Ain Shams University

Faculty of engineering

Computer and Systems Engineering department

CSE321: Software Engineering

Software Engineering Project Requirement Specifications

Logic Gates Simulator (SRS)

Submitted to Prof: Gamal A. Ebrahim

Eng : Lara wahib

Submitted by :

student name	ID
أحمد إيهاب أحمد محمد عيسى	1300072
أحمد أسامه محمد علي فرج	13E0001
خالد صلاح الدين حسن جمال	1300489
عبد الرحمن أسامه عبد الإله جابر	1300729
أميره محمد السعيد مدين	13E0042
رامي ناصر عبد الفتاح ممدوح	1300540

Abstract:

The aim of this documentation is to specify the main features of a software project.

This document is a software specification for simple software engineering project called “LogicGate simulator”.

This software is developed by group of computer and systems engineering students at faculty of engineering Ain Shams University under supervision of
CSE321 Software Engineering teaching stuff.
we hope you like it and find it useful.

Table of Contents

1. Introduction

1.1 Purpose:

This document's purpose it to state and clarify everything the users/ stakeholders need to know about the LogicGateSimulator software.

The document includes a general description of the system; the product's perspective, its general capabilities, the constraints imposed on it, the characteristics of the different user roles, and a description of the operational environment of the system.

It also includes a list of all terms used and their definitions, the project's scope, and the system's functional and nonfunctional requirements.

Several UML diagrams will be used for a better communication Language since Graphical notation is much better than words. This document includes: Use-case diagram, state diagram, interaction, diagram, detailed class diagram, and client-object diagram.

The document will also include a description of the testing process, a cost estimate, and a user guide.

1.2 List of definitions:

Simulation	Simulation is the imitation of the operation of a real-world process or system over time.
logic circuit	A circuit for performing logical operations on input signals
Node	

	A point in a network or diagram at which lines or pathways intersect or branch
operating system	The low-level software that supports a computer's basic functions, such as scheduling tasks and controlling peripherals
C++.	C++ is an object oriented programming (OOP) language, developed by Bjarne Stroustrup, and is an extension of C language
logic gate	A logic gate is an elementary building block of a digital circuit. Most logic gates have two inputs and one output. At any given moment, every terminal is in one of the two binary conditions low (0) or high (1), represented by different voltage levels. (AND, OR, XOR, NOT, NAND, NOR and XNOR)
Circuits diagrams.	A graphical representation of an electrical circuit.

Digital logic design	A system in electrical and computer engineering that uses simple number values to produce input and output operations.
Object oriented methodology	Object-oriented analysis and design (OOAD) is a popular technical approach for analyzing, designing an application, system, or business by applying the object-oriented paradigm and visual modeling throughout the development life cycles to foster better stakeholder communication and product quality.

1.3 Scope:

The product is called “LogicGateSimulator”. It is a simulator for simple and complex logical circuits. It features several logic gates including: And gate, Or gate, Not gate, Xor gate, Xnor gate, and Nand gate. It can obtain the value of all nodes and calculate the circuits’ outputs.

1.4 Overview:

This document provides details and diagrams on the “LogicGateSimulator” software. First, it provides a general description of the product; it’s perspective, capabilities’ and constrains. Secondly, it discusses the system’s requirements (**section 3**); functional and nonfunctional, and provides Requirements Validation through a “Functional Requirements Traceability matrix” (**section 6**). A Narrative description is provided on **section 5**.

After that, several diagrams are presented such as Use -Case Diagram (**section4**), Class Model (**section 7**), State Diagram (**section 8**), Sequence Diagram (**section 9**), Detailed Class Diagram (**section 10**), Client-Object Relation Diagram (**section 11**).

This document also provides a section for testing (**section 13**); providing several examples and their outputs. It also has a calculated cost estimate discussed on **section 14**.

2. General Description

2.1 product Perspective

The product is supposed to be a simple simulator that simulates logic circuits.
This logic circuits simulator provides simple mechanism to run and simulate logic circuits and find desired outputs for deferent designs.

The following are the main features that are included in the simulator

- User friendly: The system support a simple writing way so user can insert the desired circuit design.
- One node evaluation: The system supports printing the value of one node (desired one) after simulating the circuit.
- User manual: the system is provided with a user manual that is written in a user-friendly way.

2.2 General Capabilities

In this Simulator General Capability encompasses knowledge, skills, behaviors and dispositions.

It includes

- Critical and Creative thinking
- Ethical understanding
- The ability to simulate
- Supports different levels complexity of circuits

2.3 General Constrains

General design/implementation constraints include:

- The software system will run under Windows operating system.
- All codes are written in C++.
- The simulator is developed using the methodology specified in section 3 and 4.
- The documentation and code is in accordance with software Engineering requirement specifications.

2.4 User Characteristics

All users can be assumed to have the following characteristics:

- Ability to read and understand English.
- Familiarity with logic gates.
- Familiarity with circuits diagrams.
- Familiarity with Digital logic design.
- Ability to open computer and write in text file.
- Beyond the above, no further facility with computer technology can be assumed.

2.5 Environment Description

The logic circuits simulator is mainly developed using Microsoft visual studio and is written in C++ using object oriented methodology and it runs only on Microsoft Windows operating system.

Since this software is a simple simulator there is no need to use database engine.

2.6 Assumptions and Dependencies

The logic circuits simulator developers assume that the stakeholders to this software is familiar with basics of digital logic design and able to use logic gates in their logic circuit design.

It is also assumed that stakeholders are familiar with using computer, keyboard, mouse and can write in txt file.

2.7 Other resources needed

Only familiarity with basics of digital logic circuits and ability to use computer.

Since the software is a simple simulator you can run it on any computer with Microsoft windows OS.

3.System Requirements

3.1 Functional Requirements:

1)The software shall allow the end user to add a gate.to his circuit.

2)The software should allow the user to design any logic circuit whatever its complexity.

3)The software shall support (AND/OR/NAND/NOR/XOR/XNOR/NOT) gates.

4)The software can simulate the value of any node in the circuit.

5)The software can calculate the total output of the circuit.

3.2 Non-Functional Requirements:

1)The software must be written in C++.

2)Nonprofessional users must find it easy to deal with the software.

3)The user shall deal with a text editor then passes it to the software.

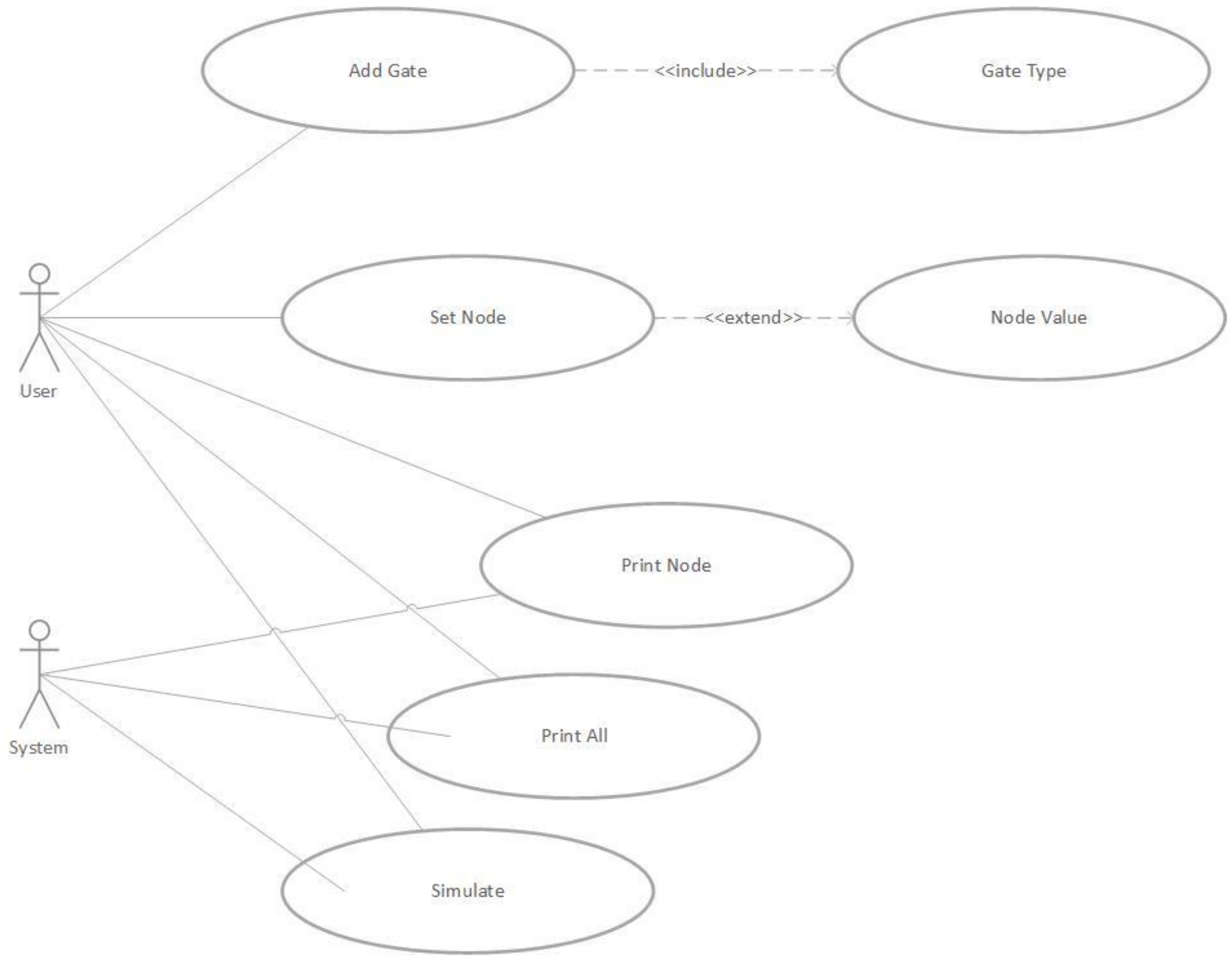
4)The software must be available within 5 weeks.

5)The speed of executions is a must (max 10 seconds to simulate the output of the circuit).

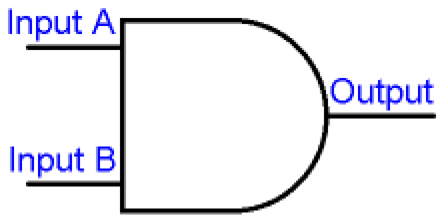
6)Deletion of any unnecessary data in memory.

7)The software should follow the common standards such that the name of the gate should be typed first then the name of the input nodes and finally the output node's name (for more info see the user guide).

4.Use -Case Diagram



5. Narrative description



-Normal scenario:

1. The user enter the gate type (which is AND gate)
2. The user enter the inputs and output symbols
3. The user SET values to the inputs (either 0 or 1)
4. Write SIM to start simulation
5. Write OUT output to view the output of the AND gate

-Exception scenario :

1. The user enter the gate type (which is AND gate)
2. The user enter the inputs and output symbols
3. The user SET values to the inputs (either characters or numbers except 0, 1)
4. There will be an error
5. The user should change this mistake
6. Write SIM to start simulation
7. Write OUT output to view the output of the AND gate

6.Requirements Validation

Functional Requirements Traceability matrix:

	1	2	3	4	5
1					
2	D*				
3	D**				
4		D***	D****		
5				D****	

Table 1: Functional Requirements Traceability Matrix

*2 depends on 1 as the user can design any logic circuit starting from adding a gate to his design.

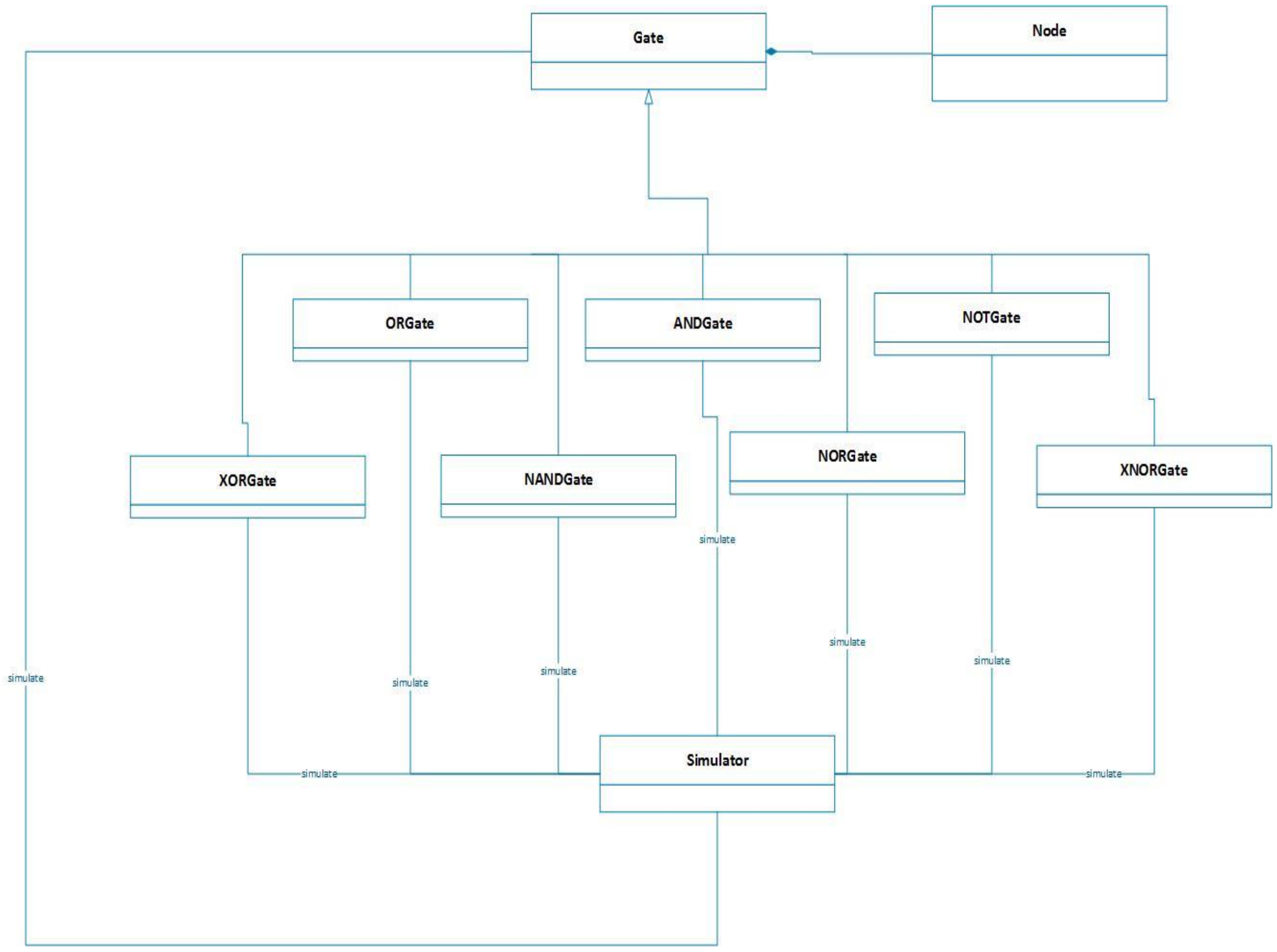
**3 depends on 1 as the system support the type of the gates that the user will add to the circuit design.

***4 depends on 2 as the software can simulate the nodes' value based on the design of the circuit.

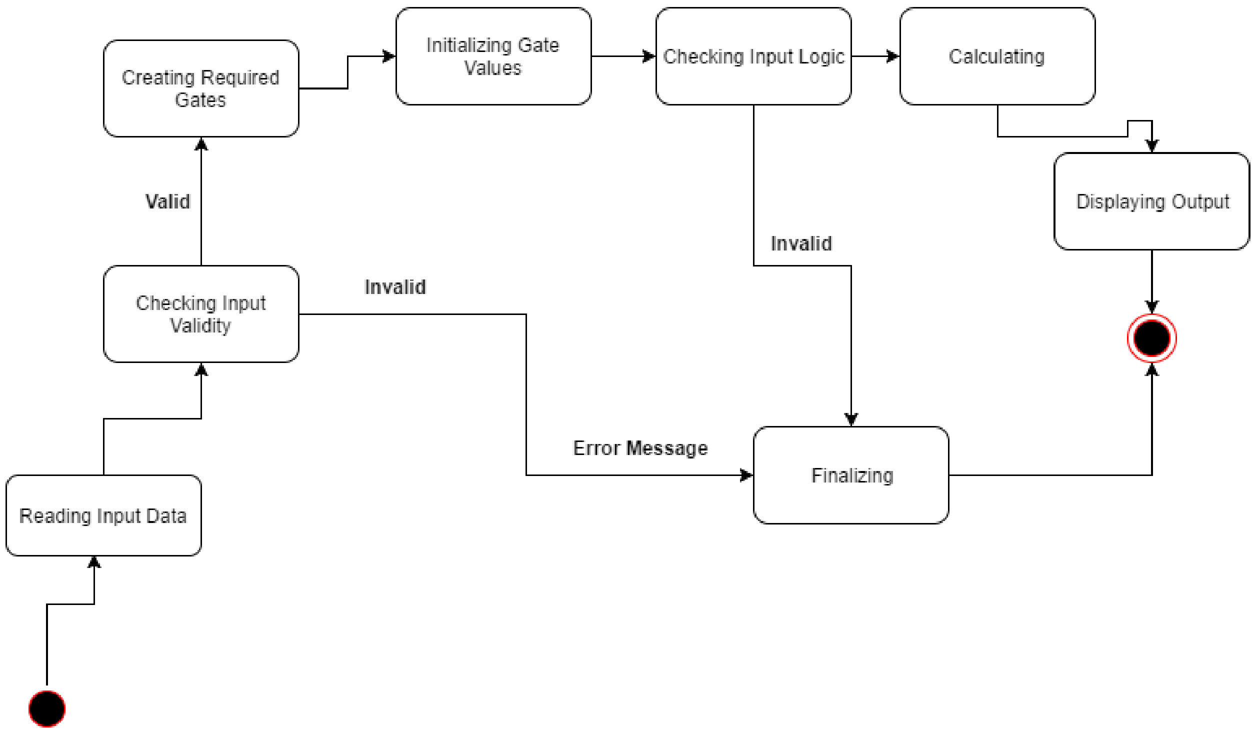
****4 depends on 3 as the software can simulate the nodes' value based on the type of the gates.

*****5 depends on 4 as the total output of the circuit depends on the nodes' value of the circuit.

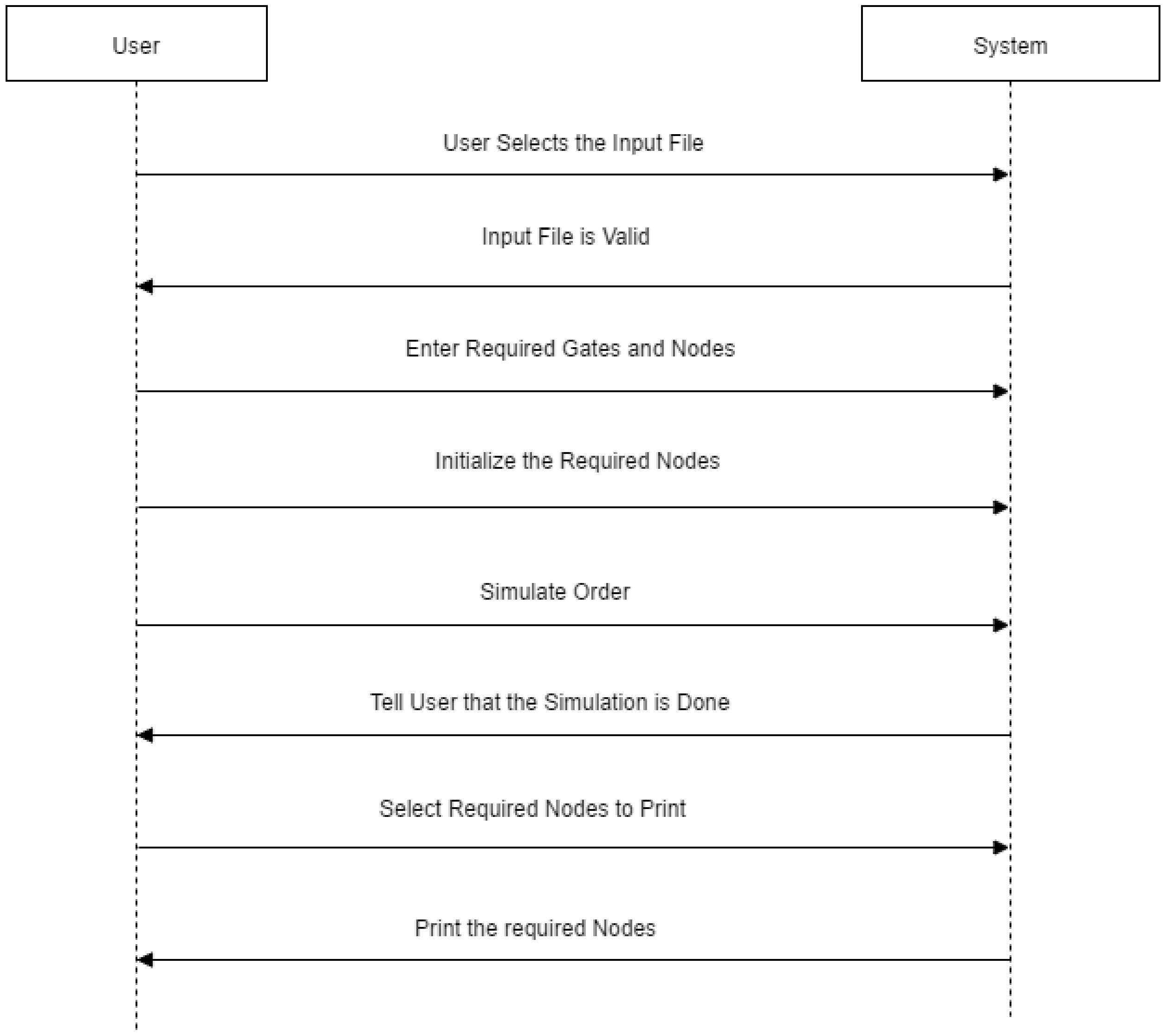
7.Class Model



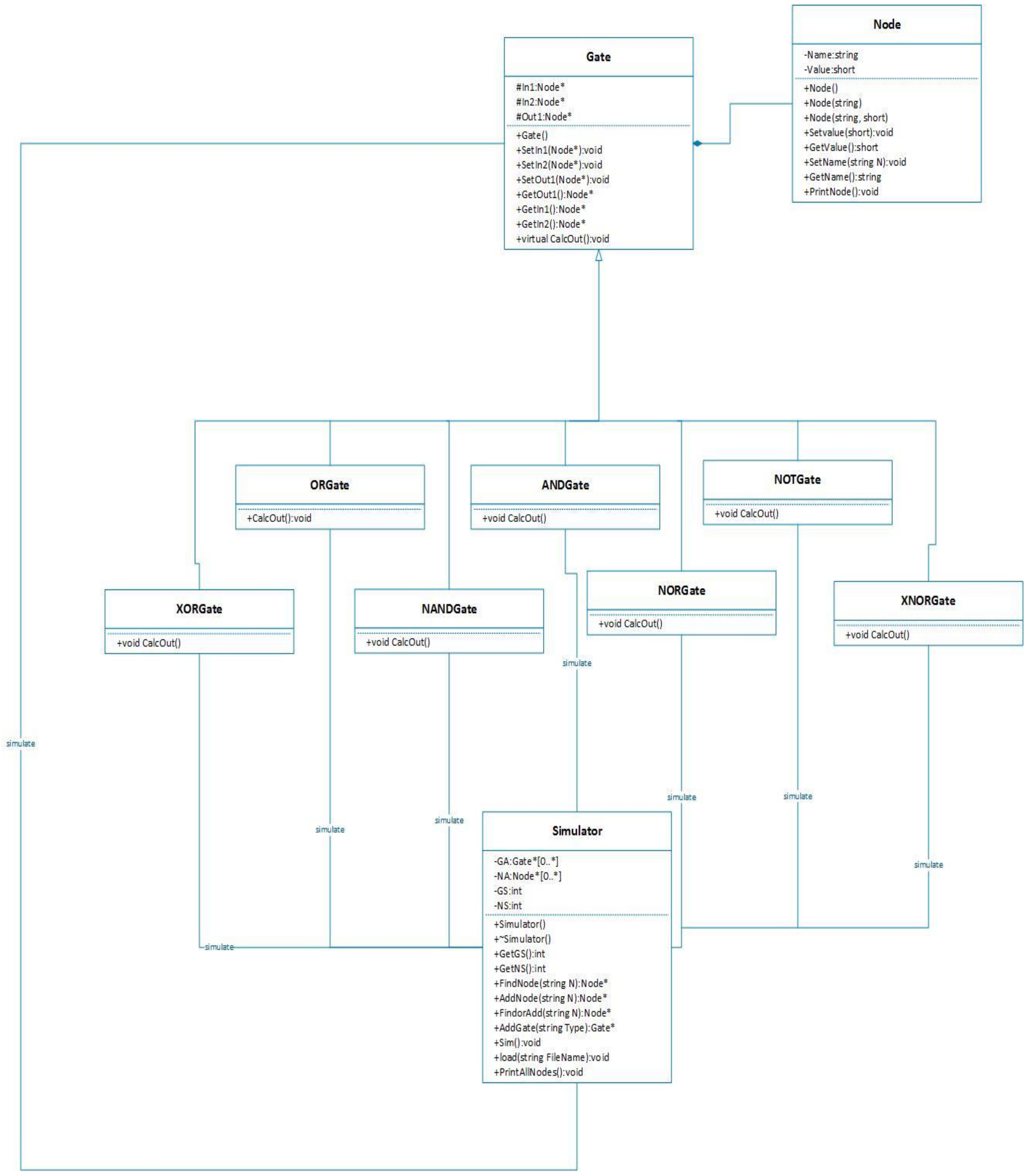
8. State Diagram



9. Sequence Diagram


User
System
User Selects the Input File
Input File is Valid
Enter Required Gates and Nodes
Initialize the Required Nodes
Simulate Order
Tell User that the Simulation is Done
Select Required Nodes to Print
Print the required Nodes

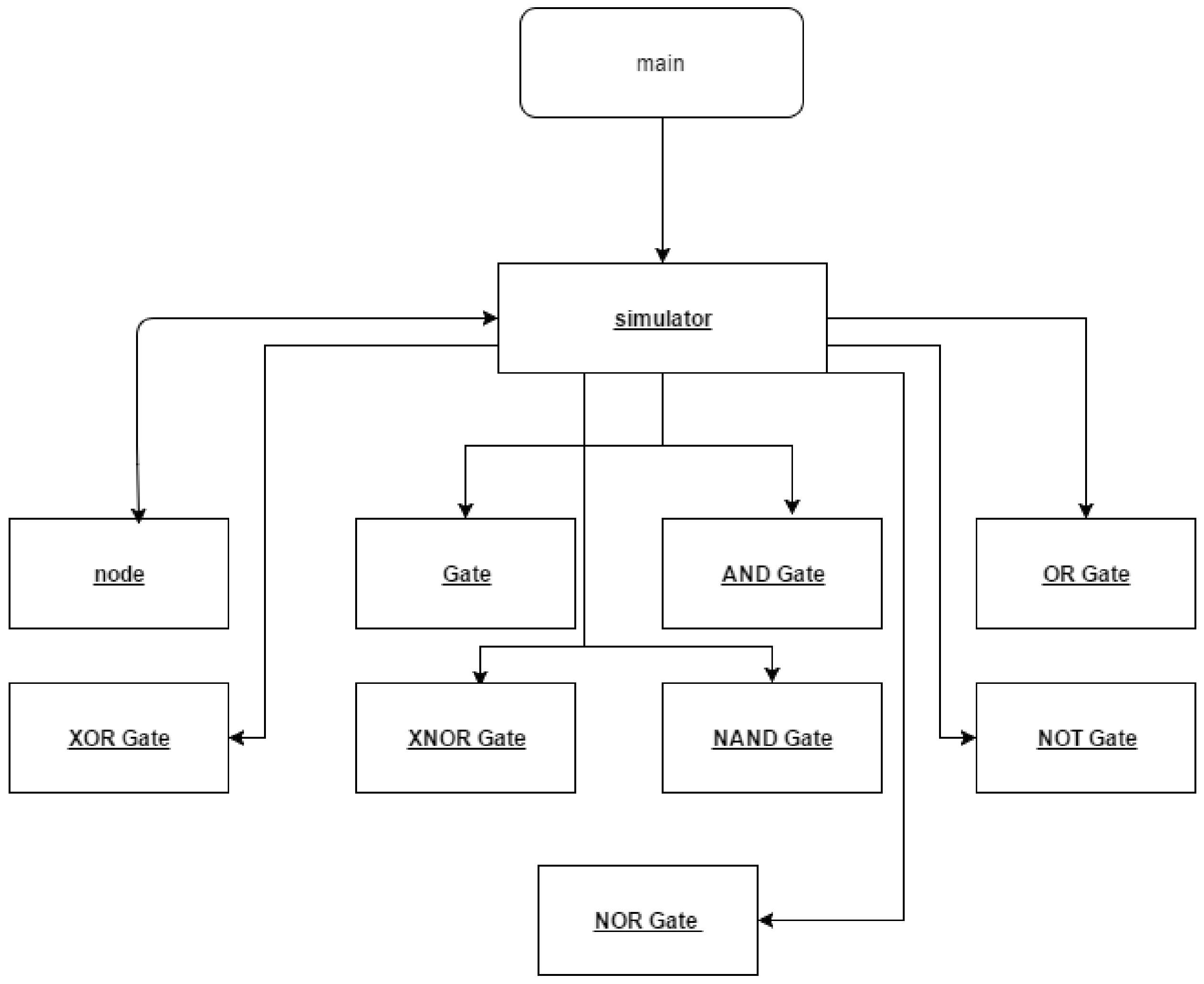
10. Detailed Class Diagram



user-defined data types

```
class Node

{

        string Name;

        short Value;

public:

        Node();

        Node(string);

        Node(string, short);

        void Setvalue(short);

        short GetValue();

        void SetName(string N);

        string GetName();

        void PrintNode();

};



class Gate

{

protected:

        Node* In1;

        Node* In2;

        Node* Out1;

public:

        Gate();

        void SetIn1(Node*);

        void SetIn2(Node*);

        void SetOut1(Node*);
        Node* GetOut1();

        Node* GetIn1();

        Node* GetIn2();

        virtual void CalcOut() = 0;


};
```

11. Client-Object Relation Diagram



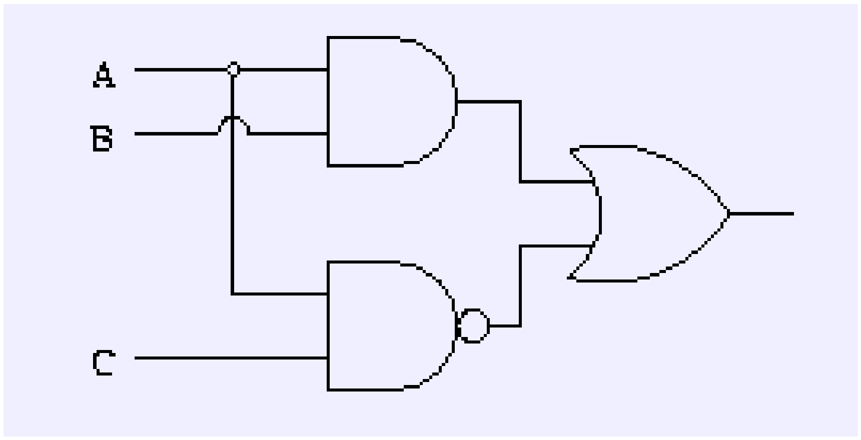
12. Detailed Design

Using PDL Language

```
Start
Get input from user
IF (valid input)
{
Creating required gates
Initialize gates values
Checking input logic
Calculating
Displaying output
}
Else
Displaying error message
End
```

13. Testing

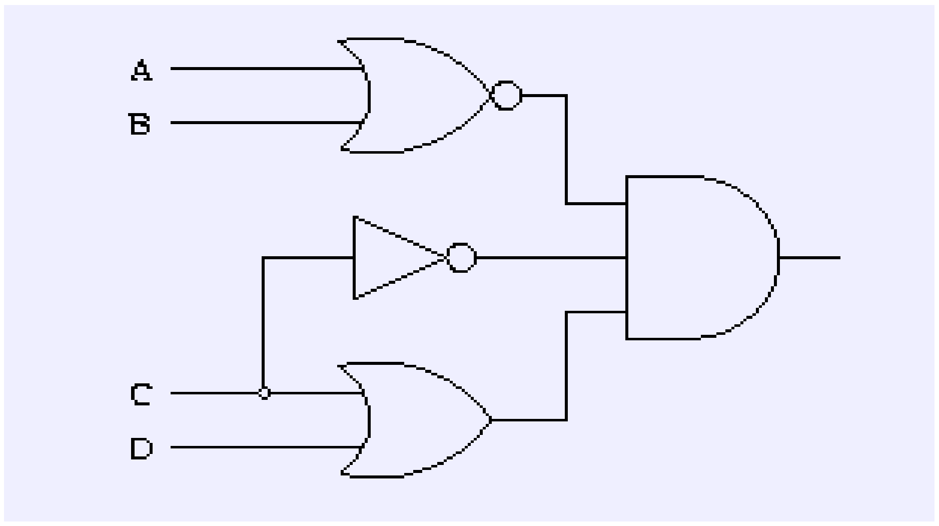
Test 1:



Code:

```
AND A B D       // D is the output for AND
NAND A C E    // E is the output for NAND
OR D E F        // F is the output for OR
SET A 0         // Give value to input A
SET B 1         // Give value to input B
SET C 1        // Give value to input C
SIM               // start simulation
OUT F           // output will be 1
OUT ALL      // output for all nodes
```

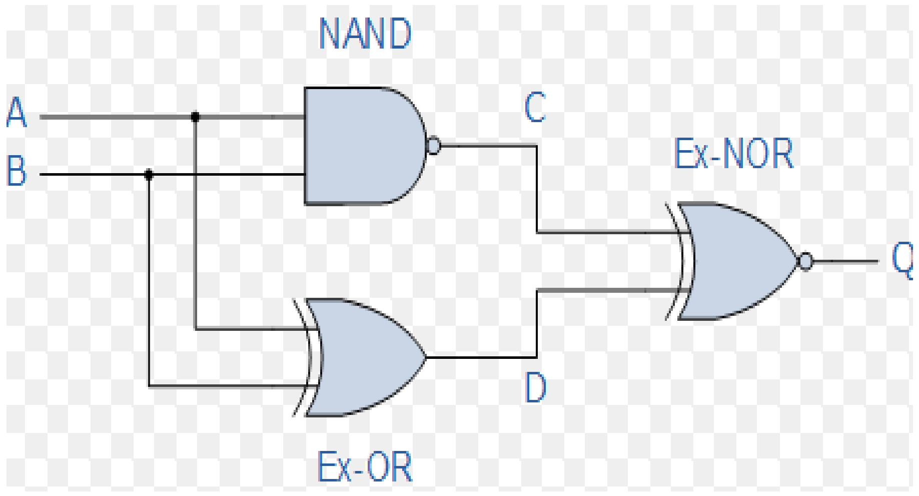
Test 2:



Code:

```
NOR A B E     // E is the output for NOR
NOT C F       // F is the output for NOT
OR C D G      // G is the output for OR
AND E F H     // H is the output for AND
AND H G I     // I is the output for AND
SET A 1       // Give value to input A
SET B 0       // Give value to input B
SET C 0       // Give value to input C
SET D 1       // Give value to input D
SIM           // start simulation
OUT I         // output will be 0
OUT ALL       // output for all nodes
```

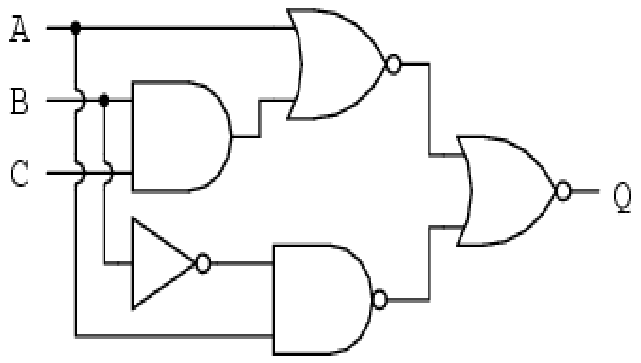
Test 3:



Code:

```
NAND A B C      // C is the output for NAND
XOR A B D       // D the output for XOR
XNOR C D Q      // Q is the output for XNOR
SET A 1         // Give value to input A
SET B 0         // Give value to input B
SIM              // start simulation
OUT Q            // output will be 1
OUT ALL         // output for all nodes
```

Test 4:



Code:

```
AND B C E       // E is the output for AND
NOR A E D       // D is the output for NOR
NOT B F         // F is the output for NOT
NAND A F G      // G is the output for NAND
NOR D G H       // H is the output for NOR
SET A 0         // Give value to input A
SET B 1         // Give value to input B
SET C 1         // Give value to input C
SIM             // start simulation
OUT H           // output will be 0
OUT ALL         // output for all nodes
```

Test Table

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	PASS/FAIL
Test 1 Input A =0 Input B =1 Input C =1	1	1	Pass
Test 2 Input A =1 Input B =0 Input C =0 Input D =1	0	0	Pass
Test 3 Input A =1 Input B =0	1	1	Pass
Test 4 Input A =0 Input B =1 Input C =1	0	0	Pass

14. Estimate Project Cost

ManMonth=0.25

ManMonth * 160 Hrs * 30$ = 0.25*160*30$= 1200$

15. User Guide

This user Guide contains 2 sections. please read them carefully.

Section1 “how to use this simulator”

Section2 “Error list”.

<u>**Section1:”how to use this simulator”**</u>

The logic Gates simulator is simple simulator that is easy to use.

You shall write your design using simple Notations (litters) and write it in text file.

To create new design:

- **Section1.1: describing your circuit design**

1. Open txt file (usually open notepad file)
2. Start typing you design firstly write Gate Name.
3. Use upper Case litters to create Gate like (AND , OR ,NOR , NAND , NOT , XOR , XNOR)
4. secondly type input and output nodes name
    - input1 node name then
    - input2 node name then
    - output node name.

example1.1: AND A B C

here AND is gate name, A input1, B input2, C output

(except for NOT gate input1 then output)

5. Use Command SET NodeName NodeValue to set Nodes values

Example1.2: SET A 1

6. Repeat Step 3, 4, 5 till finish describing your circuit design.

7. Use Command SIM to simulate your design.

8. After Using Command SIM you can print the desired Node name and value Using Command OUT NodeName

9. Use command OUT ALL to print all nodes names and values.

10. Save the text file after finish typing your design.

Section 1.2: running the simulator

11. Change file name to test.cir (the name must be test.cir so the simulator can run it)

12. copy the file

13. open simulator project folder

14. open debug folder

15. Paste the file test.cir (the file test.cir must be in the same directory as .exe file)

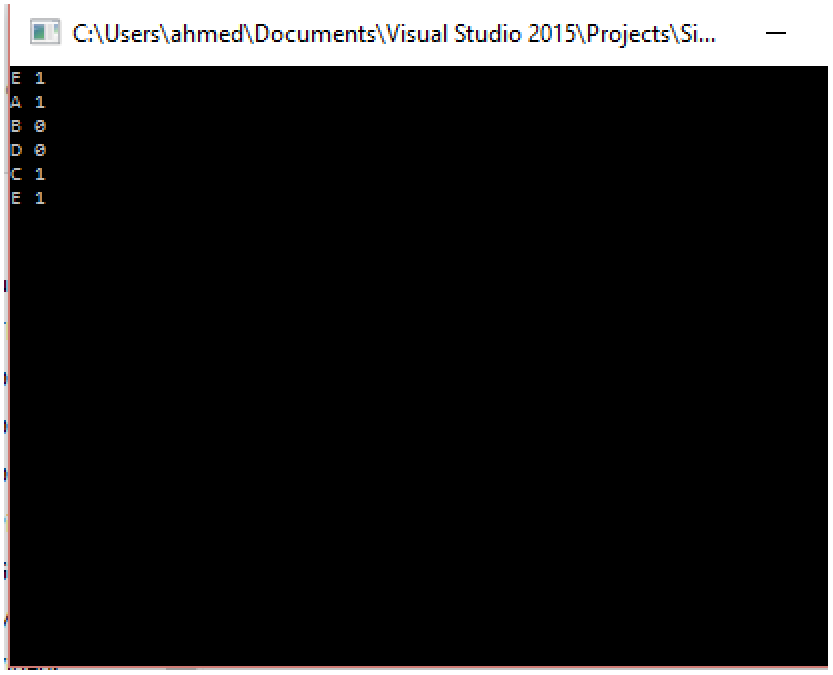
16. Double click on Simulator Project.exe and the program will start simulating your design

.

Section 1.3:example (input and output)

Example1.3:

```
AND A B D

OR C D E

SET A 1

SET B 0

SET C 1

SIM

OUT E

OUT ALL
```



C:\Users\ahmed\Documents\Visual Studio 2015\Projects\Si...

```
E 1
A 1
B 0
D 0
C 1
E 1
```

Figure 1-1 (output for example 1.3)

Section 2: Errors List:

1. Error 00:
   error 00 means that user has enter wrong Gate Name so please check all Gate name
   Remember all Gate Names Must be in Upper Case Litters

   error 00 may also appear if there is an empty line (line with no letter) in the file test.cir

   error 00 may also aprear if the file test.cir is not in the right place, so please refer to step 11 to 15.

2. Error 01:
   Node values must be either 0 or 1
   error 01 means that user is trying to set Node Value to a value that is neither 0 nor 1 (any value that is not 0 or 1).