GSoC 2025 Application – Sugar Labs

My Details:

Name: Samiksha Sharma
Email: samikshasharma7500@gmail.com
GitHub: https://github.com/samikshaa11
LinkedIn: https://www.linkedin.com/in/samiksha-sharma11
Time Zone: Indianapolis, USA – Eastern Daylight Time (EDT/EST)

I am a graduate student at Purdue University, deeply passionate about coding and contributing to real-world projects. My experience includes developing projects like Blink Detection, Hospital Management System, and Netflix Recommendation System during my master's program. I am excited to contribute to Sugar Labs, particularly the Math Games project, and leverage my skills to enhance learning experiences through interactive games.

Interest in contributing to Project:

Math Games

Technologies used:

Python: To develop math games with Sugar's framework.

Sugar Activity API: To integrate and manage Sugar activities, ensuring they work within Sugar platform.

PyGTK: Designed as Sugar Activity for user interface that will integrate well with Sugar environment.

SVG/PNG: To design and create visually appealing game assets, including backgrounds, buttons, and interactive elements, enhancing the user interface.

My Plan for the 8 Games:

1. Four Color Map Game:

Description: A game where players color a map with four colors ensuring that no adjacent regions share the same color.

Technologies:

- **Python** for game logic.
- **Sugar Activity API to package the game as a Sugar Activity**, as it integrates with Sugar.
- Using the **Sugar Network API**, I'll enable players to collaborate in real-time, sharing their progress and working together to solve the puzzle.
- **Backtracking algorithm** for the graph coloring logic.

Implementation Details:

- I'll use **graph theory** (adjacency matrices or lists) to represent map as a graph taking each region as a node.
- Implement the **backtracking algorithm** to try different colors for regions, to check if no two adjacent regions have the same color.
- **Sugar Toolkit – Sugar Activity** will be used for creating the interface, including the map and color palette and it will be designed as a Sugar activity, allowing it to integrate seamlessly with the Sugar environment. The game state will be saved in the Sugar Journal for easy access and continued play across sessions.

2. Broken Calculator:

Description: A puzzle where the calculator malfunctions, and players need to fix it by identifying missing or incorrect functions.
Technologies:

- **Python** for core logic and puzzle mechanics.
- PyGTK for GUI design, integrated with the **Sugar Activity API** to handle activity-specific functions like startup, shutdown, and saving progress within Sugar's environment.
- **State-based logic** for the calculator's operation and malfunction scenarios.

Implementation Details:

- This game will simulate a **calculator** that starts with a set of broken operations (e.g., missing functions or faulty buttons).
- Players will interact with the interface to identify and correct errors in the calculator's logic, restoring its functionality. The game will save progress in the **Sugar Journal**, enabling users to pick up where they left off, while adhering to Sugar’s lifecycle management. The activity will be designed to follow Sugar's lifecycle management, ensuring proper start-up and shutdown within the Sugar environment.
- I’ll use a simple **state machine** to manage the calculator’s correct and incorrect states.

3. Soma Cubes:

Description: A 3D puzzle game where players must arrange pieces to form a cube.
Technologies:

- **Python** for game logic.
- **PyGTK** for interface creation. The game will be designed as a Sugar activity, ensuring that players’ progress is saved in the Sugar Journal. The activity will also integrate with Sugar’s framework to manage activity lifecycle events such as start, stop, and resumption of the game.
- **3D models** (potentially using **PyOpenGL** for rendering the pieces).

Implementation Details:

- I’ll define **3D shapes** and their corresponding **arrangement rules**.
- Implement drag-and-drop functionality within the **PyGTK** interface to move pieces.
- **PyOpenGL** can be used for rendering the pieces in 3D, providing users with an interactive, visually engaging puzzle experience.

4. Fifteen Puzzle:

Description: A sliding puzzle game where players arrange tiles in a grid.
Technologies:

- **Python** for the game logic.
- **PyGTK** for UI.

- *A search algorithm** for puzzle-solving assistance (optional for hints).

Implementation Details:

- Represent the puzzle grid as a 2D list or matrix.
- I'll implement the **sliding mechanism** where tiles will move within the grid. As a Sugar activity, the game will store the puzzle state in the Sugar Journal, allowing players to pick up where they left off. The game will also adhere to Sugar's activity lifecycle, ensuring smooth transitions between states.
- I will implement the *A algorithm** to solve the Fifteen Puzzle, helping users find optimal solutions efficiently, while also providing hints if needed.

5. Euclid's Game:

Description: A game based on Euclid's algorithm to find the greatest common divisor (GCD).
Technologies:

- **Python** for the logic and implementation of Euclid's algorithm.
- **PyGTK** for user interaction and display.

Implementation Details:

- This game will display two numbers, and the player will use **Euclid's algorithm** to calculate their GCD. As a Sugar activity, the game will track and save the player's progress in the Sugar Journal. It will follow Sugar's activity lifecycle to ensure proper management of game state when transitioning between activity states.
- Provide a simple interface where players can input their guesses and see step-by-step calculations.
- Visualize the **steps of the algorithm** using simple UI elements to help players understand the process.

6. Odd Scoring:

Description: A game where players match numbers with specific properties like prime numbers or even/odd.
Technologies:

- **Python** for game logic.
- **PyGTK** for GUI.

Implementation Details:

- I'll display a set of numbers and prompt the player to identify whether they are **prime, even, or odd**. As a Sugar activity, the game will integrate with Sugar's Journal, ensuring players' scores and progress are stored for continued play. The activity will be managed within Sugar's lifecycle framework for proper handling of activity states.
- Create a **matching system** where players drag and drop the numbers into the correct category.
- Implement a scoring system based on correct matches and a time-based challenge for difficulty.

7. Make An Identity:

Description: A game where players create valid algebraic identities using numbers and operators.
Technologies:

- **Python** for algebraic identity generation and verification.
- **PyGTK** for user interface.
- **SymPy** for symbolic algebra (optional).

Implementation Details:

- Provide a set of numbers and operators for the player to create an identity. The game will be designed as a **Sugar activity**, ensuring progress is saved in the **Sugar Journal** and providing users with the ability to resume later. The game will also be integrated into the Sugar platform, utilizing its activity lifecycle management features.
- Use **algebraic rules** and basic math verification to check if the identity is valid.
- **SymPy** can be used to assist in symbolic mathematics to ensure identity correctness if necessary.

8. Number Detective:

Description: A game where players input a sequence of numbers, and the system guesses the next number.
Technologies:

- **Python** for game logic.
- **PyGTK** for interface creation.
- **Basic pattern recognition algorithms** (e.g., arithmetic sequences, geometric sequences).

Implementation Details:

- Players input a sequence of numbers, and the game tries to identify the next number based on the sequence's pattern. As a Sugar activity, the game will store the number sequences and progress in the Sugar Journal, allowing users to resume their work later. It will also integrate with the Sugar platform's lifecycle management to ensure proper handling of the game's start, stop, and resume states.
- Implement simple **pattern recognition** techniques to detect sequences such as **arithmetic progression** or **geometric progression**.
- Provide feedback if the system makes an incorrect guess, allowing the player to correct it and learn from the process.

Hours/Duration:

350 hours

After GSoC 2025:

After GSoC 2025, I plan to continue contributing to the **Sugar Learning Platform**, further enhancing the Math Games and developing new educational tools to engage and benefit learners around the world