Google Summer of Code

sugarlabs

GSoC 2025 Project Proposal for

Developing Interactive Math Game Activities for SugarLabs

(Duration: 350 hours)

DEV MONI

NATIONAL INSTITUTE OF TECHNOLOGY, AGARTALA

+91 9548264153

devmoni97@gmail.com

Dev Moni

Devmoni

Abstract

Mathematics plays a crucial role in shaping a child's cognitive abilities, fostering logical reasoning, and enhancing problem-solving skills. However, traditional methods of teaching often struggle to keep young learners engaged, making it essential to introduce interactive and intuitive learning experiences.

This project aims to develop a set of engaging math activities for the Sugar Learning Platform, leveraging gamification and visual learning to make mathematical concepts more accessible and enjoyable. These activities will focus on fundamental topics, ensuring an intuitive and child-friendly interface while integrating adaptive difficulty levels to accommodate different learning paces.

By following a structured development approach, I will design, implement, and refine these activities based on feedback from mentors and potential users. Thorough testing will be conducted to ensure a seamless user experience, making the activities both educational and entertaining.

As an active volunteer in the WeCan initiative, where I teach underprivileged children, I deeply resonate with Sugar Labs' mission. Through this project, I hope to contribute meaningfully to open education by making quality learning tools more engaging, inclusive, and effective for young learners worldwide.

Contents

1 About Me

The Student

Name: **Dev Moni**

E-mail: `devmoni97@gmail.com`

Github: `https://github.com/Devmoni`

Location: **India**

Time Zone: **IST (UTC+5:30)**

The Institute

University National Institute of Technology Agartala

Year Senior

Degree Bachelor of Technology

1.1 Programming Experience

I am Dev Moni, currently pursuing a Bachelor's degree in Civil Engineering, but my passion lies in software development, open-source contributions, and problem-solving. I have hands-on experience in full-stack development and actively contribute to open-source projects. My technical expertise includes C++, Python, JavaScript, React, and PostgreSQL, and I have previously interned at UNBIND, where I worked on backend applications and scalable systems.

1.2 My Motivation

I am deeply passionate about using technology to make education more accessible. As a member of the **WeCan Club** (An initiative of students from NIT Agartala), I have been actively involved in teaching underprivileged children in my college area. This experience has given me first-hand insight into the challenges faced by students with limited resources.

SugarLabs' mission to empower learning through open-source technology strongly resonates with me. I believe that well-designed educational tools can bridge the learning gap and create opportunities for students worldwide. Contributing to Sugar Labs would allow me to leverage my technical skills to develop impactful solutions that enhance accessibility and engagement in education.

1.3 My Projects and Experience

Multilingual FAQ Service - A full-stack web application for managing FAQs in multiple languages, built with *PostgreSQL* and *JavaScript*.

UNBIND Internship - Worked on backend development, enhancing system scalability using *JavaScript*, *Python*, and *React*.

1.4 Questionnaire

Why are you interested in working on Sugar Labs?

Sugar Labs' mission to make learning accessible through open-source technology aligns with my passion for **education and open-source development**. As a volunteer at **WeCan**, I teach underprivileged children, and I've seen firsthand how interactive learning boosts engagement. Sugar's commitment to education and community-driven development inspires me, and I am excited to contribute to a platform that makes learning more engaging for kids.

Why are you interested in this project idea?

Mathematics is crucial for developing logical reasoning, which helps children understand other subjects like **physics and computer science**. By creating interactive math activities, I aim to make math more intuitive and enjoyable. This project is a great opportunity to apply my technical skills while improving problem-solving in education.

What about your skills makes you well-suited for this project?

I have experience in **C++, Python, JavaScript, React, and GTK** and I actively contribute to open-source projects, including Sugar Labs. I have worked extensively with Sugar activities developed in Python and have a solid understanding of the platform's core development practices. I have contributed to porting several activities from Python 2 to Python 3, and in some cases, also updated them to support **GTK+3**. These contributions have given me hands-on experience with the technical stack used in Sugar and a deeper insight into its educational goals. My technical skills, along with a strong interest in interactive learning tools, make me well-suited for this project.

Do you have any other commitments during the program period?

I have **no major commitments** and will be fully dedicated to GSoC, ensuring I can focus on delivering quality contributions to Sugar Labs.

1.5 Contributions

I started using Sugar Activities in December and made my first contribution in December to Sugar www repository. I have been constantly contributing and learning from the community ever since.

Pull Request	Status	Description
#6	Open	**sugarlabs/jigsaw-puzzle-branch** WIP: Port to Gtk3+ and Python3
#17	Open	**sugarlabs/reflect** Fix GI warnings, update deprecated GObject usage
#7	Open	**sugarlabs/slider-puzzle-branch** Port to Python 3.
#12	Open	**sugarlabs/classroombroadcast** Port activity from Python 2 to Python 3 (Fixes 9)
#12	Merged	**sugarlabs/chart** Change setup.py
#15	Merged	**sugarlabs/moon-activity** Update Moon activity data for 2025 and future years
#11	Open	**sugarlabs/starchart** Port to Python 3
#37	Open	**sugarlabs/connect-the-dots-activity** fix(32): Ensure stop button functions correctly
#35	Merged	**sugarlabs/connect-the-dots-activity** Fix 33: Align activity icon in toolbar
#643	Merged	**sugarlabs/www** Update About Section UI and Fix Google Summer of Code Link
#612	Merged	**sugarlabs/www** Float Donation Button to Enhance Banner Visibility
#82	Merged	**sugarlabs/aslo-v4** Fix: navigation bar layout and functionality
#605	Merged	**sugarlabs/www** Float icon back to top
#592	Merged	**sugarlabs/www** enhancement for service section

2 Project Vision and Structure

In this section, I will outline my vision for the design and development of new math activities within the Sugar ecosystem. These activities will be crafted to enhance children's logical thinking, problem-solving skills, and mathematical intuition. My approach will focus on intuitive gameplay mechanics, interactive user experiences, and seamless integration with Sugar's existing framework.

Each activity will be designed with **engagement, education, and accessibility** in mind, ensuring that students benefit from an enriching learning experience.

I expect this structure to be significantly refined and improved with guidance from my mentor, incorporating feedback to enhance usability and educational value.

Development Approach

I have a strong foundation in **Sugar activity development, Python, and GTK**, which will serve as the backbone for building these new math-based learning tools. To ensure a structured and efficient development process, I will take an incremental approach—starting with a basic activity structure and progressively refining the features.

I will use **SugarLabs/HelloWorld** activity as a reference point to set up the framework for all math games. This will provide a reliable foundation, ensuring consistency across activities. By implementing features step by step, I will be able to test mechanics early, incorporate iterative feedback, and make necessary improvements to create an optimal learning experience.

Project Scope and Goals

This project aims to develop multiple interactive math-based activities that align with Sugar's educational philosophy. These activities will be designed using **Python and GTK**, leveraging SugarLabs' existing tools and frameworks.

3 Activities and Implementation Plan

3.1 Four Color Map Game

The **Four Color Map Game** activity is an interactive exploration of the famous mathematical theorem, which states that no more than four colors are needed to color any planar map such that no two adjacent regions share the same color. This theorem was the first to be proven using a computer.

3.1.1 Goals of the Activity

The objectives of this activity are:

- Teach children about the **Four Color Theorem** through interactive map coloring.
- Introduce the concept of **graph theory** and **planar graphs**.
- Encourage problem-solving skills by allowing users to attempt coloring maps with the minimum number of colors.
- Provide visual feedback to reinforce the understanding of adjacent regions.

3.1.2 Project Structure

The project follows the directory structure below:

```
four-color-theorem/
 four-color.activity/
    activity/
       activity.info
       icon.svg
    fourcolor.py
    gameview.py
    __init__.py
    setup.py
    .gitignore
    README.md
```

3.1.3 Game Flow

The game consists of the following steps:

1. **Map Selection:** The player selects a map (or a randomly generated one).
2. **Coloring Phase:** The player applies colors to regions, ensuring that no two adjacent regions share the same color.
3. **Validation:** The system verifies whether the coloring is correct.
4. **Completion:** If the player succeeds using four or fewer colors, they win. Otherwise, they must adjust their coloring.

3.1.4 Key Components

Graph-Based Representation (Core Logic) The regions of a map are represented as a **planar graph**, where each region corresponds to a vertex and adjacent regions are connected by edges. The goal is to color the graph using at most four colors.

Graph Coloring Algorithm

```python
import networkx as nx

def four_color_map(graph):
    """Color a planar graph using at most four colors."""
    colors = {}
    available_colors = [1, 2, 3, 4]

    for node in nx.algorithms.coloring.greedy_color(graph,
  strategy="largest_first"):
        neighbor_colors = {colors[neighbor] for neighbor in
  graph.neighbors(node) if neighbor in colors}
        for color in available_colors:
            if color not in neighbor_colors:
                colors[node] = color
                break
    return colors
```

Listing 1: Graph Coloring for Four Color Theorem

Graph Representation of a Map

```python
import networkx as nx

def create_graph_from_map(region_adjacencies):
    """Generate a graph from a list of region adjacencies."""
    graph = nx.Graph()
    for region, neighbors in region_adjacencies.items():
        graph.add_node(region)
        for neighbor in neighbors:
            graph.add_edge(region, neighbor)
    return graph
```

Listing 2: Convert a Map into a Planar Graph

3.1.5 Enhancements & Future Improvements

To improve engagement and educational value, the following features can be added:

Gameplay Features

- Challenge Mode: Players must complete maps using the fewest possible colors.
- Hint System: Suggestions for correcting invalid color choices.

Educational Enhancements

- Explainability Features: Show how the four color theorem applies to real-world maps.
- Graph-Theoretic View: Display the corresponding planar graph alongside the map.

Collaboration Features

- Multiplayer Mode: Two players compete to color a map efficiently.

3.2 Fifteen Puzzle

The Fifteen Puzzle is a 4x4 sliding tile game where players arrange numbers from 1 to 15 in order by sliding tiles into an empty space. The challenge is to rearrange the shuffled tiles back into order by making legal moves.

3.2.1 Goals of the Activity

The objectives of this activity are:

- Develop problem-solving skills through spatial reasoning.
- Teach players about permutation parity and solvability of puzzles.
- Provide an interactive, visually engaging experience using GTK3 and the Sugar toolkit.
- Offer an educational approach to understanding algorithms for solving tile-based puzzles.

3.2.2 Project Structure

The project follows the directory structure below:

```
fifteen-puzzle/
 fifteen-puzzle.activity/
    activity/
       activity.info
       icon.svg
    puzzle.py
    gameview.py
    init.py
    setup.py
    .gitignore
    README.md
```

3.2.3 Game Flow

The game consists of the following steps:

- **Start Screen:** The user begins with a shuffled grid of tiles numbered 1 to 15, with one empty space.
- **Gameplay:** The player moves tiles adjacent to the empty space to rearrange the grid into numerical order.
- **Win Condition:** The game ends when all tiles are placed in order from 1 to 15 with the empty space in the bottom-right corner.

3.2.4 Key Components

Solvability Check (Core Logic) The game ensures that only solvable puzzles are generated by checking inversion counts.

```python
def is_solvable(puzzle):
    """Check if the puzzle is solvable using inversion count.
    """
    flat_puzzle = [tile for row in puzzle for tile in row if
    tile != 0]
    inversions = sum(
        1 for i in range(len(flat_puzzle))
        for j in range(i + 1, len(flat_puzzle))
        if flat_puzzle[i] > flat_puzzle[j]
    )
    return inversions % 2 == 0
```

Listing 3: Solvability Check for Fifteen Puzzle

3.2.5 Enhancements Future Improvements

To improve engagement, the following features can be added:

Gameplay Features

- Different Difficulty Levels: Allow different grid sizes (e.g., 3x3, 5x5 variations).
- Undo Functionality: Enable players to reverse moves for better learning.

Educational Enhancements

- Hint System: Suggest possible moves when the player is stuck.
- Move Counter: Track and display the number of moves taken.

Collaboration Features

- Multiplayer Mode: Implement a competitive mode where two players solve puzzles in parallel.

3.3 Broken Calculator

Broken Calculator is a mathematical puzzle where some buttons on a calculator are disabled, requiring players to reach a target number using only the available operations. This activity challenges players to think strategically and use creative problem-solving techniques.

3.3.1 Goals of the Activity

The objectives of this activity are:

- Develop problem-solving skills by requiring players to find alternative ways to reach the target number.
- Enhance understanding of mathematical operations through an interactive game.
- Provide an engaging and accessible learning experience with GTK3 and the Sugar toolkit.

3.3.2 Project Structure

The project follows the directory structure below:

```
broken-calculator/
 broken-calculator.activity/
    activity/
       activity.info
       icon.svg
    calculator.py
    gameview.py
    init.py
    setup.py
    .gitignore
    README.md
```

3.3.3 Game Flow

The game consists of the following steps:

- **Start Screen:** The player is presented with a calculator where some number and operation buttons are disabled.
- **Gameplay:** The player must reach a given target number using only the available buttons and operations.
- **Solution Validation:** The system checks if the player successfully reaches the target using a limited number of steps.
- **Win Condition:** The player wins if they successfully reach the target number using the allowed operations.

3.3.4 Key Components

Core Logic: The game ensures solvability using Breadth-First Search (BFS) to verify that the target number can be reached with the given constraints.

```python
def bfs_reachable(start, target, allowed_ops):
    """Use BFS to check if target can be reached."""
    from collections import deque
    queue = deque([(start, [])])
    visited = set()

    while queue:
        value, path = queue.popleft()
        if value == target:
            return path
        if value in visited:
            continue
        visited.add(value)
        for op in allowed_ops:
            queue.append((apply_operation(value, op), path + [
   op]))
    return None  # No solution
```

Listing 4: BFS Reachability for Broken Calculator

3.3.5 Enhancements and Future Improvements

To improve engagement and challenge, the following features can be added:

Gameplay Features

- Dynamic Challenges: Randomly disable different sets of buttons in each session.
- Difficulty Levels: Easy (basic operations) to Hard (limited operations and numbers).
- Timed Mode: Players must reach the target within a time limit.

Visual and Interactive Features

- Hints System: Provide step suggestions if the player is stuck.
- Undo Feature: Allow players to revert their last move.

Educational Enhancements

- Step Tracking: Show the history of operations used to reach the target.
- Multiple Solution Exploration: Encourage players to find alternative paths.

Collaboration Features

- Multiplayer Mode: Players can compete by solving puzzles with different constraints.

3.4 Euclid's Game

Euclid's Game is an interactive math puzzle designed to teach students the concept of the Greatest Common Divisor (GCD) using Euclid's Algorithm. The game provides an engaging way to explore mathematical reasoning through an interactive approach.

3.4.1 Goals of the Activity

The objectives of this activity are:

- Teach Euclid's Algorithm interactively.
- Help children understand the fundamental properties of GCD.
- Encourage strategic thinking through a two-player mode (Human vs AI or Human vs Human).
- Make the game visually engaging and easy to use with GTK3 and the Sugar toolkit.

3.4.2 Project Structure

The project follows the directory structure below:

```
euclid-game/
 euclid-game.activity/
    activity/
       activity.info
       icon.svg
    euclid.py
    gameview.py
    init.py
    setup.py
    .gitignore
    README.md
```

3.4.3 Game Flow

The game consists of the following steps:

- **Start Screen:** The user selects two numbers and chooses to play against AI or another player.
- **Gameplay:** The game presents two numbers (a and b). The player selects a multiple of the smaller number (b) to subtract from the larger (a). The numbers update as $a = a - k \cdot b$. The process continues until $b = 0$, and the remaining number is the GCD.
- **Win Condition:** The game ends when the GCD is reached, and the player who makes the final move wins.

3.4.4 Key Components

Euclidean Algorithm (Core Logic) This module implements the GCD calculation with step tracking.

Euclid's Game is a mathematical game where two players take turns subtracting the smaller number from the larger until they reach 1. It teaches players about the Euclidean algorithm in an interactive way.

Key Steps:

1. Implement the recursive Euclidean algorithm for turn-based subtraction.
2. Create a UI where players can select their moves.
3. Validate moves and display the updated numbers after each turn.

```python
def play_turn(a, b):
    """Execute one step of Euclidean subtraction."""
    if b == 0:
        return a
    return play_turn(b, a % b)
```

Listing 5: Euclidean Game Step

3.4.5 Enhancements Future Improvements

To improve engagement, the following features can be added:

Gameplay Features

- Different Difficulty Levels: Beginner mode (guided steps) and advanced mode (players play against AI).
- Hint System: Provides suggestions based on the best possible move.

Educational Enhancements

- Learning Mode: Explain GCD steps with textual hints.
- Step Tracking: Keep track of moves for reviewing the game later.

Collaboration Features

- Multiplayer Mode: Allow two players to compete over a network.

3.5 Number Detective

The **Number Detective** activity is a pattern recognition game where players input a sequence of numbers, and an AI attempts to predict the next number. If the AI prediction is incorrect, players can correct it, allowing the system to learn and improve over time.

3.5.1 Goals of the Activity

The objectives of this activity are:

- Teach children to recognize number patterns, including arithmetic and geometric sequences.
- Introduce the concept of **machine learning** by allowing AI to make predictions and learn from corrections.
- Encourage logical thinking and pattern recognition skills.
- Make the experience interactive and visually engaging using **GTK3 and the Sugar Toolkit**.

3.5.2 Project Structure

The project follows the directory structure below:

```
number-detective/
 number-detective.activity/
    activity/
       activity.info
       icon.svg
    detect.py
    gameview.py
    __init__.py
    setup.py
    .gitignore
    README.md
```

3.5.3 Game Flow

The game consists of the following steps:

1. **User Input:** The player enters a sequence of numbers.
2. **AI Prediction:** The AI analyzes the pattern and predicts the next number.
3. **Player Feedback:**
    - If the prediction is correct, the game continues.
    - If incorrect, the player can input the correct number, which helps improve AI learning.
4. **Score & Learning Mode:** The AI adapts based on user corrections and adjusts future predictions accordingly.

3.5.4 Key Components

Pattern Detection & Prediction (Core Logic) The AI uses rule-based detection for common patterns and **linear regression** for more complex predictions.

```python
def detect_pattern(sequence):
    """Identify if a sequence follows arithmetic or geometric progression."""
    if len(sequence) < 3:
        return None  # Not enough data

    diffs = [sequence[i + 1] - sequence[i] for i in range(len(sequence) - 1)]
    ratios = [sequence[i + 1] / sequence[i] for i in range(len(sequence) - 1) if sequence[i] != 0]

    if all(d == diffs[0] for d in diffs):
        return "Arithmetic", diffs[0]
    elif all(r == ratios[0] for r in ratios):
        return "Geometric", ratios[0]
    return "Unknown", None
```

Listing 6: Detect Arithmetic and Geometric Patterns

```python
from sklearn.linear_model import LinearRegression
import numpy as np

def predict_next_number(sequence):
    """Predict the next number using linear regression."""
    X = np.array(range(len(sequence))).reshape(-1, 1)
    y = np.array(sequence)
    model = LinearRegression().fit(X, y)
    return round(model.predict([[len(sequence)]])[0])
```

Listing 7: Predict Next Number Using Linear Regression

3.5.5 Enhancements & Future Improvements

To make the game more engaging and effective, the following features can be added:

Gameplay Features

- **Multiple Pattern Types:** Extend AI support to detect Fibonacci, quadratic, or custom user-defined patterns.
- **Hint System:** Provide hints for players when AI is uncertain.

Visual & Interactive Features

- **Animations:** Show visually how the AI predicts the next number.
- **Sound Effects:** Indicate correct and incorrect predictions.

Educational Enhancements

- **Explainable AI:** Display step-by-step reasoning for AI predictions.
- **Challenge Mode:** Players receive progressively harder sequences to solve.

Collaboration Features

- **Multiplayer Mode:** Players can challenge each other with custom sequences.

3.6 Soma Cube Activity

The **Soma Cube** activity brings to life a classic 3D spatial puzzle invented by Danish mathematician Piet Hein. The challenge is to assemble seven uniquely shaped pieces made of unit cubes into various complex structures—most notably, a 3×3×3 cube.

3.6.1 Goals of the Activity

- Develop **spatial reasoning** and **3D visualization skills**.
- Encourage exploration of geometry through hands-on manipulation.
- Inspire creativity by letting users build custom shapes like sofas, tunnels, and pyramids.
- Introduce the concept of **combinatorics** and **symmetry**.

3.6.2 Project Structure

```
soma-cube/
 soma-cube.activity/
    activity/
       activity.info
       icon.svg
    cubeview.py
    solver.py
    pieces.py
    setup.py
    README.md
```

3.6.3 Game Flow

1. **Piece Selection:** User selects and places Soma pieces on a 3D grid.
2. **Assembly Phase:** Users manipulate and rotate pieces to construct shapes.
3. **Validation:** The system checks if the structure matches the target form.
4. **Free Build Mode:** Users can experiment and save custom creations.

3.6.4 Key Components

3D Grid and Piece Manipulation Each Soma piece is defined by a set of relative cube positions and can be rotated in 3D. The interface allows dragging, dropping, and rotating pieces.

Solver (Optional AI Help)

```python
def solve_soma_cube(pieces, cube_size=(3,3,3)):
    # Backtracking algorithm to place pieces into a cube
    # Returns one or more solutions
    ...
```

Listing 8: Backtracking Solver for Soma Cube

3.7 Odd Scoring Activity

Odd Scoring is a turn-based logic game involving a chip on a linear board of N cells. The player and the computer alternate moving the chip left by 1, 2, or 3 steps. The winner is the one with an even number of total steps when the chip reaches the final cell.

3.7.1 Goals of the Activity

- Teach **game theory** and **strategic thinking**.
- Encourage mathematical reasoning around parity and optimal moves.
- Help students explore algorithms for turn-based games.

3.7.2 Project Structure

```
odd-scoring/
 odd-scoring.activity/
    activity/
       activity.info
    game.py
    strategy.py
    ui.py
    setup.py
    README.md
```

3.7.3 Game Flow

1. **Board Setup:** The user chooses the length N (odd number).
2. **Gameplay:** Players take turns moving the chip.
3. **Scoring:** Once the chip reaches the end, total steps are tallied.
4. **Victory Check:** Player with even steps wins.

3.7.4 Key Components

Winning Strategy An AI component can determine optimal moves using minimax or dynamic programming.

```python
def best_move(position, history):
    # Return optimal move (1, 2, or 3 steps)
    # Use parity of steps to predict outcome
    ...
```

Listing 9: Parity-Based Strategy

3.8 Latin Squares Activity

This activity challenges users to complete a grid so that every number appears exactly once per row and column. The advanced mode introduces symmetry along the main diagonal, turning it into a deeper puzzle.

3.8.1 Goals of the Activity

- Introduce concepts of **Latin squares** and **permutations**.
- Foster logical thinking and visual pattern recognition.
- Strengthen understanding of **matrix symmetry**.

3.8.2 Project Structure

```
latin-squares/
 latin-squares.activity/
    activity/
       activity.info
    board.py
    solver.py
    ui.py
    setup.py
    README.md
```

3.8.3 Game Flow

1. **Puzzle Selection:** Simple or symmetric mode.
2. **Interaction:** Swap or cycle cells to achieve valid configuration.
3. **Validation:** Check for uniqueness and symmetry constraints.
4. **Completion:** Puzzle is solved when all conditions are met.

3.8.4 Key Components

```python
def is_valid_latin_square(grid):
    size = len(grid)
    for row in grid:
        if len(set(row)) != size:
            return False
    for col in zip(*grid):
        if len(set(col)) != size:
            return False
    return True
```

Listing 10: Latin Square Validator

```python
def is_symmetric(grid):
    size = len(grid)
    for i in range(size):
        for j in range(i+1, size):
            if grid[i][j] != grid[j][i]:
                return False
    return True
```

Listing 11: Diagonal Symmetry Checker

If you want, I can help generate the full '.activity' folders or code scaffolds for any of these. Want to start with one of them?

Note: The sequence of activities can be adjusted based on mentor feedback and Sugar Labs' requirements to ensure the most impactful games are prioritized.

4 Road-map

I am flexible with my working hours and can work during night time (IST Time Zone) as well. This summer, I have no commitments as such. I can devote 7-8 hours per day during that time. Also, I check my emails at least twice a day. So, communication can be done through mail, and my response can be expected within less than 24 hours.

4.1 Project Timeline

Phase	Timeline	Key Activities
Community Bonding Period (May 8 - June 1)		
Planning and Research	May 8 - June 1	• Engage with the mentor to refine project goals. • Set up the development environment and test compatibility with Sugar. • Study Sugar's codebase and understand its architecture. • Review existing Sugar math activities to identify design patterns. • Gather feedback from potential users (students and educators) to refine requirements.
Coding Phase 1 (June 2 - July 14)		
Development	June 2 - June 30	• Implement core game mechanics for selected activities. • Design user interfaces using `Gtk.Grid()` and ensure consistency. • Ensure proper logic for activity rules and interactions. • Test fundamental functionalities and refine based on initial feedback.

Testing and Refinements	July 1 - July 14	• Conduct thorough testing of implemented activities. • Fix bugs and optimize the codebase. • Improve UI/UX based on usability feedback. • Prepare a midterm evaluation report summarizing progress.
Midterm Evaluation (July 14 - July 18)		
Evaluation and Feedback	July 14 - July 18	• Submit midterm evaluation report. • Gather feedback from the mentor and community. • Plan the next phase based on received feedback.
Coding Phase 2 (July 15 - August 25)		
Advanced Development	July 15 - August 4	• Implement AI features and advanced game logic. • Ensure cross-activity consistency in UI/UX. • Add localization and accessibility improvements. • Conduct usability testing and refine activities.
Final Testing and Bug Fixing	August 5 - August 11	• Perform extensive playtesting with users. • Fix critical and minor bugs. • Optimize code efficiency and performance.

Documentation and Final Preparations	August 12 - August 25	• Complete user documentation and developer guides. • Finalize submission materials. • Prepare for final project review.
Final Submission (August 25 - September 1)		
Project Submission	August 25 - September 1	• Submit the final work product. • Complete mentor evaluations. • Write and submit a final project summary.

4.2 Post GSoC

Sugar Labs provides an excellent platform for me to apply my programming skills while creating engaging and educational activities for children. Through this project, I aim to make learning mathematics more interactive and accessible.

Beyond GSoC, I plan to continue contributing to Sugar Labs by improving the activities, adding new features, and refining the user experience based on feedback. Additionally, I would love to explore other areas within Sugar's ecosystem where my skills can help enhance educational tools and make learning more intuitive and enjoyable for young learners.

References

[1] Sugar Labs, `Sugar Documentation`
https://github.com/sugarlabs/sugar-docs?tab=readme-ov-file#sugar-desktop

[2] Sugar Labs, `Sugar Development Environment`
https://github.com/sugarlabs/sugar/blob/master/docs/development-environment.md

[3] Sugar Labs, `Developing a Sugar Activity`
https://github.com/sugarlabs/sugar-docs/blob/master/src/desktop-activity.md

[4] GTK, `GTK Documentation`
https://docs.gtk.org/

[5] Python Software Foundation, `Python Documentation`
https://docs.python.org/3/

[6] Pygame Community, `Pygame Documentation`
https://www.pygame.org/docs/

[7] Reinhard Diestel, *Graph Theory*
https://www.math.uni-hamburg.de/home/diestel/books/graph.theory/

[8] Tom M. Mitchell, *Machine Learning*
https://www.cs.cmu.edu/~tom/mlbook.html

[9] Christopher M. Bishop, *Pattern Recognition and Machine Learning*
https://www.microsoft.com/en-us/research/people/cmbishop/

[10] Euclidean Algorithm, *Wikipedia*
https://en.wikipedia.org/wiki/Euclidean_algorithm

[11] Four Color Theorem, *Wikipedia*
https://en.wikipedia.org/wiki/Four_color_theorem