Final project

Workshop: Software Product

Project name: Boolean Logic

Project goal: The goal of the project is a playful and interactive application for understanding and improving mental logic and Boolean algebra.

Introduction

Boolean Logic is a web application that includes a strategic board game, two challenge games that involve analyzing Boolean formulas and a graphical simulator of building logic circuits using logic gates. The goal of the project is to improve logical thinking in general and Boolean algebra in particular in a fun and challenging game-like way and through experimentation in building logic circuits.

This project is suitable for the Software Product workshop because its product will be a web application with a server side and a client side and several features (games and logic circuit simulator).

Project description

As stated, the app will include four main features: a strategic board game, two challenge games that involve analyzing Boolean formulas and a logic circuit simulator.

Feature Description:

The board game:

The board game is played on a 7x7 grid board, so you can think of it as a chessboard where all the black squares are logic gates squares and all the white squares are bits square.

The game has two different game modes- for two players (or against the computer) and for one player.

mode 1 – two players:

At the beginning of the game, all the squares of the logic gates are filled with randomly selected logic gates. Each player in turn chooses a bit square and chooses a bit that he places in the square (0 or 1). This bit is colored in the player's color. Each logic gate receives two values: a value for the two bits above and below it, and a value for the two bits to its right and left. When a logic gate receives a value, the bits for which this value is valid (the two bits on either side of the gate or the two bits above and below the gate) earn or lose points to the player who placed them (the color of each bit is the same as the color of the player who placed it) according to the following rules: If the bit is equal to the gate value - the player earns a point, if the bit is different from the gate value - the player loses a point. The game ends when all the squares are filled and the winner is the player with the highest number of counters, in this mode you can play against the computer or one-on-one online.

A sketch of some game stages in this mode:

start of game:

	or		and		or	
nand		or		xor		and
	and		nor		nxor	
xor		nand		nor		or
	or		and		nor	
nor		xor		nxor		and
	or		nand		nor	

red score: 0 blue score: 0

middle of game:

	or	1	and	1	or	
nand		or		xor		and
	and	0	nor		nxor	

xor		nand		nor		or
	or		and		nor	
nor		xor		nxor		and
	or		nand		nor	

red score: 2 blue score: 0

As you can see in the example, after three turns in the game, the value of the "and" gate in the top row for the two bits on its sides is 1 and since their value is also one - each of these bits earns a point for the player who placed them, in this case one point for the blue player and one point for the red player, for the "or" gate in the second row - its value for the two bits above and below it is 1, Therefore, the bit above it earns a point for the red player because its value is also 1 and is equal to the gate value, the bit below the "or" gate deducts a point for the blue player because it is 0 and is different from the gate value (which is 1), meaning that the blue player has so far earned a point and lost a point, so he currently has 0 points, and the red player has earned two points.

End of game:

1	or	1	and	1	or	0
nand	0	or	1	xor	1	and
0	and	0	nor	0	nxor	0

xor	0	nand	0	nor	1	or
1	or	1	and	1	nor	0
nor	1	xor	1	nxor	0	and
0	or	1	nand	0	nor	0

red score: 10 blue score: -7

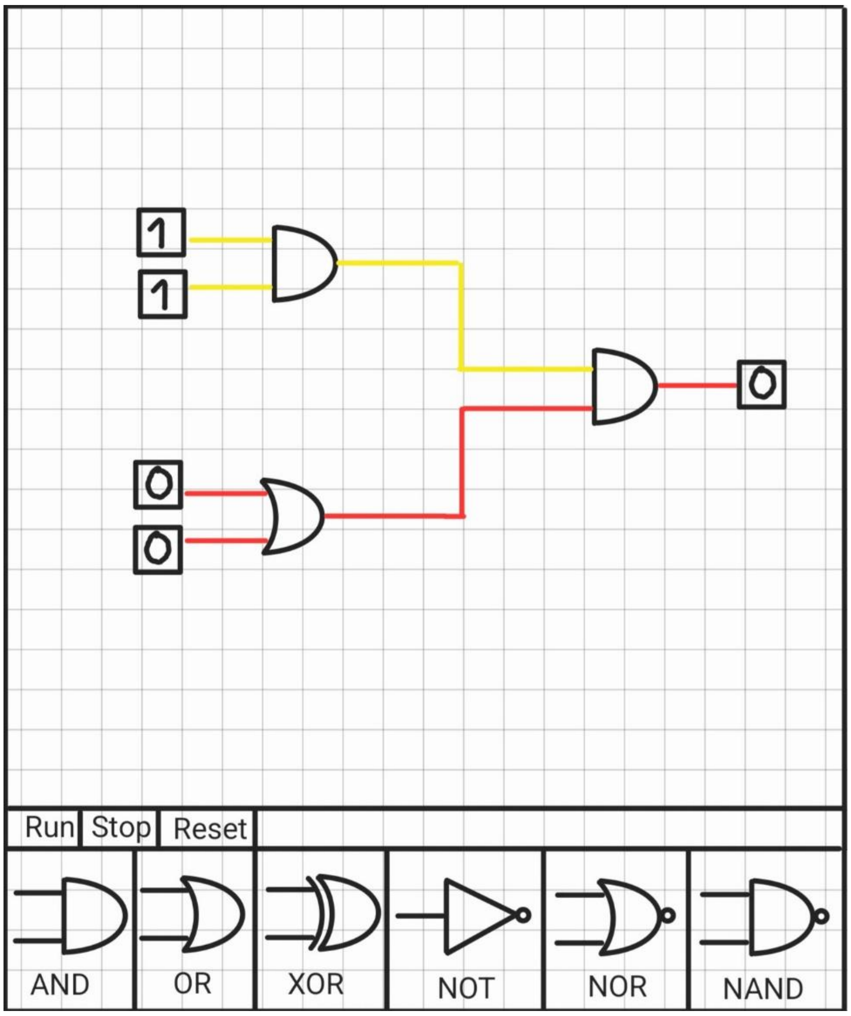
Mode 3 – one player:

In this mode, the game is the same as described, only there is one player who needs to fill the entire board and reach the maximum score that can be achieved on the current board (i.e. with the current logical gates that have been drawn). Once the player has filled the board, the computer tells him by how many points he is far from an optimal solution (i.e. a solution of filling bits in the board that will achieve the maximum number of points that can be achieved on this board), and the player can keep changing the bits on the board until he reaches an optimal solution.

The logic circuit simulator:

The logic circuit simulator provides a free-form workspace in which users can build digital circuits. Users can drag logic gates from a ToolBar containing all the logic gates, wires to connect the gates, and wire split points onto a canvas, connect them using wires, and assign binary values to input nodes. The system visualizes signal propagation in real time, showing where signals flow, stop, or branch.

Partial sketch of the simulator:



In the sketch, a red wire represents a wire that has 0 passing through it and a yellow wire represents a wire that has 1 passing through it.

Boolean Formula Evaluation Game:

This game is for one or two players.

In the game, the player or both players will be given a Boolean formula with Boolean variables and a placement, and will have to solve the formula.

The score in this game will be based on the correctness of the solution and the time it took to reach the solution

Example of a formula as given in this game:

$$(\bar{x} \land \bar{y} \land \bar{z}) \lor (\bar{x} \land \bar{y} \land z) \lor \overline{(x \land z)} =$$

with the placement: x = 1, y = 0, z = 0

<u>Boolean Formula Equivalence Game:</u>

This game is for one or two players.

In the game, the player or both players will be given two Boolean formulas with Boolean variables like the example formula we showed in the previous section, the formulas look syntactically different but may or may not be logically equivalent. The player's task is to determine whether the two formulas are Logically equivalent or not equivalent.

The score in this game will be based on the correctness of the solution and the time it took to reach the solution

<u>App visibility:</u>

When you enter the application, the main screen will include three buttons: one for the game, one for the simulator, and one for an explanation of the different logical gates.

When you click the game button, you go to a screen with three buttons - one for each game mode - if you choose one of the two game modes that are for two players, you go to a screen with two buttons: one for online play and if you click this button, then as soon as there is another player waiting to play, a game is applied.

The second button is for playing against the computer, and when you click on it, a small window opens to select the computer's difficulty level (there will be 2 or 3 difficulty levels), and after you select the difficulty level, the game begins.

When you press the game button in mode 3, the game starts immediately, and when you press the explanation or simulator button, it also happens immediately.

Project design

The app follows a client–server architecture.

The system is divided into two top-level architectural layers:

Client (Frontend) – responsible for user interaction, visualization, local logic execution, and offline gameplay.

Server (Backend) – responsible for online multiplayer coordination, authoritative validation, and synchronization between remote clients.

<u>Main modules:</u>

<u>Logic Gate Handler:</u>

Purpose: The Logic Gate Module provides the fundamental, deterministic computation of Boolean logic gates. It represents the lowest-level logical building block of the system and is reused across multiple modules. This module is completely independent of any specific game or user interface.

Components:

Gate Evaluator - Computes the output of a logic gate (AND, OR, XOR, NOT, NAND, NOR, etc.) given one or more binary inputs.
This component encapsulates the truth tables of all supported gates.

Input/Output Interface - Defines a uniform interface for passing inputs into a gate and retrieving outputs.

This abstraction allows the same gate logic to be used both in grid-based board evaluation and in wire-based circuit simulation.

Module technology - TypeScript

<u>Board Game Engine:</u>

Purpose: The Board Game Engine implements all rules and mechanics of the board-based Boolean logic game.

Components:

Move Validator - Checks whether a player move is legal according to the board rules, cell type restrictions, and turn order.

Score Calculator - Evaluates logic gates affected by a turn and updates player scores based on correctness of outcomes.
Uses the Logic Gate Module for gate evaluation.

Game State Manager - Maintains the full board state, including gate placements, bit values, current turn, and game completion detection.

Module technology – TypeScript

<u>**Computer Player Engine:**</u>

Purpose: Provides deterministic decision-making for the player-versus-computer mode of the board game. Uses deterministic algorithms such as Minimax, Alpha-Beta pruning, and heuristics.

Components:

Move Generator - Generates all legal moves available for a given board state.

Search Best Move - Explores possible future game states using Minimax with optional Alpha–Beta pruning, and return the move for this turn.

Heuristic Evaluator - Assigns scores to non-terminal board states to guide the search algorithm.

Module technology – TypeScript

Logic Circuit Simulator:

Purpose: Allows users to construct and simulate logical circuits visually.
Users can place gates, connect them with wires, assign inputs, and observe real-time signal propagation.

Components:

Circuit Canvas - Represents the simulation workspace where gates and wires are placed and rendered dynamically.

Gate Component - Represents an interactive logic gate within the simulator.
It updates its output dynamically when input signals change.

Wire Component - Represents connections between gates and inputs.
Visualizes signal values using color-coded animated propagation (yellow for 1, red for 0).

Pulse Animator - Controls the animated flow of signals along wires to simulate current propagation over time.

Module technology – TypeScript, HTML5 Canvas for wire rendering, SVG for gates and connection points

Boolean Formula Game Engine:

Purpose: Implements logic-based games centered around Boolean formulas, including formula evaluation and formula equivalence challenges.
This module is separate from the board game and circuit simulator.

Components:

Formula Parser - Parses textual Boolean expressions into an internal structured representation (e.g., expression trees).
Handles operator precedence, parentheses, and unary/binary operators.

Expression Evaluator - Evaluates parsed expressions using provided variable assignments.
Uses the Logic Gate Module for logical computation.

Game Mode Manager - Controls which formula-based game mode is active (evaluation or equivalence).

Formula Evaluation Mode - Validates player answers for Boolean formulas and calculates score based on correctness and response time.

Formula Equivalence Mode - Determines whether two Boolean formulas are logically equivalent.

Timer Manager & Scoring Manager - Measure response time and calculate score accordingly.

Module technology – TypeScript.

Description of architecture layers:

Client:

Purpose: The client provides the user interface, manages user interactions, renders visual elements, and executes all logic required for single-player and local gameplay.

Contained Modules:

- Board Game Engine
- Computer Player Engine
- Logic Circuit Simulator
- Boolean Formula Game Engine

- Logic Gate handler

Key Responsibilities:

- Rendering the board, circuits, and formulas
- Handling user input
- Running local game logic
- Displaying animations and feedback

Technologies:

- React with TypeScript
- State management: React Context or Zustand/Redux
- Canvas and SVG for visualization
- HTML/CSS for layout and styling

Server:

Purpose: The server supports online multiplayer functionality and acts as the authoritative source of truth for competitive games.

Contained Modules:

- Board Game Engine (for validation)
- Boolean Formula Game Engine (for validation)
- Logic Gate handler

Key Responsibilities:

- Managing game rooms and player connections
- Validating moves and results
- Synchronizing state between clients

Components:

Game Room Manager - Creates and manages multiplayer sessions and player assignments.

WebSocket Manager - Handles real-time bidirectional communication with connected clients.

Technologies:

- Node.js runtime
- Express.js for REST endpoints (game creation, joining)
- WebSocket for real-time updates