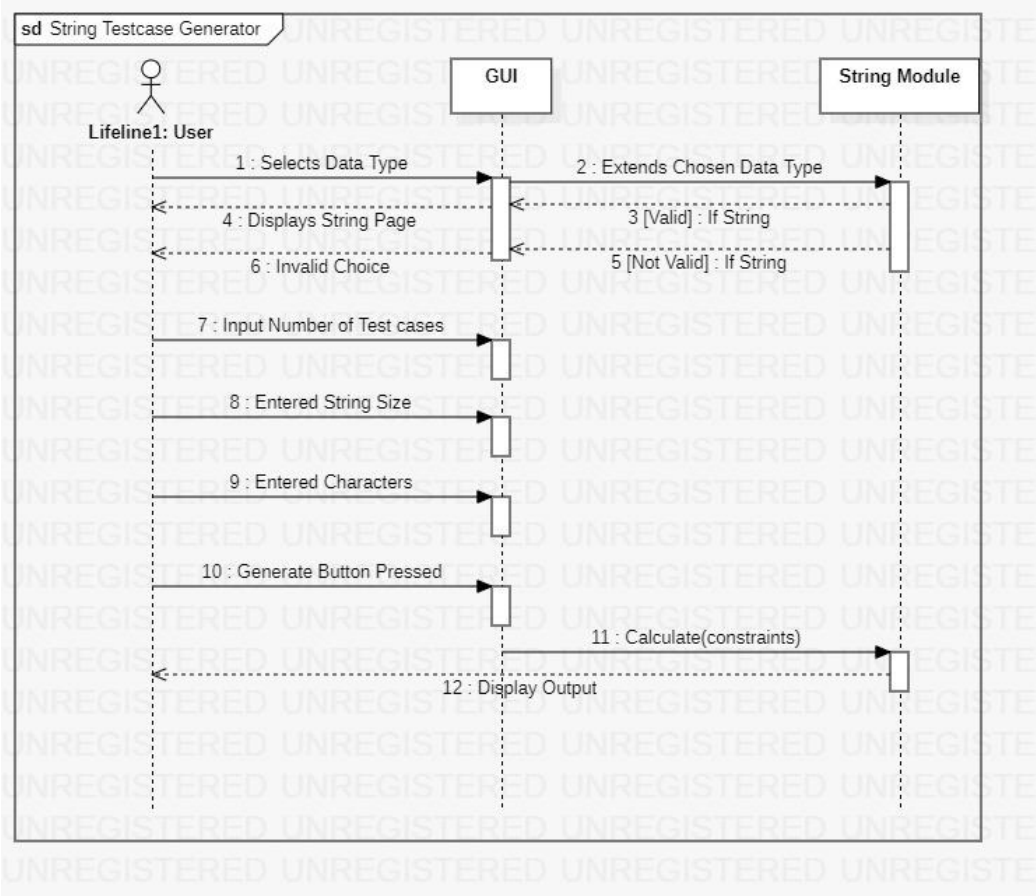
Array Collaboration Diagram

Figure 20, Array Collaboration Diagram



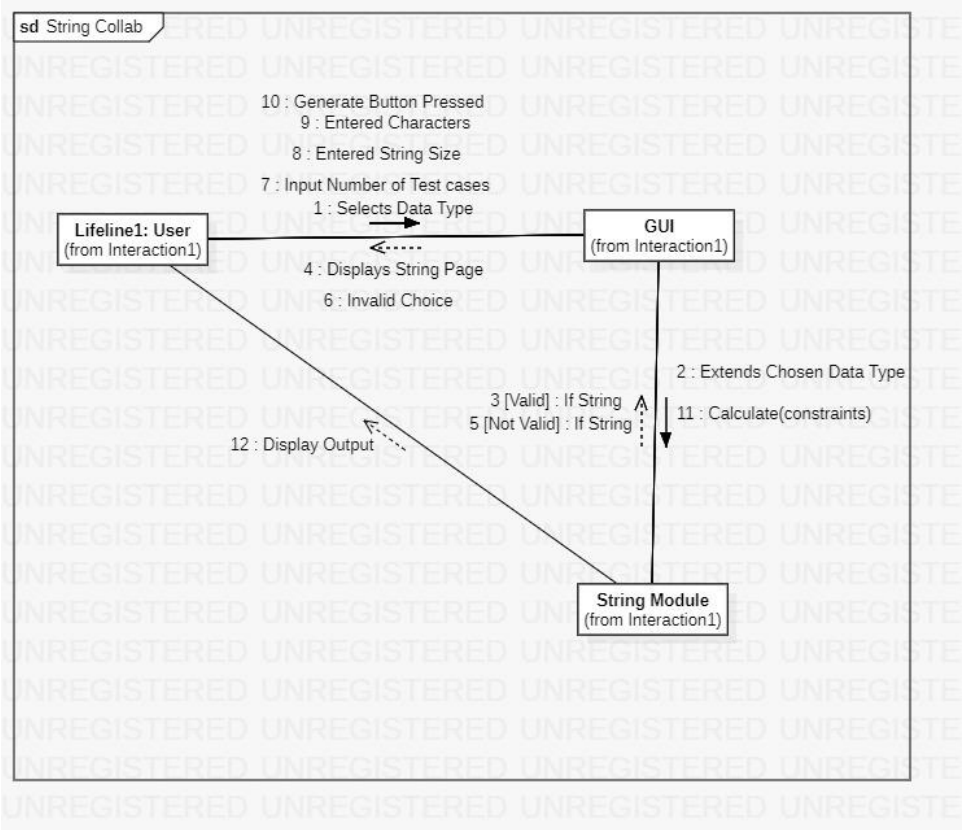
String Sequence Diagram

Figure 21, String Sequence Diagram



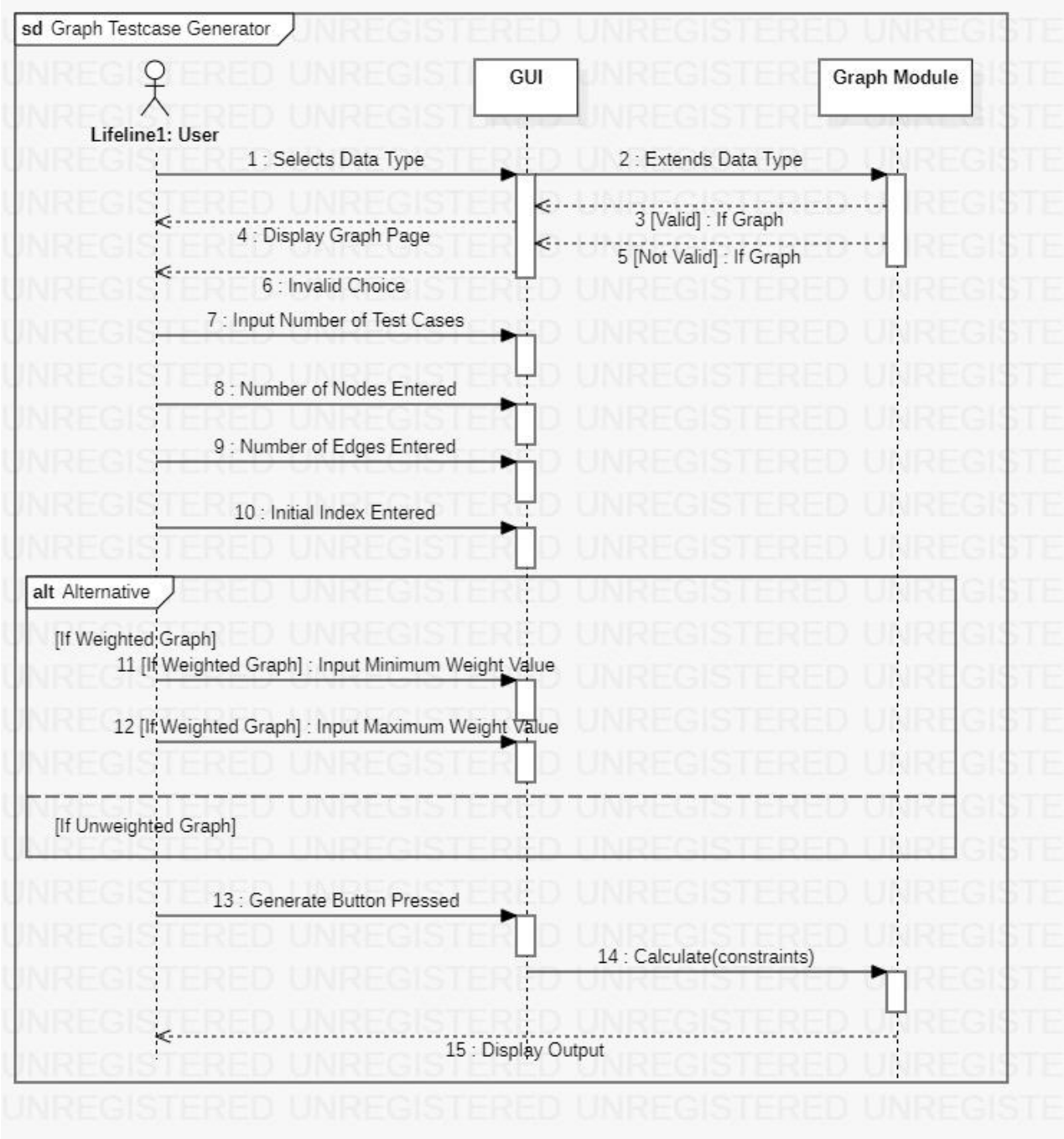
String Collaboration Diagram

Figure 22, String Collaboration Diagram



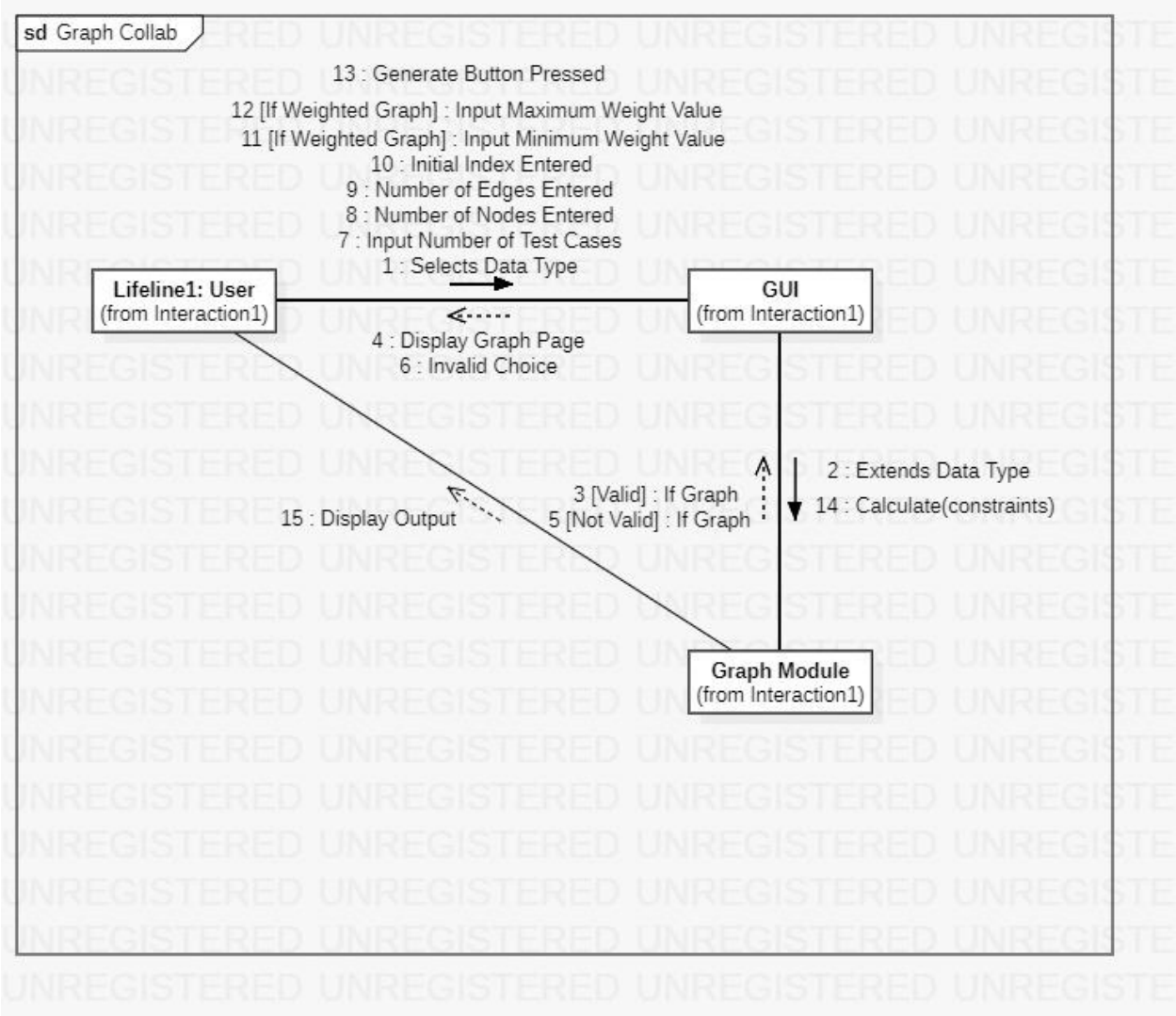
Graph Sequence Diagram

Figure 23, Graph Sequence Diagram



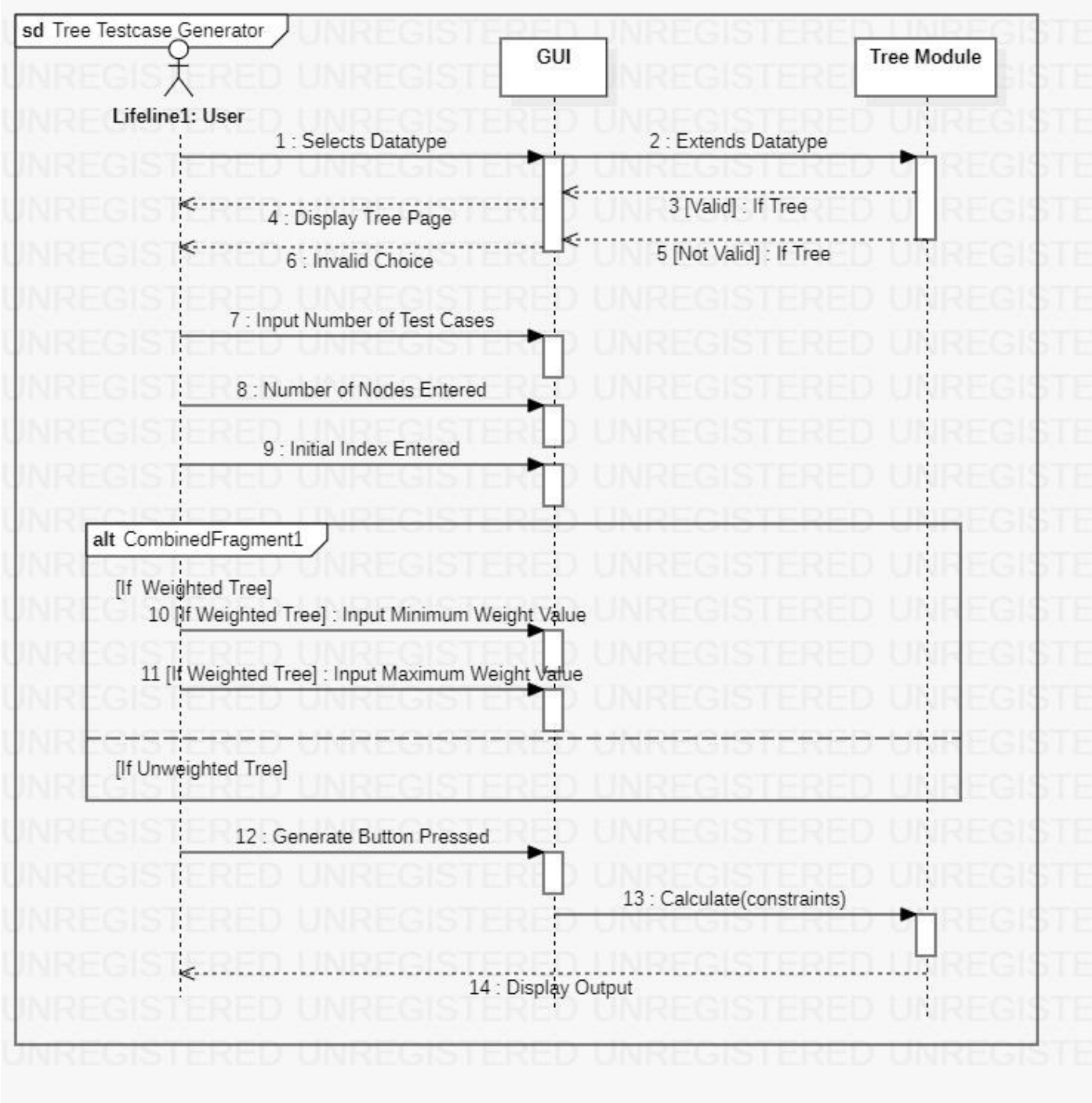
Graph Collaboration Diagram

Figure 24, Graph Collaboration Diagram



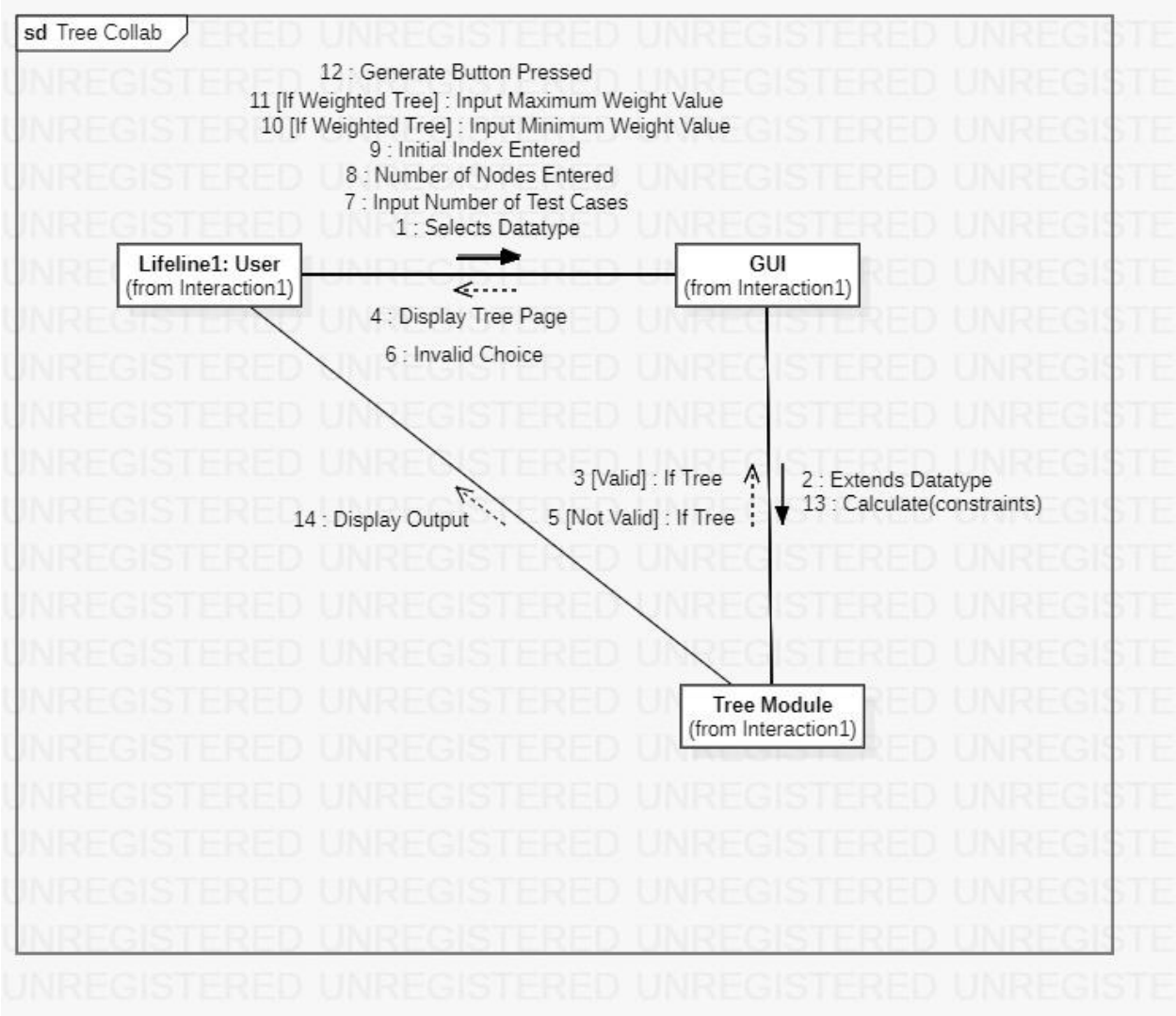
Tree Sequece Diagram

Figure 25, Tree Sequence Diagram



Tree Collaboration Diagram

Figure 26, Tree Collaboration Diagram



6.2 Codes And Standards

6.2.1 Style.css

```
@import
url('https://fonts.googleapis.com/css2?family=Noto+Serif:wght@400;700&display
=swap');
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    scroll-behavior: smooth;
}

body {
    font-family: 'Poppins', sans-serif;
}

.navbar {
    display: flex;
    align-items: center;
    padding: 20px;
    background-color: #000;
}

nav {
    flex: 1;
    text-align: center;
}

nav ul {
    display: inline-block;
    list-style-type: none;
}

nav ul li {
    display: inline-block;
    margin: 0 6rem 0 6rem;
}

a {
    text-decoration: none;
    color: #fff;
}

nav ul li:hover{
    background-color: #fff;
```

```
    padding: 5px;
    color: #000;
}

nav ul li a:hover{
    color: #000;
    font-weight: bold;
}

.active{
    color: yellow;
    font-weight: bold;
}

.menu-icon {
    width: 28px;
    margin-left: 20px;
    display: none;
}

.form{
    display: inline;
    float: left;
    width: 50vw ;
    margin: 50px;
    padding: 20px;
    margin-right: 0;
    margin-top: 100px;
}

#form-type-2{
    height: 30px;
    width: 40vw;
    margin: 10px ;
    padding-left: 10px;
    font-size: 15px;
    font-weight: bold;
    background-color: yellow;
}

form{
    margin-top: 30px;
}

input{
    display: block;
```

```
    width: 40vw;
    margin: 5px;
    padding: 5px;
}

.hidden{
    display: none;
}

.btn{
    background-color: #000;
    color: #fff;
    padding: 10px;
    margin: 5px;
    font-size: 20px;
}

.btn:hover{
    background-color: yellow;
    color: #000;
    font-weight: bold;
}

.download{
    background-color: rgb(8, 114, 25);
    margin: 20px 0 0 50px;
}

.download:hover{
    box-sizing: border-box;
    box-shadow: yellow;
    background-color: rgb(8, 114, 25);
    color: #fff;
}

.result{
    display: inline;
    float: left;
    width: 40vw ;
    margin: 50px;
    padding: 20px;
    margin-left: 0;
}
textarea{
    width: auto;
    height: 300px ;
```

```
    margin-top: 20px;
    font-size: 1.5rem;
    padding: 2rem;
}

@media only screen and (max-width: 1300px) {
    nav ul {
        position: absolute;
        left: 0;
        top: 70px;
        background: #333;
        width: 100%;
        overflow: hidden;
        transition: max-height 0.5s;
    }
    nav ul li {
        display: block;
        margin-right: 50px;
        margin-top: 10px;
        margin-bottom: 10px;
    }
    nav ul li a {
        color: #fff;
    }
    .menu-icon {
        display: block;
        cursor: pointer;
    }
    .navbar{
        background-color: #fff ;
    }
}

@media only screen and (max-width: 1100px){
    .result{
        display: block;
        width: 80vw;
    }
    .res{
        width: 100% ;
    }
}
```

6.2.2 HTML

6.2.2.1 Integer

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link rel="stylesheet" type="text/css" href="../CSS/style.css" >
      <title>TestCase Generator</title>

   </head>

   <body>
      <!-- Navigation Menu -->
      <div class="navbar">
         <nav>
            <ul id="MenuItems">
               <li><a href="#" class="active">Integer</a></li>
               <li><a href="string.html">String</a></li>
               <li><a href="./array.html">Array</a></li>
               <li><a href="./tree.html">Tree</a></li>
               <li><a href="./graph.html">Graph</a></li>
            </ul>
         </nav>
         <img src="../menu.png" class="menu-icon" onclick="menuToggle()">
      </div>


      <div class="form">
            <label for="test">Number of Test Cases *</label>
            <input id="test" name="test" type="number" required>
            <label for="min">Minimum Value *</label>
            <input id="min" name="min" type="number" required>
            <label for="max">Maximum Value *</label>
            <input id="max" name="max" type="number" required>
            <button class="btn" onclick="intgen()">Generate</button>
      </div>


      <div class="result">
         <h3>Output :</h3>
         <textarea id="res"></textarea>
         <br><br>
         <button type="button" class="btn download"
onclick="saveTextAsFile(res.value,'download.txt')">Download</button>
```

```
        </div>

        <!-------JS for Toggle Menu------->
        <script>
            var MenuItems = document.getElementById("MenuItems");
            MenuItems.style.maxHeight = "0px";

            function menuToggle() {
                if (MenuItems.style.maxHeight == "0px") {
                    MenuItems.style.maxHeight = "200px"
                } else {
                    MenuItems.style.maxHeight = "0px"
                }
            }
        </script>

        <script type="text/javascript" src="../JS/Data
Types/Integer/num.js"></script>
        <script type="text/javascript" src="../JS/script.js"></script>
    </body>
</html>
```

6.2.2.2 Array

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" type="text/css" href="../CSS/style.css" >
        <title>TestCase Generator</title>
    </head>

    <body>
        <!-- Navigation Menu -->
        <div class="navbar">
            <nav>
                <ul id="MenuItems">
                    <li><a href="./integer.html">Integer</a></li>
                    <li><a href="string.html">String</a></li>
                    <li><a href="#" class="active">Array</a></li>
                    <li><a href="./tree.html">Tree</a></li>
                    <li><a href="./graph.html">Graph</a></li>
                </ul>
            </nav>
            <img src="../menu.png" class="menu-icon" onclick="menuToggle()">
```

```
        </div>


        <div class="form">
            <label for="test">Number of Test Cases *</label>
            <input id="test" name="test" type="number" required>
            <label for="dim">Enter Dimensions of Array *(comma
seperated)</label>
            <input id="dim" name="dim" type="text" required>
            <label for="min">Minimum Value *</label>
            <input id="min" name="min" type="number" required>
            <label for="max">Maximum Value *</label>
            <input id="max" name="max" type="number" required>
            <button class="btn" onclick="arrgen()">Generate</button>
        </div>


        <div class="result">
           <h3>Output :</h3>
           <textarea id="res"></textarea>
           <button type="button" class="btn download"
onclick="saveTextAsFile(res.value,'download.txt')">Download</button>
        </div>

        <!-------JS for Toggle Menu------->
        <script>
           var MenuItems = document.getElementById("MenuItems");
           MenuItems.style.maxHeight = "0px";

           function menuToggle() {
              if (MenuItems.style.maxHeight == "0px") {
                 MenuItems.style.maxHeight = "200px"
              } else {
                 MenuItems.style.maxHeight = "0px"
              }
           }
        </script>

        <script type="text/javascript" src="../JS/Data Types/Array/array.js"></script>
        <script type="text/javascript" src="../JS/script.js"></script>
    </body>
</html>
```

6.2.2.3 String

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link rel="stylesheet" type="text/css" href="../CSS/style.css" >
      <title>TestCase Generator</title>
   </head>

   <body>
      <!-- Navigation Menu -->
      <div class="navbar">
         <nav>
            <ul id="MenuItems">
               <li><a href="./integer.html">Integer</a></li>
               <li><a href="#" class="active">String</a></li>
               <li><a href="./array.html">Array</a></li>
               <li><a href="./tree.html">Tree</a></li>
               <li><a href="./graph.html">Graph</a></li>
            </ul>
         </nav>
         <img src="../menu.png" class="menu-icon" onclick="menuToggle()">
      </div>


      <div class="form">
         <label for="test">Number of Test Cases *</label>
         <input id="test" name="test" type="number" required>
         <label for="size">Enter Size of the String *</label>
         <input id="size" name="size" type="number" required>
         <label for="chars">Enter the characters of the string</label>
         <input id="chars" name="chars" type="text">
         <button type="submit" class="btn" onclick="strgen()">Generate</button>
      </div>


      <div class="result">
         <h3>Output :</h3>
         <textarea id="res"></textarea>
         <button type="button" class="btn download"
onclick="saveTextAsFile(res.value,'download.txt')">Download</button>
      </div>

      <!-------JS for Toggle Menu------->
      <script>
```

```
        var MenuItems = document.getElementById("MenuItems");
        MenuItems.style.maxHeight = "0px";

        function menuToggle() {
            if (MenuItems.style.maxHeight == "0px") {
                MenuItems.style.maxHeight = "200px"
            } else {
                MenuItems.style.maxHeight = "0px"
            }
        }
    </script>

    <script type="text/javascript" src="../JS/Data Types/String/string.js"
></script>
    <script type="text/javascript" src="../JS/script.js"></script>
  </body>
</html>
```

6.2.2.4 Graphs

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="../CSS/style.css" >
    <title>TestCase Generator</title>
  </head>

  <body>
    <!-- Navigation Menu -->
    <div class="navbar">
      <nav>
        <ul id="MenuItems">
          <li><a href="./integer.html">Integer</a></li>
          <li><a href="./string.html">String</a></li>
          <li><a href="./array.html">Array</a></li>
          <li><a href="./tree.html">Tree</a></li>
          <li><a href="#" class="active">Graph</a></li>
        </ul>
      </nav>
      <img src="../menu.png" class="menu-icon" onclick="menuToggle()">
    </div>


    <div class="form">
      <div class="form-type">
```

```
            <select id="form-type-2" onchange="verify(event)">
                <option value="uug" selected>Undirected Unweighted
Graph</option>
                <option value="dug">Directed Unweighted Graph</option>
                <option value="dwg">Directed Weighted Graph</option>
            </select>
        </div>
            <label for="test">Number of Test Cases *</label>
            <input id="test" name="test" type="number" required>
            <label for="nodes">Number of Nodes *</label>
            <input id="nodes" name="nodes" type="number" required>
            <label for="edges">Number of Edges *</label>
            <input id="edges" name="edges" type="number" required>
            <label for="min_wt" id="l_min" class="hidden">Minimum
Weight</label>
            <input id="min_wt" name="min_wt" type="number" class="hidden">
            <label for="max_wt" id="l_max" class="hidden">Maximum
Weight</label>
            <input id="max_wt" name="max_wt" type="number" class="hidden">
            <button class="btn" onclick="uuggen()"
id="generate">Generate</button>
        </div>


    <div class="result">
        <h3>Output :</h3>
        <textarea id="res"></textarea>
        <button type="button" class="btn download"
onclick="saveTextAsFile(res.value,'download.txt')">Download</button>
    </div>

    <!-------JS for Toggle Menu------->
    <script>
        var MenuItems = document.getElementById("MenuItems");
        MenuItems.style.maxHeight = "0px";

        function menuToggle() {
            if (MenuItems.style.maxHeight == "0px") {
                MenuItems.style.maxHeight = "200px"
            } else {
                MenuItems.style.maxHeight = "0px"
            }
        }

        function verify(event) {
            var value = event.target.value ;
```

```
            let l_min = document.getElementById("l_min") ;
            let min_wt = document.getElementById("min_wt") ;
            let l_max = document.getElementById("l_max") ;
            let max_wt = document.getElementById("max_wt") ;
            let generate = document.getElementById("generate") ;

            if(value=="dwg"){
               l_min.style.display = "block" ;
               min_wt.style.display = "block" ;
               l_max.style.display = "block" ;
               max_wt.style.display = "block" ;
               generate.setAttribute("onclick","dwggen()") ;
            }

            else{
               l_min.style.display = "none" ;
               min_wt.style.display = "none" ;
               l_max.style.display = "none" ;
               max_wt.style.display = "none" ;
               if(value=="dug"){
                  generate.setAttribute("onclick","duggen()") ;
               }
               else{
                  generate.setAttribute("onclick","uuggen()") ;
               }
            }
         }
      </script>
      <script type="text/javascript" src="../JS/Data Types/Graph/UUGraph.js"
></script>
      <script type="text/javascript" src="../JS/Data Types/Graph/DUGraph.js"
></script>
      <script type="text/javascript" src="../JS/Data Types/Graph/DWGraph.js"
></script>
      <script type="text/javascript" src="../JS/script.js"></script>
   </body>
</html>
```

6.2.2.5 Trees

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link rel="stylesheet" type="text/css" href="../CSS/style.css" >
      <title>TestCase Generator</title>
```

```
    </head>

    <body>
        <!-- Navigation Menu -->
        <div class="navbar">
            <nav>
                <ul id="MenuItems">
                    <li><a href="./integer.html">Integer</a></li>
                    <li><a href="./string.html">String</a></li>
                    <li><a href="./array.html">Array</a></li>
                    <li><a href="#" class="active">Tree</a></li>
                    <li><a href="./graph.html">Graph</a></li>
                </ul>
            </nav>
            <img src="../menu.png" class="menu-icon" onclick="menuToggle()">
        </div>


        <div class="form">
            <div class="form-type">
                <select id="form-type-2" onchange="verify(event)">
                    <option selected value="ut">Unweighted Tree</option>
                    <option value="wt">Weighted Tree</option>
                </select>
            </div>
                <label for="test">Number of Test Cases *</label>
                <input id="test" name="test" type="number" required>
                <label for="nodes">Number of Nodes *</label>
                <input id="nodes" name="nodes" type="number" required>
                <label for="min_wt" id="l_min"class="hidden">Minimum Weight</label>
                <input id="min_wt" name="min_wt" type="number" class="hidden">
                <label for="max_wt" id="l_max" class="hidden">Maximum
Weight</label>
                <input id="max_wt" name="max_wt" type="number" class="hidden">
                <button class="btn" id="generate"
onclick="uwtgen()">Generate</button>
            </div>


        <div class="result">
            <h3>Output :</h3>
            <textarea id="res"></textarea>
            <button type="button" class="btn download"
onclick="saveTextAsFile(res.value,'download.txt')">Download</button>
        </div>
```

```
        <!-------JS for Toggle Menu------->
        <script>
            var MenuItems = document.getElementById("MenuItems");
            MenuItems.style.maxHeight = "0px";

            function menuToggle() {
                if (MenuItems.style.maxHeight == "0px") {
                    MenuItems.style.maxHeight = "200px"
                } else {
                    MenuItems.style.maxHeight = "0px"
                }
            }

            function verify(event) {
                var value = event.target.value ;
                let l_min = document.getElementById("l_min") ;
                let min_wt = document.getElementById("min_wt") ;
                let l_max = document.getElementById("l_max") ;
                let max_wt = document.getElementById("max_wt") ;
                let generate = document.getElementById("generate") ;

                if(value=="wt"){
                    l_min.style.display = "block" ;
                    min_wt.style.display = "block" ;
                    l_max.style.display = "block" ;
                    max_wt.style.display = "block" ;
                    generate.setAttribute("onclick","wtgen()") ;
                }

                else{
                    l_min.style.display = "none" ;
                    min_wt.style.display = "none" ;
                    l_max.style.display = "none" ;
                    max_wt.style.display = "none" ;
                    generate.setAttribute("onclick","uwtgen()") ;
                }
            }
        </script>
        <script src="../JS/Data Types/Tree/UWTree.js" type="text/javascript"
></script>
        <script src="../JS/Data Types/Tree/WTree.js" type="text/javascript"
></script>
        <script type="text/javascript" src="../JS/script.js"></script>
    </body>
</html>
```

6.2.3 JavaScript

6.2.3.1 Integer

```
function num(test,min,max) {
    let result = '';
    for(let i=0;i<test;i++){
        result += Math.floor(Math.random() * (max - min + 1)) + min + '\n' ;
    }
    return result ;
}

let result ;
function intgen(){
    let test = parseInt(document.getElementById('test').value) ;
    let min = parseInt(document.getElementById('min').value) ;
    let max = parseInt(document.getElementById('max').value) ;
    console.log(test,min,max) ;
    result = num(test,min,max) ;

    console.log(result) ;
    var txtar = document.getElementById('res') ;
    txtar.value = result ;
}
```

6.2.3.2 Array

```
const randarrgen=(dim,min,max)=>{
    min = Math.ceil(min);
    max = Math.floor(max);
    var string = "";
    const row = dim[0], col = dim[1];
    for(var i=0;i<row;i++){
        for(var j=0;j<col;j++){
            const num = Math.floor(Math.random()*(max - min + 1))+min
            string+=num+' ';
        }
        string+='\n';
    }
    return string;
}

function array(test,dim,min,max){
    min = Number(min);
    max = Number(max);
    dim = dim.split(",");
    var result = "";
```

```
    for(let i=0;i<test;i++){
        result+=randarrgen(dim,min,max)+'\n';
    }
    return result;
}

let result ;
function arrgen(){
    let test = parseInt(document.getElementById('test').value) ;
    let dim = document.getElementById('dim').value ;
    let min = parseInt(document.getElementById('min').value) ;
    let max = parseInt(document.getElementById('max').value) ;
    console.log(test,dim,min,max) ;
    result = array(test,dim,min,max) ;

    console.log(result) ;
    var txtar = document.getElementById('res') ;
    txtar.value = result ;
}
```

6.2.3.3 String

```
const randstrgen=(size,chars)=>{
    var string = "";
    for(var i=0;i<size;i++){
        string+=chars[Math.round(Math.random() * (chars.length - 1))];
    }
    return string;
}

function string(test, size, chars){
    chars = chars===''? "qwertyuioplkjhgfdsazxcvbnm" : chars;
    var result="";
    for(let i=0;i<test;i++) {
        result+=randstrgen(size,chars)+'\n';
    }
    return result;
};

let result ;
function strgen(){
    let test = parseInt(document.getElementById('test').value) ;
    let size = parseInt(document.getElementById('size').value) ;
    let chars = document.getElementById('chars').value ;
    console.log(test,size,chars) ;
    result = string(test,size,chars) ;
```

```
    console.log(result) ;
    var txtar = document.getElementById('res') ;
    txtar.value = result ;
}
```

6.2.3.4 Graphs

```
function num(test,min,max) {
    let result = '';
    for(let i=0;i<test;i++){
        result += Math.floor(Math.random() * (max - min + 1)) + min + '\n' ;
    }
    return result ;
}

// var edges = 6 ;
// var nodes = 4;
// var test = 2;

const randduggen=(nodes,edges)=>{

    var string = "";
    var container = new Set();

    for(var i=1;i<=edges;i++){
        var a = Number(num(1,1,nodes));
        var b = Number(num(1,1,nodes));
        var p = [a,b];
        while(container.has(`${p[0]},${p[1]}`)){
            var a = Number(num(1,1,nodes));
            var b = Number(num(1,1,nodes));
            var p = [a,b];
        }
        container.add(`${p[0]},${p[1]}`);
    }

    container.forEach((elem)=>{string+=elem.replace(/,/g,'       ')+'\n'});
    return string;
};

function DUGraph(test,dug_nodes,dug_edges){
    dug_nodes = Number(dug_nodes);
    dug_edges = Number(dug_edges);
    var result = "";
    for(let i=0;i<test;i++){
```

```
        result+=randduggen(dug_nodes,dug_edges)+'\n\n';
    }

    return result;
};

let result1 = "Edge   Edge\nFrom   To\n" ;
function duggen(){
    let result1 = "Edge   Edge\nFrom   To\n" ;
    let test = parseInt(document.getElementById('test').value) ;
    let nodes = parseInt(document.getElementById('nodes').value) ;
    let edges = parseInt(document.getElementById('edges').value) ;

    console.log(test,nodes,edges) ;
    result1 += DUGraph(test,nodes,edges) ;

    console.log(result1) ;
    var txtar = document.getElementById('res') ;
    txtar.value = result1 ;
}
function num(test,min,max) {
    let result = '';
    for(let i=0;i<test;i++){
        result += Math.floor(Math.random() * (max - min + 1)) + min + '\n' ;
    }
    return result ;
}

// let nodes = 5 ;
// let edges = 8 ;
// let min_wt = 5 ;
// let max_wt = 9 ;
// let test = 1 ;

const randdwggen=(nodes,edges,min_weight,max_weight)=>{

    var string = "";
    var container = new Set();

    for(var i=1;i<=edges;i++){
        var a = Number(num(1,1,nodes));
        var b = Number(num(1,1,nodes));
        var p = [a,b];
        while(container.has(`${p[0]},${p[1]}`)){
            var a = Number(num(1,1,nodes));
            var b = Number(num(1,1,nodes));
```

```
            var p = [a,b];

        }
        container.add(`${p[0]},${p[1]}`);
    }

    container.forEach((elem)=>{string+=elem.replace(/,/g,'       ')+"
"+Number(num(1,min_weight,max_weight))+'\n'});
    return string;
};

function
DWGraph(test,dwg_nodes,dwg_edges,dwg_min_weight,dwg_max_weight){
    dwg_nodes = Number(dwg_nodes);
    dwg_edges = Number(dwg_edges);
    dwg_min_weight = Number(dwg_min_weight);
    dwg_max_weight = Number(dwg_max_weight);
    var result = "";
    for(let i=0;i<test;i++){

result+=randdwggen(dwg_nodes,dwg_edges,dwg_min_weight,dwg_max_weight
)+'\n\n';
    }

    return result;
}


function dwggen(){
    let result2 = "Edge   Edge  Weight\nFrom   To\n" ;
    let test = parseInt(document.getElementById('test').value) ;
    let nodes = parseInt(document.getElementById('nodes').value) ;
    let edges = parseInt(document.getElementById('edges').value) ;
    let min_wt = parseInt(document.getElementById('min_wt').value) ;
    let max_wt = parseInt(document.getElementById('max_wt').value) ;

    if(!min_wt)
        min_wt = 0 ;
    if(!max_wt)
        max_wt = 0 ;

    console.log(test,nodes,edges,min_wt,max_wt) ;
    result2 += DWGraph(test,nodes,edges,min_wt,max_wt) ;

    console.log(result2) ;
    var txtar = document.getElementById('res') ;
```

```
    txtar.value = result2 ;
}
function num(test,min,max) {
    let result = '';
    for(let i=0;i<test;i++){
        result += Math.floor(Math.random() * (max - min + 1)) + min + '\n' ;
    }
    return result ;
}

// let nodes = 4;
// let edges = 7;
// let test = 2 ;

const randuuggen=(nodes,edges)=>{

    var string = "";
    var container = new Set();

    for(var i=1;i<=edges;i++){
        var a = Number(num(1,1,nodes));
        var b = Number(num(1,1,nodes));
        var p = [a,b];
        var rev_p = [b,a];

        while(container.has(`${p[0]},${p[1]}`) ||
container.has(`${rev_p[0]},${rev_p[1]}`)){
            var a = num(1,1,nodes);
            var b = num(1,1,nodes);
            a=Number(a);
            b=Number(b);
            var p = [a,b];
            var rev_p = [b,a];
            //console.log(p,rev_p) ;
        }
        container.add(`${p[0]},${p[1]}`);
    }

    container.forEach((elem)=>{string+=elem.replace(/,/g,'       ')+'\n'});
    return string;
}

function UUGraph(test, uug_nodes,uug_edges){
    uug_nodes=Number(uug_nodes);
    uug_edges=Number(uug_edges);
    var result = "";
```

```
    for(let i=0;i<test;i++){
        result+=randuuggen(uug_nodes,uug_edges)+'\n\n';
    }

    return result;
}


function uuggen(){
    let result = "Edge   Edge\nFrom   To\n" ;
    let test = parseInt(document.getElementById('test').value) ;
    let nodes = parseInt(document.getElementById('nodes').value) ;
    let edges = parseInt(document.getElementById('edges').value) ;

    console.log(test,nodes,edges) ;
    result += UUGraph(test,nodes,edges) ;

    console.log(result) ;
    var txtar = document.getElementById('res') ;
    txtar.value = result ;
}

// console.log("No. of Test Cases: " + test);
// console.log("Number of Nodes: " + nodes);
// console.log("Number of Edges: " + edges);
// console.log("Output: ");

// console.log(UUGraph(test,nodes,edges));

// module.exports = UUGraph ;
```

6.2.3.5 Trees

```
function num(test,min,max) {
    let result = '';
    for(let i=0;i<test;i++){
        result += Math.floor(Math.random() * (max - min + 1)) + min + '\n' ;
    }
    return result ;
}

// let nodes = 6;
// let test = 3 ;

class Tree {
    constructor(nodes) {
```

```
        this.nodes = nodes;
        this.adj = Array.from({length: nodes}, () => []);
    }
    addEdge(n, w) {
        this.adj[n].push(w);
    }
    descendants(node) {
        let visited = new Set([node]);
        for (let node of visited) {
            for (let elem of this.adj[node]) {
                if (!visited.has(elem)) visited.add(elem);
            }
        }
        return visited;
    }
}

function shuffle(array) {
    for (var i = array.length - 1; i > 0; i--) {
        var j = Math.floor(Math.random() * (i + 1));
        var temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }
    return array;
}

function randtreegen(nodes) {
    let string = "";
    let t = new Tree(nodes);
    let [root, ...children] = shuffle([...Array(nodes).keys()]);
    let edges = [];
    let a;
    for (let b of children) {
        do {
            a = num(1,0, nodes-1); // make zero based
            a = Number(a) ;
        } while (t.descendants(b).has(a));
        t.addEdge(a, b);
        edges.push([a, b]);
    }
    string+=edges.join('\n').replace(/,/g,'        ');
    return string
}

function UWTree(test, tree_nodes) {
```

```
    tree_nodes = Number(tree_nodes);
    var result = "";
    for(let i=0;i<test;i++){
        result += randtreegen(tree_nodes) + '\n\n';
    }

    return result;
}


function uwtgen(){
    let result = "Parent  Child\n\n" ;
    let test = parseInt(document.getElementById('test').value) ;
    let nodes = parseInt(document.getElementById('nodes').value) ;
    console.log(test,nodes) ;
    result += UWTree(test,nodes) ;

    console.log(result) ;
    var txtar = document.getElementById('res') ;
    txtar.value = result ;
}

// console.log("No. of Test Cases: " + test);
// console.log("Number of Nodes: " + nodes);
// console.log("Output: ");

// console.log(UWTree(test,nodes));

// module.exports = UWTree ;
function num(test,min,max) {
    let result = '';
    for(let i=0;i<test;i++){
        result += Math.floor(Math.random() * (max - min + 1)) + min + '\n' ;
    }
    return result ;
}

// let nodes = 4 ;
// let min_wt = 2 ;
// let max_wt = 11;
// let test = 3 ;

class WeightedTree {
    constructor(nodes) {
        this.nodes = nodes;
        this.adj = Array.from({length: nodes}, () => []);
```

```
    }
    addEdge(n, w) {
        this.adj[n].push(w);
    }
    descendants(node) {
        let visited = new Set([node]);
        for (let node of visited) {
            for (let elem of this.adj[node]) {
                if (!visited.has(elem)) visited.add(elem);
            }
        }
        return visited;
    }
}

function shuffle(array) {
    for (var i = array.length - 1; i > 0; i--) {
        var j = Math.floor(Math.random() * (i + 1));
        var temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }
    return array;
}

function randwtgen(nodes,min_weight,max_weight) {
    let string = "";
    let t = new WeightedTree(nodes);
    let [root, ...children] = shuffle([...Array(nodes).keys()]);
    let edges = [];
    let a;
    for (let b of children) {
        do {
            a = num(1,0, nodes-1); // make zero based
            a=Number(a) ;
        } while (t.descendants(b).has(a));
        t.addEdge(a, b);
        edges.push([a, b,num(1,min_weight,max_weight)]);
    }
    string+=edges.join("\n").replace(/,/g,'        ');
    return string
}

function WTree(test, tree_nodes,wt_min_weight,wt_max_weight) {
    tree_nodes = Number(tree_nodes);
    wt_min_weight = Number(wt_min_weight);
```

```
    wt_max_weight = Number(wt_max_weight);
    var result = "";
    for(let i=0;i<test;i++){
        result += randwtgen(tree_nodes,wt_min_weight,wt_max_weight) + '\n\n' ;
    }

    return result;
}


function wtgen(){
    let result1 = "Parent  Child Weight\n\n" ;
    let test = parseInt(document.getElementById('test').value) ;
    let nodes = parseInt(document.getElementById('nodes').value) ;
    let min_wt = parseInt(document.getElementById('min_wt').value) ;
    let max_wt = parseInt(document.getElementById('max_wt').value) ;

    if(!min_wt)
        min_wt = 0 ;
    if(!max_wt)
        max_wt = 0 ;

    console.log(test,nodes,min_wt,max_wt) ;
    result1 += WTree(test,nodes,min_wt,max_wt) ;

    console.log(result1) ;
    var txtar = document.getElementById('res') ;
    txtar.value = result1 ;
}

// console.log("No. of Test Cases: " + test);
// console.log("Number of Nodes: " + nodes);
// console.log("Minimum Weight: " + min_wt);
// console.log("Maximum Weight: " + max_wt);
```

6.3 Constraints, alternatives and tradeoffs

6.3.1 Constraints

The follow is a table of the design constraints that the system SHALL meet. The list of constraints was produced from the initial project documentation provided by the requirements expert.

Table of Design Constraints

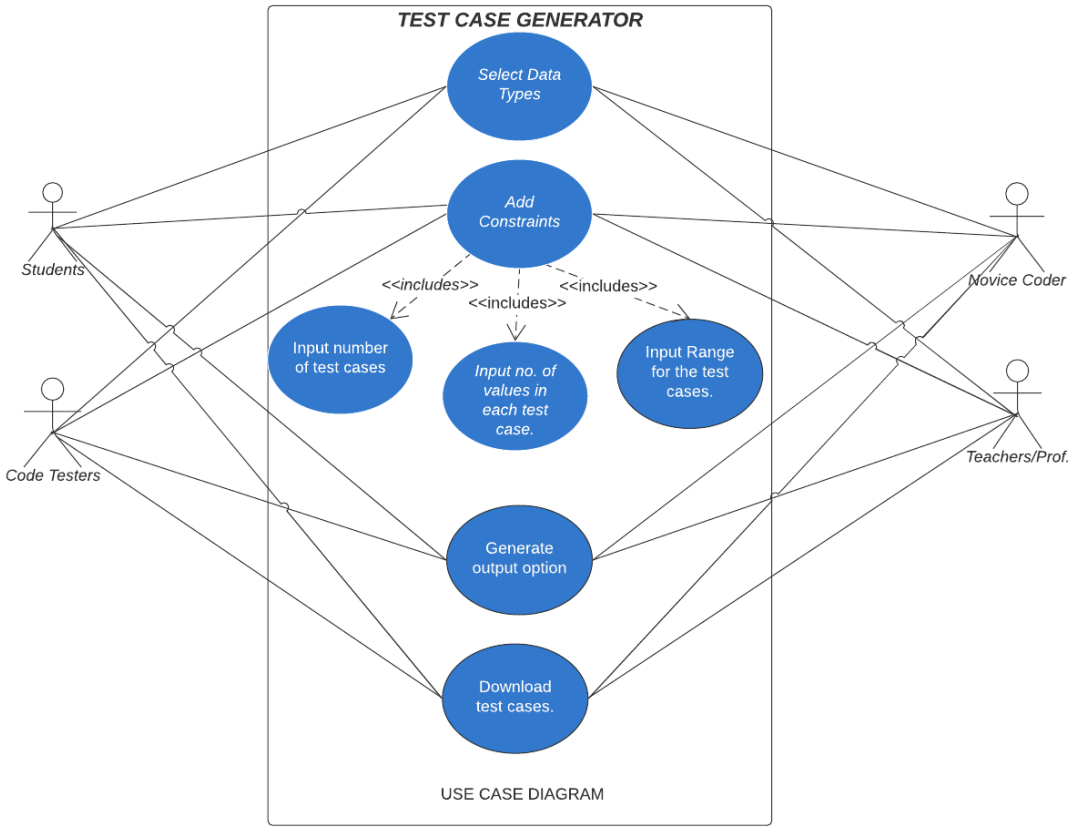
ID	Origin	***Shall Requirement***

ID	Origin	*Shall Requirement*
1	*Use case diagram*	*The system SHALL be able to guide the user to the constraints page of the selected datatype.*
2	*Use case diagram*	*The system SHALL be able to verify the validity of the entered constraints.*
3	*Use case diagram*	*The system SHALL be able to provide random outputs that are strictly adhered to the constraints.*
4	*Use case diagram*	*The output SHALL not be encrypted*
5	*Use case diagram*	*The system SHALL provide a easy to implement version of the test cases which are to be used.*
6	*Use case diagram*	*The system SHALL not be able to give any assistance with the code in particular.*
7	*Use case diagram*	*The system SHALL allow a user to discard all generated test cases and request for new test cases.*
8	*Use case diagram*	*The system SHALL allow the recipient to receive email from "blacklisted" domains if the email has a stamp.*
9	*Use case diagram*	*The system SHALL allow the user to download a particular test case as a .txt file*

External constraints

Design Constraint	Description
JavaScript	The source code is written in JavaScript and because of this, the source code written to implement the external features of the Test Case Generator will also be written in JavaScript.
Object Oriented Design and Programming	Test Case Generator will also employ the Object-Oriented Design and Programming methods to best integrate with coding software.

Figure 27, Use Case Diagram

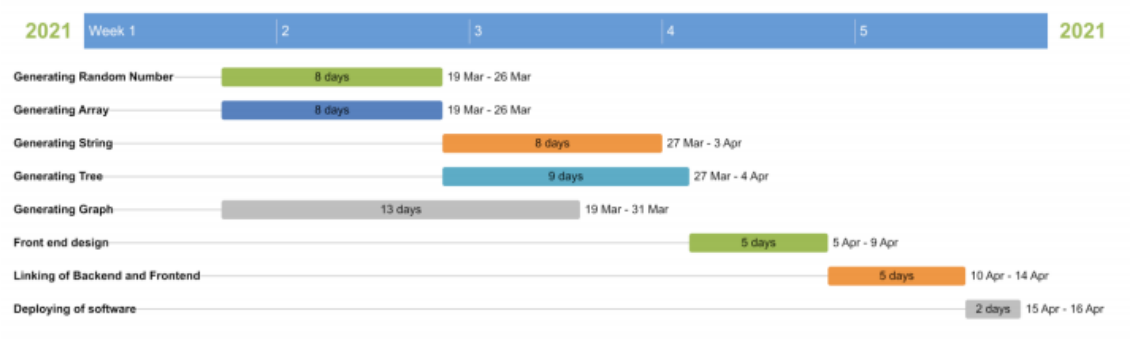


7 Schedule, Tasks and Milestones

7.1 Schedule:

The following Model Shows the Timeline and Schedule for the Project

Figure 28, Schedule Diagram



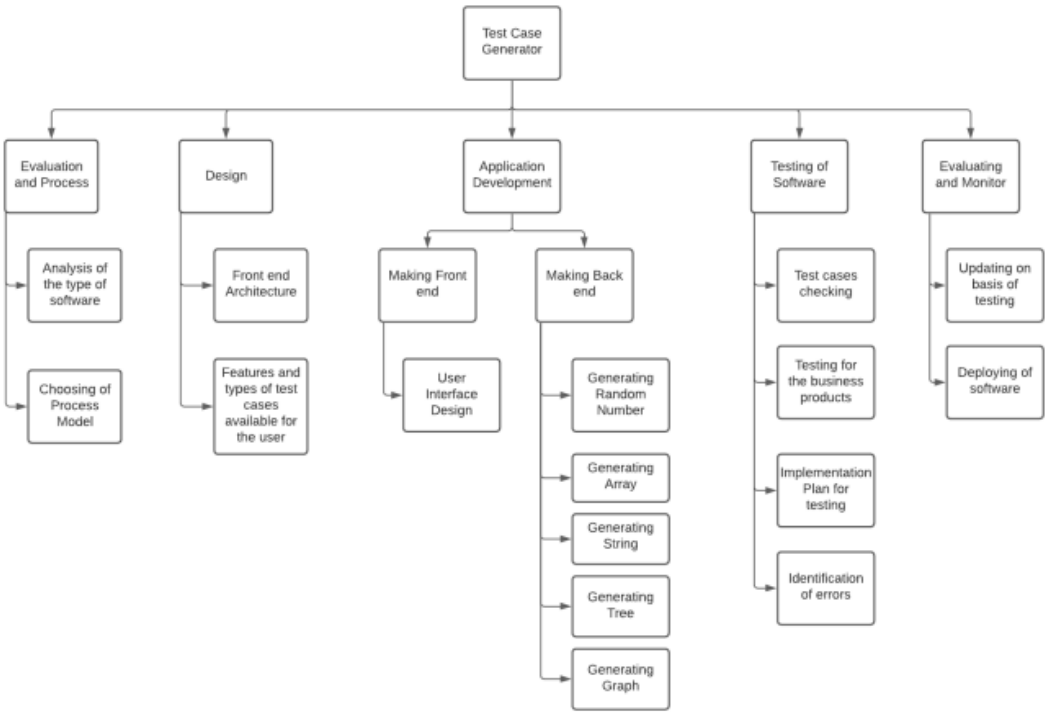
7.2 Tasks:

The following Diagram Outlays the different tasks and breakdown of work that was completed by our Team

Figure 29, Activity Network Diagram

TASK	LABEL	PREDECESSOR	STAFF REQUIRED	ESTIMATION DURATION
Evaluation and Process	A	-	3	19 Days
Design	B	A	3	18 Days
Application Design (Development)	C	B	3	29 Days
Application, construction and implementation	D	C	3	19 Days
Evaluating and Monitoring	E	D	3	13 Days

Figure 30, Work Breakdown Structure



7.3 Milestones:

7.3.1 Milestone 1: - (18 March, 2021)

- EVALUATION AND PROCESS REQUREMENTS
- SOFTWARE AND DESIGN SPECIFICATIONS

7.3.2 Milestone 2: - (16 April)

- DESIGN OF SOFTWARE
- PROTOTYPE GUI

7.3.3 Milestone 3: - (18th May)

- DEVELOPMENT OF SOFTWARE

- TESTING OF SOFTWARE

8 Project Demonstration:

LINK TO GITHUB REPOSITORY:

https://github.com/21Shadow10/Test-Case-Generator

LINK TO WEBPAGE:

https://21shadow10.github.io/Test-Case-Generator/HTML/integer.html

LINK TO GOOGLE DRIVE DOCUMENT WITH DEMO VIDEO AND PPT:

https://drive.google.com/drive/folders/1eeJqi5qsiTBBSTO1rJAEQBnvusVLqDgB?usp=sharing

The following Pictures give a step-by-step demonstration of how our software works

Figure 31, STEP 1: CHOOSE DATATYPE

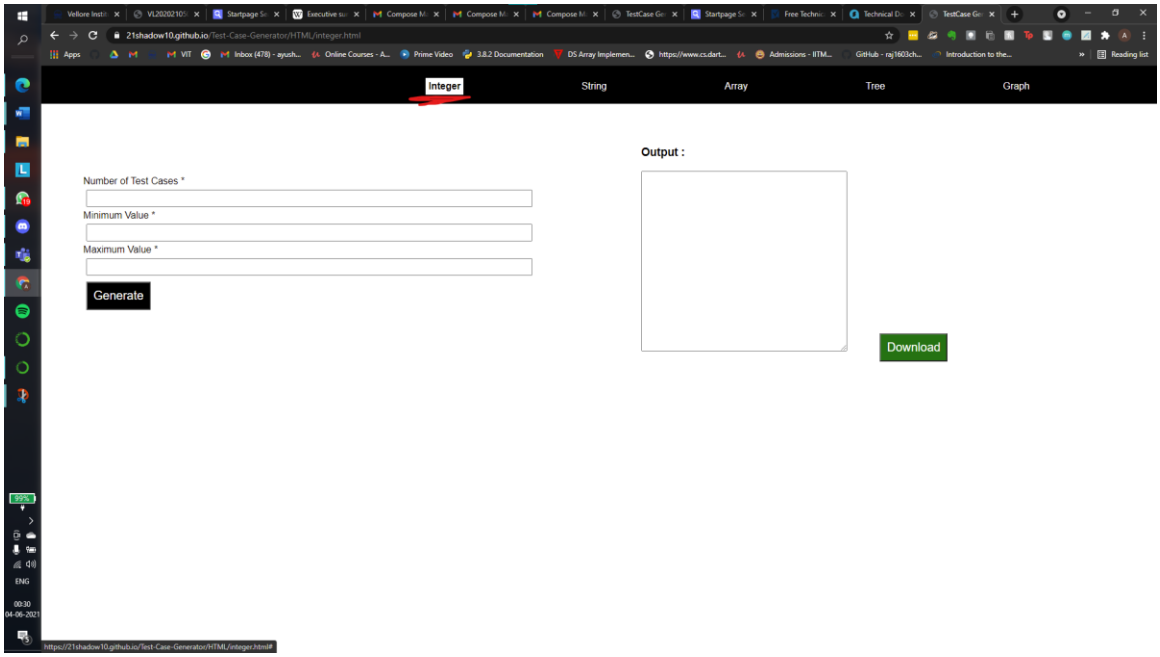


Figure 32, STEP 2: ENTER CONSTRAINTS AND PRESS GENERATE

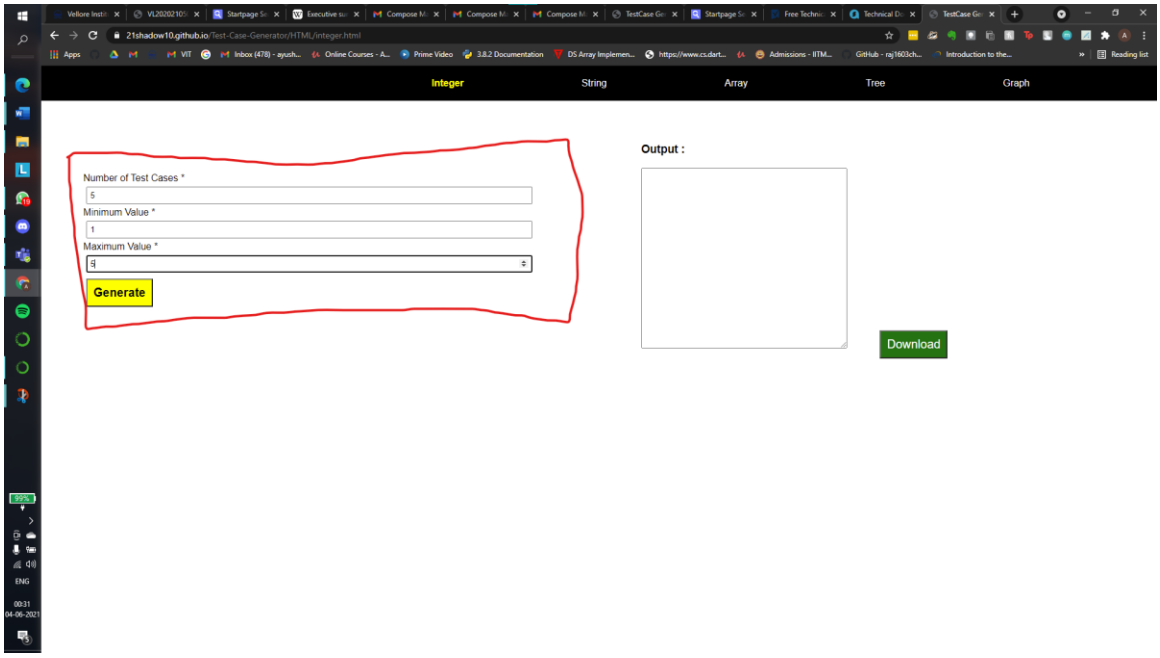


Figure 33, STEP 3: CLICK ON DOWNLOAD

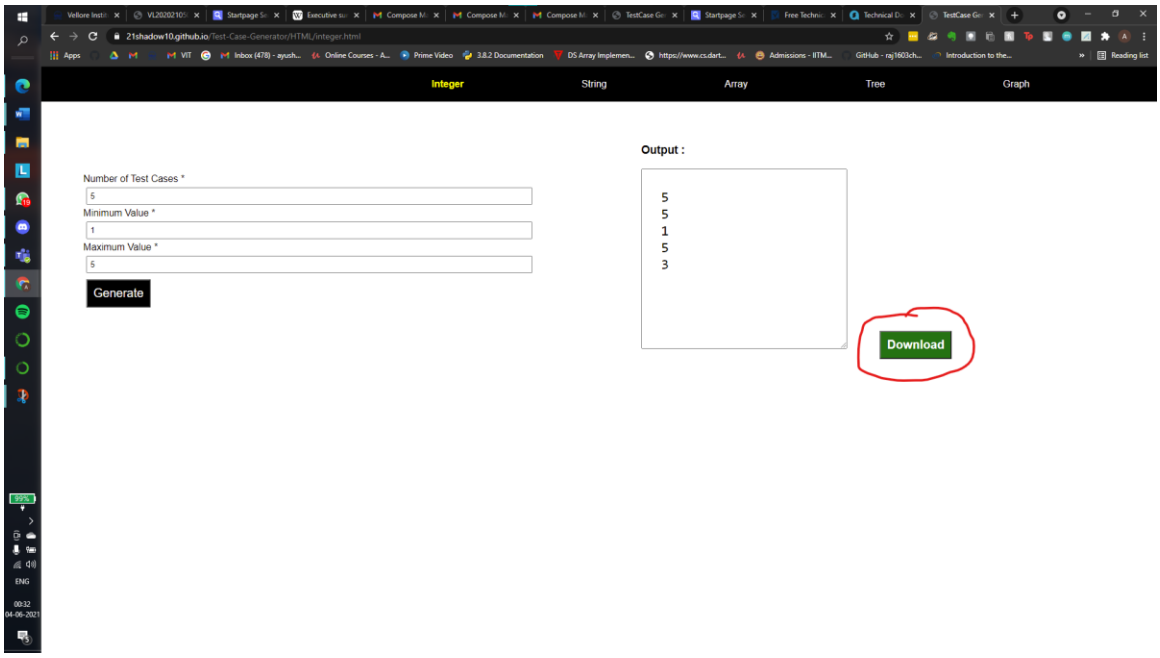
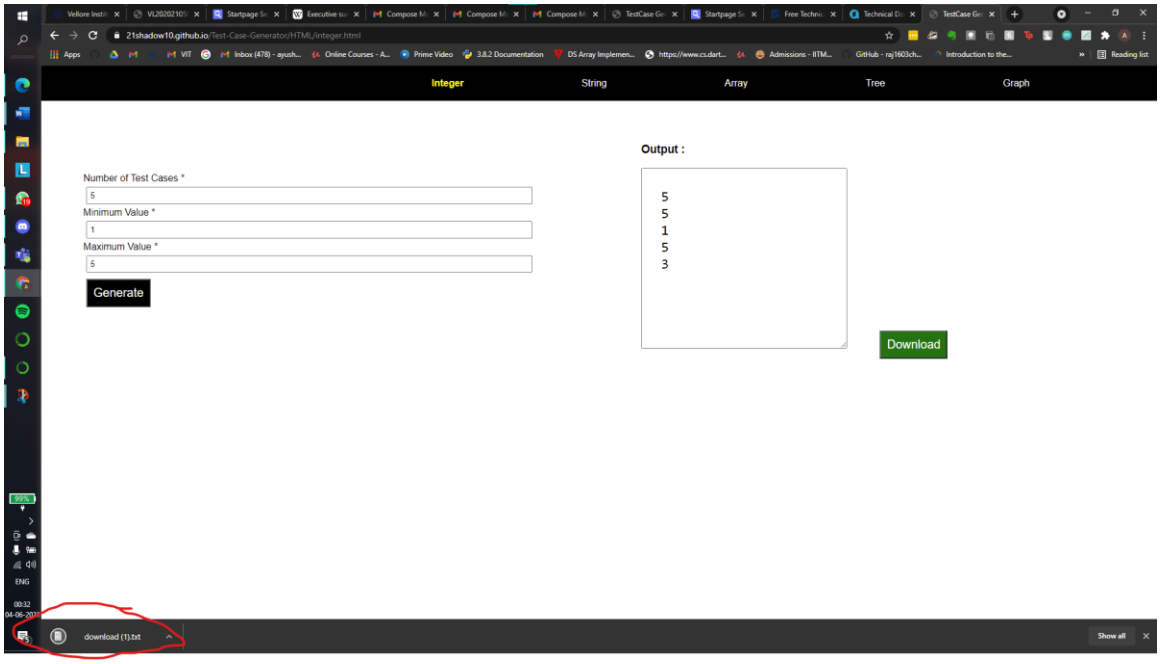


Figure 34, STEP 4: SEE OUTPUT AND ACCESS .TXT FILE FROM DOWNLOADS



9 Cost Analysis / Result and Discussion (as applicable):

9.1 Test Report

9.1.1 INTEGER

TEST CASE ID	TEST OBJECTIVE	TEST DATA			EXPECTED RESULT	ACTUAL RESULT	TEST PASS/FAIL
		No. Of test cases	Min. value	Max. value			
1.1	Integer Module Check	2	1	3	Random 2 integers generated	1,3	PASS
1.2	Integer Module Check	3	2	5	Random 3 integers generated	2,4,2	PASS
1.3	Integer Module Check	4	1	6	Random 4 integers generated	4,3,5,2	PASS
1.4	Integer Module Check	3	4	9	Random 3 integers generated	4,6,9	PASS

9.1.2 STRING

TEST CASE ID	TEST OBJECTIVE	TEST DATA			EXPECTED RESULT	ACTUAL RESULT	TEST PASS /FAIL
		No. Of test cases	Size	Characters			
2.1	String Module Check	2	4	Empty (selects from all	Random 2 strings generated	Wtpd, jclq	PASS

				the alphabets)			
2.2	String Module Check	3	5	A,p,y,t,b, e	Random 3 strings generated	Ypyte, eabtp, bbpya	PASS
2.3	String Module Check	4	6	Q,c,r,s,t, h,e,b	Random 4 strings generated	Cetbrb, brsttt, Rbbhqt, trcccs	PASS
2.4	String Module Check	3	5	R,u,b,t,d, y,q,l,m,n	Random 3 strings generated	Tyqdy, Luuuy, ubqmy	PASS

9.1.3 ARRAY

TEST CASE ID	TEST OBJECTIVE	TEST DATA				EXPECTED RESULT	ACTUAL RESULT	TEST PASS /FAIL
		No. Of test cases	Dimension	Min. Value	Max. value			
3.1	Array Module Check	2	4x2	1	9	Random 2 arrays generated	3 4 3 3 4 4 2 7 9 4 5 1 5 7 3 1	PASS
3.2	Array Module Check	1	5x2	2	8	Random 1 arrays generated	5 5 5 4 8 7 2 2 6 3	PASS
3.3	Array Module Check	2	3x3	3	7	Random 2 arrays generated	3 5 7 3 5 4 4 5 4 3 7 3 3 6 6 5 6 5	PASS
3.4	Array Module Check	3	4x1	4	12	Random 3 arrays	12 8	PASS

						generated	10 7 7 11 4 8 6 5 4 5	

9.1.4 TREE (Unweighted)

TEST CASE ID	TEST OBJECTIVE	TEST DATA		EXPECTED RESULT	ACTUAL RESULT		TEST PASS/FAIL
		No. Of test cases	No. Of nodes		Parent node	Child node	
4.1	Tree Module Check	2	4	Random 2 trees are generated	1 1 0 1 3 3	0 3 2 2 0 1	PASS
4.2	Tree Module Check	3	6	Random 3 trees are generated	5 1 1 2 4 5 0 3 1 4 4 1 4 3	4 3 5 1 0 4 2 0 3 1 5 4 2 0	PASS

					0	1	
4.3	Tree Module Check	4	4	Random 4 trees are generated	2 3 0 2 1 0 0 2 1 0 2 0	0 1 3 1 3 2 2 3 0 2 3 1	PASS

9.1.5 TREE (Weighted)

TEST CASE ID	TEST OBJECTIVE	TEST DATA				EXPECTED RESULT	ACTUAL RESULT			TEST PASS/FAIL
		No. Of test cases	No. Of nodes	Min weight	Max weight		Parent node	Child node	Weight	
5.1	Tree Module Check	2	4	2	8	Random 2 trees are generated	1 1 3 1 0 3	0 2 1 0 2 1	6 4 8 3 8 4	PASS
5.2	Tree Module Check	2	6	1	9	Random 2 trees are generated	0 3 4 1 5	2 0 3 4 1	3 1 7 7 5	PASS

							0 3 0 3 2	5 1 4 0 3	2 9 3 4 8	
5.3	Tree Module Check	3	4	2	11	Random 3 trees are generated	2 0 1 3 1 2 2 1 0	0 3 2 2 3 0 0 2 3	8 4 4 5 9 9 3 7 5	PASS

9.1.6 GRAPH (Undirected Unweighted)

TEST CASE ID	TEST OBJECTIVE	TEST DATA			EXPECTED RESULT	ACTUAL RESULT		TEST PASS/FAIL
		No. Of test cases	No. Of nodes	No. Of edges		Parent node	Child node	
6.1	Graph Module Check	2	4	6	Random 2 graphs are generated	3 4 1 4 3 2 4 1 4 1 3 4	4 2 1 1 3 2 3 2 2 3 2 4	PASS

6.2	Graph Module Check	1	3	5	Random 1 graphs are generated	1 1 3 1 3	3 2 2 1 3	PASS
6.3	Graph Module Check	2	4	7	Random 2 graphs are generated	1 3 4 1 2 3 3 2 1 4 4 3 2 1	4 2 4 3 1 4 3 3 2 1 4 3 4 3	PASS

9.1.7 GRAPH (Directed Unweighted)

TEST CASE ID	TEST OBJECTIVE	TEST DATA			EXPECTED RESULT	ACTUAL RESULT		TEST PASS/FAIL
		No. Of test cases	No. Of nodes	No. Of edges		Parent node	Child node	
7.1	Graph Module Check	2	3	5	Random 2 graphs are generated	2 3 2 3 1 3 3 2 3 2	3 1 1 3 3 2 3 1 1 3	PASS
7.2	Graph Module	1	4	7	Random 1	2 4	1 3	PASS

	Check				graphs are generated	1 1 1 3 1	1 4 2 1 3	
7.3	Graph Module Check	2	4	6	Random 2 graphs are generated	2 3 4 2 3 3 1 4 1 3 2 2	4 1 3 1 2 3 4 3 2 1 3 1	PASS

9.1.8 GRAPH (Directed Weighted)

TEST CASE ID	TEST OBJECTIVE	TEST DATA					EXPECTED RESULT	ACTUAL RESULT			TEST PASS/FAIL
		No. Of test cases	No. Of nodes	No. Of edges	Min weight	Max weight		Parent node	Child node	Weight	
8.1	Graph Module Check	2	4	6	2	8	Random 2 graphs are generated	1 2 4 1 4 1 4 2 2 2 3 3	4 1 2 3 3 1 1 1 3 4 1 3	5 8 5 7 2 5 3 5 2 3 6 5	PASS
8.2	Graph Module Check	1	6	9	0	5	Random 1 graphs are	3 4 4 6	3 1 6 1	5 5 1 0	PASS

							generated	365 2 3	52 45 6	41 44 4	
8.3	Graph Module Check	1	5	8	5	9	Random 1 graphs are generated	52 13 25 24	31 23 34 42	98 58 76 69	PASS

9.1.9 FRONTEND ELEMENTS

TEST CASE ID	TEST OBJECTIVE	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	TEST PASS/FAIL
9.1	Download button check	Click on download button	To download the generated test case in .txt file	.txt file downloaded	PASS
9.2	Generate button check	Click on generate button	To display the output in the output box	Output Displayed	PASS
9.3	Navigation bar check	Click on various data types in navigation bar	Navigate to the corresponding data type page	Corresponding data types page for inputs loaded.	PASS

9.2 Result:

We have been able to achieve our goals by creating a successful and viable test case generator which is accurate with all the values that it provides.

9.3 Discussion:

There are many ways that this generator can be used in the day-to-day activities of all the actors for this product.

We have to the best of our efforts delivered with a product that can be used in algorithms from basic DSA to even Cases such as Page ranking in web mining and essentially any methods in which these data types are used

Thank You