(1) Program / Project Title
Project Title – PROPOLOGI CALCULATOR: A TRUTH TABLE GENERATOR APPLICATION SOFTWARE WITH HISTORY FEATURE
(2) Program/Project Proponent
*Project Manager – **Alvin L. Almerol*** *Project Designer – **Justin Salvador*** *Developer – **Kyle R. Magnaye*** *System Analyst – **Alliah E. Bombais*** *Business Analyst – **Edrick L. De Villa*** *Technical Writer – **Eliana B. Ojeda***
(3) Program/Project Co-proponent and Research Assistant/s
Ms. Angie Payne MSIT, MCP
(4) Program/Project Proponent's Department/College/Office
College of Computer and Information Sciences Computer Science
(5) Background and Significance

Propositional Logic studies the logical relationships and properties that result from methods of joining or changing statements to create more complex propositions, statements, or sentences. Since studying propositional logic can be complicated, the utilization of computer programs and the concept of representing propositional logic through the truth table is an efficient way to determine the truth values of certain propositions. The use of truth tables is a great way to get started when learning how to evaluate logical expressions or prove arguments. However, it is cumbersome to formulate and construct manually due to its intricacy.

With that, various computer programs and software such as Propositional Logic Calculator existed. Basically, the functionality of the said software is the same as the typical calculator software application. However, the proponents of this project, have seen an opportunity to improve the mentioned software application. As the application has the capability to print outputs in the same manner as the truth table that is organized in rows and columns, whenever a scenario of printing of truth values is too long and the user made a mistake that cleared the printed result in the display. There is no way for the user to look back on the truth values that have been solved. Hence, lead the proponent of this project to the idea of adding a feature that users could look back at the previous solved truth values or in a nutshell, a history feature. The application provides users with a time-saving solution to the task of manually constructing truth tables for propositional expressions, a task that is both tedious and error-prone. This study was conducted primarily with the intention of helping individuals like students and anyone else who could benefit from the application's capabilities.

(6) Aims / Objectives / Hypotheses

To add a feature in a Propositional Logic Calculator. To improve the user experience of the mentioned application software. To obtain further knowledge about the implementation of discrete structures, specifically the concept of propositional logic through programming.

(7) Materials and Methods

The project proposes a calculator which showcases (and computes for) the concepts of propositional logic discussed throughout chapter 1 in this course's material *Discrete Mathematics and Its Applications*. (Rosen, 2019)

(8) Safety and Monitoring Plan (if applicable)

-Not Applicable

(9) Limitations

The proposed calculator focuses solely on working with the concepts of propositional logic, specifically in the generation of truth tables. These concepts include:

1. Negation
2. Conjunction
3. Disjunction
4. Conditional Connective
5. Biconditional Logical Connective

By extension, one can perform operations which showcases advanced concepts such as:

- Finding the converse, inverse, and contrapositive of a propositional statement.
- Forming compound propositions.

The representation of a prepositional variable is denoted by the letters given within the calculator, which span from 'p' to 'v'. These letters can be entered into the calculator to form a word that functions as one variable, but it is not intended to natively support full string cases.

The cases which can be entered into the calculator are virtually limitless; constrained only by the number of resources that Python can allocate for the calculator. The calculator can display a huge, tabulated result of indefinite length should the propositional case entered be complex. One can also extract the result from the calculator onto any text-editing program.

(10) Ethical Considerations

A. Informed Consent (Applies to studies using human subjects) --Not Applicable

B. Risks and Side Effects (Applies to studies using human subjects) --Not Applicable

C. Benefits to Subjects (Applies to studies using human subjects) --Not Applicable

D. Costs to Subject (Applies to studies using human subjects) --Not Applicable

E. Compensation to Subject (Applies to studies using human subjects) --Not Applicable

F. Provisions for vulnerable subjects (Applies to studies using human subjects) --Not Applicable

G. Subject Privacy and Data Confidentiality (Applies to studies using human subjects)

--Not Applicable

(11) Plan for Dissemination of Findings

As part of compliance with the Discrete Structures 2 course's midterm requirements for all BSCS 2–4 students, the findings will be disseminated in the form of an application and research presentation. After scrutinization and approval by this course's instructor, Ms. Angie Payne, by giving a passing mark for this project proposal, the following steps for dissemination will be enacted:

1. **Uploading an open-source repository to GitHub.**

An open-source repository of *Propologicalculator* will be uploaded to GitHub. This allows developers and other researchers to download, use, and potentially contribute to the development of the application. The uploaded repo will include a readme file and any other dependencies needed for the application to work in a Windows OS environment – which is what was used for the development of this application.

2. **Used as a credential when presenting a Curriculum Vitae/Resume.**

As the program was developed for a major course requirement (Discrete Structures 2), all students involved in the development of this program may cite *Propologicalulator* as one of their qualifications and/or notable achievements. As it is uploaded to GitHub, one can easily inspect and assess the program's integrity and presentation at will.

3. Publication to a public journal hosting site.

This project proposal could be uploaded to a public journal hosting site, such as *Google Scholar* or *ResearchGate*, which may include a document that details the algorithms used to calculate propositional logic equivalences. This makes the project accessible to other students or researchers bearing an identical study and allows them to use this proposal as a resource for their research. Furthermore, as the code is open source, it is possible for them to download and/or contribute to the expansion of this project.

4. Collaboration with other students/used as future research.

The program and proposal itself can be further disseminated by means of collaborating with other students should an idea involving the aspect of propositional logic. As the computer science course leans toward computation, it is likely that this project may remain relevant and be expanded upon by the members of this proposal later.

(12) References

Amos, D. (2022, July 18). Python GUI programming with Tkinter. Real Python. Retrieved December 16, 2022, from https://realpython.com/python-gui-tkinter/

Chauhan Follow Web Developer & Content Writer at Student, O. (n.d.). Simple calculator flowchart. Share and Discover Knowledge on SlideShare. Retrieved December 16, 2022, from https://www.slideshare.net/OmiChauhan/simple-calculator-flowchart

The Collaborative Interface Design Tool. Figma. (n.d.). Retrieved December 16, 2022, from https://www.figma.com/

Klement, K. C. (n.d.). Propositional Logic. Internet encyclopedia of philosophy. Retrieved December 16, 2022, from https://iep.utm.edu/prop-log/

Leroy, K. (n.d.). Pseudocode Examples for Functions. OpenStax CNX. Retrieved December 16, 2022, from https://cnx.org/contents/MDgA8wfz@22.2:6Uvbhb3A@7/Pseudocode-Examples-for-Functions

ParthJadhav. (2022, September 27). Tkinter-designer/instructions.md at master · Parthjadhav/Tkinter-designer. GitHub. Retrieved December 16, 2022, from https://github.com/ParthJadhav/Tkinter-Designer/blob/master/docs/instructions.md#fformatting-1

Rosen, K. H. (2019). Discrete mathematics and its applications. McGraw-Hill.

Team, E. (2022, December 13). C program for a simple Calculator - C examples. Notesformsc. Retrieved December 16, 2022, from https://notesformsc.org/c-simple-calculator/

(13) Appendices

A. Flowchart

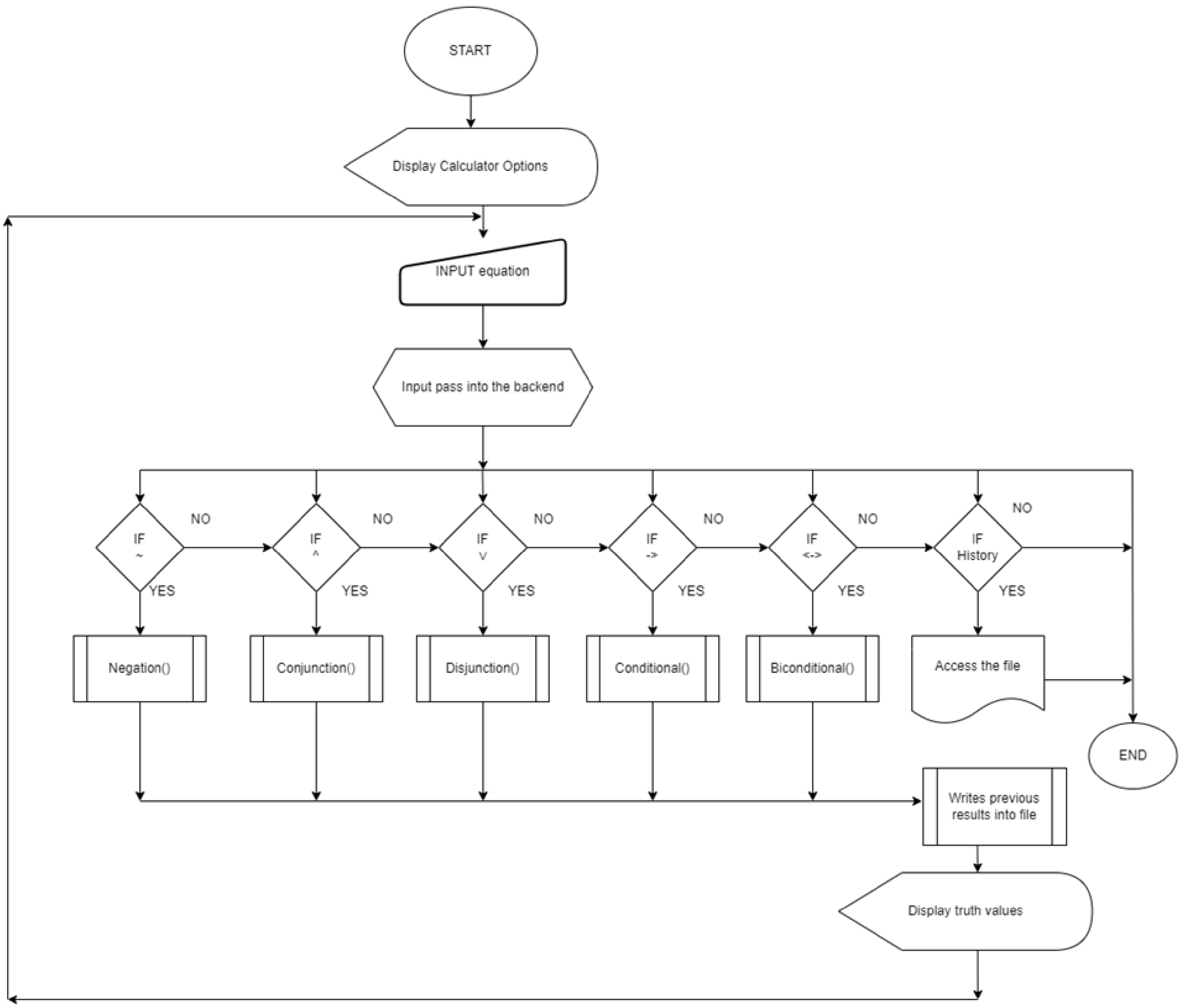

Chart 1: Program flowchart

Above is the PROPOLOGI CALCULATOR flowchart that shows how the data flows throughout the program. As the program starts operating, it will allow the user to manually input the data just as how a classic calculator works. Then, the data will be process by using multiple decision points. Each point will then utilize the different predefined processes based on the type of data that was input. The program will end when all the calculated data was processed and displayed, as well as when the user has done accessing the history.

B. Pseudocode

```
ALGORITHM Propositional Logic Calculator
FUNCTION isWellFormed(P)
  bracketLevel = 0
  FOR c IN P DO
    IF c == "(" THEN
      bracketLevel += 1
    END IF
    IF c == ")" THEN
      IF bracketLevel == 0 THEN
        RETURN False
      END IF
      bracketLevel -= 1
    END IF
  END FOR
  RETURN bracketLevel == 0
END FUNCTION

FUNCTION parseNegation(P, truthValues)
  RETURN NOT parseProposition(P, truthValues)
END FUNCTION

FUNCTION parseConjunction(P, Q, truthValues)
  RETURN parseProposition(P, truthValues) AND parseProposition(Q, truthValues)
END FUNCTION

FUNCTION parseDisjunction(P, Q, truthValues)
  RETURN parseProposition(P, truthValues) OR parseProposition(Q, truthValues)
END FUNCTION

FUNCTION parseConditional(P, Q, truthValues)
  RETURN (NOT parseProposition(P, truthValues)) OR parseProposition(Q, truthValues)
END FUNCTION

FUNCTION parseBiconditional(P, Q, truthValues)
  RETURN parseProposition(P, truthValues) == parseProposition(Q, truthValues)
END FUNCTION

FUNCTION parseProposition(P, truthValues)
  P = P.replace(" ", "")
  IF NOT isWellFormed(P) THEN
    RETURN "Error"
  END IF
  WHILE P[0] == "(" AND P[-1] == ")" AND isWellFormed(P[1:len(P) - 1]) DO
    P = P[1:len(P) - 1]
  END WHILE
  IF len(P) == 1 THEN
    RETURN truthValues[P]
  END IF
```

```
bracketLevel = 0
FOR i = len(P)-1 DOWN TO 0 DO
  IF P[i] == "(" THEN
    bracketLevel += 1
  END IF
  IF P[i] == ")" THEN
    bracketLevel -= 1
  END IF
  IF P[i] == "→" AND bracketLevel == 0 THEN
    RETURN parseConditional(P[0:i], P[i + 1:], truthValues)
  END IF
  IF P[i] == "↔" AND bracketLevel == 0 THEN
    RETURN parseBiconditional(P[0:i], P[i + 1:], truthValues)
  END IF
END FOR
bracketLevel = 0
FOR i = len(P)-1 DOWN TO 0 DO
  IF P[i] == "(" THEN
    bracketLevel += 1
  END IF
  IF P[i] == ")" THEN
    bracketLevel -= 1
  END IF
  IF P[i] == "∨" AND bracketLevel == 0 THEN
    RETURN parseDisjunction(P[0:i], P[i + 1:], truthValues)
  END IF
END FOR
bracketLevel = 0
FOR i = len(P)-1 DOWN TO 0 DO
  IF P[i] == "(" THEN
    bracketLevel += 1
  END IF
  IF P[i] == ")" THEN
    bracketLevel -= 1
  END IF
  IF P[i] == "∧" AND bracketLevel == 0 THEN
    RETURN parseConjunction(P[0:i], P[i + 1:], truthValues)
  END IF
END FOR
bracketLevel = 0
FOR i = len(P)-1 DOWN TO 0 DO
  IF P[i] == "(" THEN
    bracketLevel += 1
  END IF
  IF P[i] == ")" THEN
    bracketLevel -= 1
  END IF
  IF P[i] == "¬" AND bracketLevel == 0 THEN
    RETURN parseNegation(P[i + 1:], truthValues)
  END IF
```

```
    END FOR
    RETURN "Error"
END FUNCTION

FUNCTION writeTruthTable(P)
    truthValues = {}

    FOR i FROM 0 TO length of P
        IF P[i] is a letter from "A" to "Z"
            SET truthValues[P[i]] = True
        END IF
    END FOR

    SET output to a new StringIO object
    SET sys.stdout to output

    FOR EACH statement IN keys of truthValues
        PRINT statement, " | "
    END FOR
    PRINT P

    FOR EACH truthValue IN values of truthValues
        IF truthValue is True
            PRINT "T", " | "
        ELSE
            PRINT "F", " | "
        END IF
    END FOR

    IF parseProposition(P, truthValues) is True
        PRINT "T"
    ELSE
        PRINT "F"
    END IF

    SET j to length of values of truthValues - 1

    WHILE True in values of truthValues
        SET variable to the key at index j in keys of truthValues
        SET truthValues[variable] to not truthValues[variable]

        IF truthValues[variable] is False
            FOR EACH truthValue IN values of truthValues
                IF truthValue is True
                    PRINT "T", " | "
                ELSE
                    PRINT "F", " | "
                END IF
            END FOR
```

```
        IF parseProposition(P, truthValues) is True
          PRINT "T"
        ELSE
          PRINT "F"
        END IF

        SET j to length of values of truthValues - 1
      ELSE
        SET j to j - 1
      END IF
    END WHILE

    SET sys.stdout to the original stdout object
    RETURN the value of output as a string
END FUNCTION

FUNCTION Conjunction()
    Press the "V" key
    Release the "V" key
END FUNCTION

FUNCTION Conjuction()
    Press the "V" key
    Release the "V" key
END FUNCTION

FUNCTION Disjunction()
    Press the "Λ" key
    Release the "Λ" key
END FUNCTION

FUNCTION Conditional()
    Press the "→" key
    Release the "→" key
END FUNCTION

FUNCTION Biconditional()
    Press the "↔" key
    Release the "↔" key
END FUNCTION

FUNCTION Negation()
    Press the "¬" key
    Release the "¬" key
END FUNCTION

FUNCTION OpenP()
    Press the "(" key
    Release the "(" key
END FUNCTION
```

```
FUNCTION CloseP()
  Press the ")" key
  Release the ")" key
END FUNCTION

FUNCTION LetterP()
  Press the "P" key
  Release the "P" key
END FUNCTION

FUNCTION LetterQ()
  Press the "Q" key
  Release the "Q" key
END FUNCTION

FUNCTION LetterR()
  Press the "R" key
  Release the "R" key
END FUNCTION

FUNCTION LetterS()
  Press the "S" key
  Release the "S" key
END FUNCTION

FUNCTION LetterT()
  Press the "T" key
  Release the "T" key
END FUNCTION

FUNCTION LetterU()
  Press the "U" key
  Release the "U" key
END FUNCTION

FUNCTION LetterV()
  Press the "V" key
  Release the "V" key
END FUNCTION

FUNCTION Backspace()
  Press the Backspace key
  Release the Backspace key
END FUNCTION

CLASS ConsoleGUI
  FUNCTION __init__(master)
    // Initialize the Console GUI
    self.master = master
```

```
        self.frame = create a frame
        self.frame is packed to the top
        self.command_entry = create an entry widget for user input
        self.command_entry is packed to the bottom, filling the remaining space
        self.command_entry is set to execute the command when 'Enter' is pressed
        the focus is set to the command entry widget
        self.console_output = create a text widget to display output
        the widget is initially set to 'disabled'
        self.console_output is packed to the left, filling the remaining space

    FUNCTION execute_command()
        // Execute the user's command and display output
        get the command from the command_entry widget
        use the writeTruthTable() function to generate output based on the command
        set the console_output widget to 'normal' mode to enable writing to it
        add the output to the console_output widget
        set the console_output widget back to 'disabled' mode to prevent user input
        scroll the console_output widget to the end

    FUNCTION save_console_text()
        // Save console output to a file
        get the command from the command_entry widget
        delete the command from the command_entry widget
        use the writeTruthTable() function to generate output based on the command
        open a dialog box to choose a filename to save the output to
        if a filename is selected, create a new file and write the output to it

    FUNCTION clear_console()
        // Clear the console output and command entry widgets
        set the console_output widget to 'normal' mode to enable writing to it
        delete all text in the console_output widget
        set the console_output widget back to 'disabled' mode to prevent user input
        delete the command from the command_entry widget

    FUNCTION load_file(filename)
        // Load contents of file and return them
        open the file with the given filename
        read the contents of the file and return them

    FUNCTION load_file_handler()
        // Open a dialog box to choose a file to load and display contents in console output widget
        open a dialog box to choose a file to load
        if a file is selected, load its contents and display them in the console_output widget

    FUNCTION run()
        // Start the main loop for the program
        start the main loop for the program

Class PropologicalGUI:
    Function __init__(self, master):
```

Set self.master to master
Set the title of self.master to "Propological"
Set the minimum and maximum size of self.master to 506 x 600 pixels
Create an instance of the ConsoleGUI class and assign it to self.console

Create 15 buttons with the following parameters:
Text: "V", "Λ", "→", "↔", "¬", "(", ")", "P", "Q", "R", "S", "T", "U", "V", "⌫"
Font: "Cambria" size 20
Width: 6
Foreground (text) color: white
Background color: maroon
Commands: Conjunction, Disjunction, Conditional, Biconditional, Negation, OpenP, CloseP, LetterP, LetterQ, LetterR, LetterS, LetterT, LetterU, LetterV, Backspace
Place the buttons at the following (x, y) coordinates:
(20, 440), (140, 440), (260, 440), (380, 440), (20, 520), (140, 520), (260, 520), (20, 280), (140, 280), (260, 280), (380, 280), (20, 360), (140, 360), (260, 360), (380, 360)

Create a "Done" button with the following parameters:
Text: "Done"
Font: "Cambria" size 20
Width: 6
Foreground (text) color: white
Background color: maroon
Command: execute_command method of self.console
Place the button at (380, 520)

Create a "Clear" button with the following parameters:
Text: "Clear"
Font: "Cambria" size 20
Width: 6
Foreground (text) color: white
Background color: maroon
Command: clear_console method of self.console
Place the button at (380, 130)

Create a "Save" button with the following parameters:
Text: "Save"
Font: "Cambria" size 20
Width: 6
Foreground (text) color: white
Background color: maroon
Command: save_console_text method of self.console
Place the button at (380, 70)

Create a "Load" button with the following parameters:
Text: "Load"
Font: "Cambria" size 20
Width: 6
Foreground (text) color: white
Background color: maroon

Command: load_file_handler method of self.console
Place the button at (380, 10)

if __name__ == '__main__':
Create a Tkinter root window and assign it to root
Create an instance of the PropologicalGUI class and pass root as an argument, assign it to app
Call the run method of self.console
END Propositional Logic Calculator

C. Logo

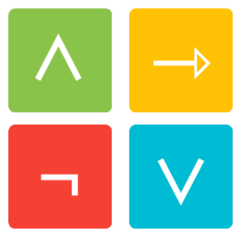

Figure 1(left) & 2(right): PROPOLOGI-CALCULATOR Logos

The PROPOLOGI-CALCULATOR Application has two logos, the main logo, and the mini logo. The first logo with the name of the application is the main logo. The logo has a minimalist look but without lacking details. It has this sleek look that adapts the classic calculator logos yet has its own vibe and colors with the propositional logic's logical symbols. This logo will welcome the users as its begins to run. The second logo on the right is the mini logo. This will be displayed at the top-left area of the application. Even though it is just the simpler version of the main logo, it carries the main details that will highlight the application's purpose.

D. Work Breakdown Structure

WORK BREAKDOWN STRUCTURE
Project Title: PROPOLOGI CALCULATOR: A TRUTH TABLE GENERATOR APPLICATION SOFTWARE WITH HISTORY FEATURE

1.0 REQUIREMENTS	
	1.1 Problem statement
	1.2 Objectives
	1.3 System Requirements
	1.4 Project/Process Requirements
2.0 PROBLEM PLANNING	
	2.1 Work breakdown structures
	2.2 Linear responsibility chart
	2.3 Gantt chart
3.0 SYSTEM DESIGN	
	3.1 Identify inputs and outputs
	3.2 Flowchart
	3.3 Pseudocode

4.0 IMPLEMENTATION
- 4.1 GUI design
  - 4.1.1 Software frames, color, size, text
- 4.2 UI UX design
  - 4.2.1 Logo Designing
  - 4.2.2 Button placements
  - 4.2.3 Pictures and icon placement
- 4.3 Back-end functionality
  - 4.3.1 Input for propositional statements from user
  - 4.3.2 Propositional Statements Parsing
  - 4.3.3 Truth Table generation implementation
  - 4.3.4 File Handling for presets and save function

5.0 TESTING / VERIFICATION
- 5.1 Debug
  - 5.1.1 Assess software UI UX design
  - 5.1.2 Debug back-end errors
- 5.2 Animate
- 5.3 Test Limitations

6.0 DOCUMENTATION
- 6.1 Document system diagram
- 6.2 Document system implementation
- 6.3 Document groupmates experience
- 6.4 Instruction Manual

7.0 DEPLOYMENT
- 7.1 Powerpoint Presentation
- 7.2 Program walkthrough

8.0 CONCLUSIONS AND RECOMMENDATIONS
- 8.1 Conclusions
- 8.2 Recommendations

Chart 2: Work Breakdown Structure (WBS)

Chart 2 shows the Work Breakdown Structure of the whole project. The PROPOLOGI CALCULATOR is visually, hierarchically, and deliverable-focused disassembled. It enables the researchers to deconstruct the scope of their projects and see all the tasks necessary to finish them. From the requirements needed to start the application, down to the drawing of conclusions, every part of the process is included and shown in the work breakdown structure.

E. Fibonacci Agile Estimation

FIBONACCI AGILE ESTIMATION

WBS	ACTIVITY	Complexity (1 2 3 5 8)	Duration (1 2 3 5 8)	Average (60-40)	Earned Points	Weight
1	REQUIREMENTS					
1.1	Problem statement	1	1	1	0.01295336788	1.295336788
1.2	Objectives	1	1	1	0.01295336788	1.295336788
1.3	System Requirements	1	1	1	0.01295336788	1.295336788
1.4	Project/Process Requirements	1	1	1	0.01295336788	1.295336788
2	PROBLEM PLANNING					
2.1	Work breakdown structures	3	1	2.2	0.02849740933	2.849740933
2.2	Linear responsibility chart	3	1	2.2	0.02849740933	2.849740933
2.3	Gantt chart	3	1	2.2	0.02849740933	2.849740933
3	SYSTEM DESIGN					
3.1	Identify inputs and outputs	2	1	1.6	0.0207253886	2.07253886
3.2	Flowchart	3	3	3	0.03886010363	3.886010363
3.3	Pseudocode	3	3	3	0.03886010363	3.886010363

4	IMPLEMENTATION					
4.1	GUI design					
4.1.1	Software frames, color, size, text	5	2	3.8	0.04922279793	4.922279793
4.2	UI UX design					
4.2.1	Logo Design	3	2	2.6	0.03367875648	3.367875648
4.2.2	Button placements	2	1	1.6	0.0207253886	2.07253886
4.2.3	Pictures and icon placements	2	2	2	0.02590673575	2.590673575
4.3	Back-end functionality					
4.3.1	Input for propositional statements from user	3	5	3.8	0.04922279793	4.922279793
4.3.2	Propositional Statements Parsing	8	5	6.8	0.08808290155	8.808290155
4.3.3	Truth Table generation implementation	8	5	6.8	0.08808290155	8.808290155
4.3.4	File Handling for presets and save function	5	5	5	0.06476683938	6.476683938
5	TESTING / VERIFICATION					
5.1	Debug					
5.1.1	Assess software UI UX design	3	2	2.6	0.03367875648	3.367875648
5.1.2	Debug back-end errors	5	2	3.8	0.04922279793	4.922279793
5.2	Animate	5	3	4.2	0.05440414508	5.440414508
5.3	Test Limitations	3	1	2.2	0.02849740933	2.849740933
6	DOCUMENTATION					
6.1	Document system diagram	3	3	3	0.03886010363	3.886010363
6.2	Document system implementation	3	3	3	0.03886010363	3.886010363
6.3	Document groupmates experience	1	3	1.8	0.02331606218	2.331606218
6.4	Instruction Manual	2	2	2	0.02590673575	2.590673575
7	DEPLOYMENT					
7.1	Powerpoint Presentation	2	2	2	0.02590673575	2.590673575
7.2	Program walkthrough	2	2	2	0.02590673575	2.590673575
8	CONCLUSIONS AND RECOMMENDATIONS					
8.1	Conclusions	5	8	6.2	0.08031088083	8.031088083
8.2	Recommendations	5	8	6.2	0.08031088083	8.031088083
			Total	77.2		

Chart 3: Fibonacci Agile Estimation

The Fibonacci Agile estimation is a method of estimating the amount of work required to finish a proposed project. Sequence diagrams are a common unit used by groups and teams to assess their tasks. The more points given; the more effort the team estimates the assignment will require. Chart 3 presents how each task included in the Work Breakdown Structure using the Fibonacci scale with one (1) being the easiest and eight (8) as the highest. Each task was assessed based on their complexity and duration. The team utilized the Microsoft Excel software to encode and compute its average together with the earned points and its weight on the whole project.

F. Linear Responsibility Chart

LINEAR RESPONSIBILITY CHART (LRC)

WBS	ACTIVITY	MEMBERS					
1	REQUIREMENTS	ALLIAH	ALVIN	EDRICK	ELLIANA	JUSTIN	KYLE
1.1	Problem statement	R	R	R	R	R	R
1.2	Objectives	I	A	C	R	C	I
1.3	System Requirements	I	R	I	C	I	I
1.4	Project/Process Requirements	I	A	I	I	R	I
2	PROBLEM PLANNING						
2.1	Work breakdown structures	R	R	R	R	R	R
2.2	Linear responsibility chart	R	C	C	C	C	C
2.3	Gantt chart	C	A	R	C	I	I

3	**SYSTEM DESIGN**						
3.1	Identify inputs and outputs	I	R	I	C	I	I
3.2	Flowchart	I	R	I	I	I	C
3.3	Pseudocode	I	R	I	I	I	C
4	**IMPLEMENTATION**						
4.1	GUI design						
4.1.1	Software frames, color, size, text	R	I	I	I	I	R
4.2	UI UX design						
4.2.1	Button placements	R	I	I	I	I	R
4.2.2	Pictures and icon placements	R	I	I	I	I	R
4.3	Back-end functionality						
4.3.1	Input for propositional statements from user	C	I	I	I	I	R
4.3.2	Propositional Statements Parsing	C	I	I	I	I	R
4.3.3	Truth Table generation implementation	C	I	I	I	I	R
4.3.4	File Handling for presets and save function	C	I	I	I	I	R
5	**TESTING / VERIFICATION**						
5.1	Debug						
5.1.1	Assess software UI UX design	R	I	I	I	I	R
5.1.2	Debug back-end errors	C	I	I	I	I	R
5.2	Animate	R	I	I	I	I	R
5.3	Test Limitations	R	I	I	I	I	R
6	**DOCUMENTATION**						
6.1	Document system diagram	A	R	C	C	C	A
6.2	Document system implementation	A	C	C	R	C	A
6.3	Document groupmates experience	I	A	I	I	I	I
6.4	Instruction Manual	I	A	I	I	I	R
7	**DEPLOYMENT**						
7.1	Powerpoint Presentation	R	A	I	I	I	I
7.2	Program walkthrough	R	R	R	R	R	R

LEGEND:
R = *RESPONSIBLE*
A = *ACCOUNTABLE*
C = *CONSULTANT*
I = *INFORMEE*

Chart 4: Linear Responsibility Chart (LRC)

By further assessing the tasks after the team performed the Work Breakdown Structure, and after each task was assessed using the Fibonacci Agile Estimation—the researchers make use of the Linear Responsibility Chart to determine the roles each member needs to perform. Due to the overlapping involvements in project management, the LRC is what distinguishes the project members and demonstrates the authority and responsibility relationships among the group. It establishes a connection between project tasks or activities and the accountable party, ensuring that the project is carried out effectively to meet its goals within the allotted time. The chart 4 shows the tasks and roles each member was assigned to: R as responsible, A for accountable, C for consultant, and I being the informee.

G. Three-Point Estimate

WBS	ACTIVITY	Optimistic	Most Likely	Pessimistic	3-Point Estimate
1	REQUIREMENTS				
1.1	Problem statement	1	1	1	1
1.2	Objectives	1	1	1	1
1.3	System Requirements	1	1	2	1.166666667
1.4	Project/Process Requirements	1	1	2	1.166666667
2	PROBLEM PLANNING				
2.1	Work breakdown structures	1	2	3	2
2.2	Linear responsibility chart	1	2	3	2
2.3	Gantt chart	1	2	3	2
3	SYSTEM DESIGN				
3.1	Identify inputs and outputs	1	2	3	2
3.2	Flowchart	2	3	5	3.166666667
3.3	Pseudocode	2	3	5	3.166666667
4	IMPLEMENTATION				
4.1	GUI design				
4.1.1	Software frames, color, size, text	1	2	3	2
4.2	UI UX design				
4.2.1	Logo Design	1	1	3	1.333333333
4.2.2	Button placements	1	1	2	1.166666667
4.2.3	Pictures and icon placements	1	1	3	1.333333333
4.3	Back-end functionality				
4.3.1	Input for propositional statements from user	2	4	7	4.166666667
4.3.2	Propositional Statements Parsing	2	4	7	4.166666667
4.3.3	Truth Table generation implementation	2	5	7	4.833333333
4.3.4	File Handling for presets and save function	3	5	7	5
5	TESTING / VERIFICATION				
5.1	Debug				
5.1.1	Assess software UI UX design	1	2	4	2.166666667
5.1.2	Debug back-end errors	1	2	3	2
5.2	Animate	2	3	5	3.166666667
5.3	Test Limitations	1	2	3	2
6	DOCUMENTATION				
6.1	Document system diagram	2	3	6	3.333333333
6.2	Document system implementation	2	3	6	3.333333333
6.3	Document groupmates experience	2	3	6	3.333333333
6.4	Instruction Manual	1	2	3	2
7	DEPLOYMENT				
7.1	Powerpoint Presentation	1	2	3	2
7.2	Program walkthrough	1	2	3	2
8	CONCLUSIONS AND RECOMMENDATIONS				
8.1	Conclusions	10	16	20	15.66666667
8.2	Recommendations	10	16	20	15.66666667
				TOTAL	99.33333333

Chart 5: Three-Point Estimate

The Three-Point Estimate is used to calculate the likely outcomes or costs of anticipated events. The group applied this method to generate reasonable predictions and determine an expected value using the limited data from a detailed project evaluation. It is considered as *'Optimistic'* when you expect that everything will work out and that there aren't likely to be any obstacles. *'Pessimistic'* when you think that nothing will go as predicted and that the end will be more disastrous than anticipated. *'Most Likely'* is the estimate in the middle of the two extremes. It is the benefit you receive when you have reached a compromise and foresee that some aspects of a future event will either go well or poorly. The unit that the group used is weeks.

H. Gantt Chart

Chart 6: Project Gantt Chart

To visually display the list of activities with respect to time, the group utilized the Gantt chart. The list of the activities is located on the chart's left side, and a suitable time scale is located along the top —the team has used a weekly time scale ranging from November 5, 2022, to February 18, 2023. A bar is used to symbolize each activity, and the position and length of the bar correspond to the activity's beginning, middle, and finish dates. Below is the legend that represents the different tiles in the Gantt chart:

LEGENDS
Divider
Midterm and Final Presentation
Work task
Summary Task

Chart 7: Gantt Chart Legends

The light, green-colored tiles serve as the divider among the major activities in the project. The teal-colored tiles represent the two major deadlines there are: Midterm and Final Presentation. The light red tiles are the work tasks with respect to the number of weeks that the group took to perform. Lastly, the primary red represents the summary task. It is the cumulation of weeks that the sub activities took to be finished.

	(14) Prepared by	(15) Endorsed by
Signature		
Name of proponent	Alliah E. Bombais Alvin L. Almerol Edrick L. De Villa Eliana B. Ojeda Justin Salvador Kyle R. Magnaye	Ms. Angie Payne, MSIT, MCP
Designation/position	Student	Faculty-In-Charge
Date	02/18/2023	