T.C.
BAHÇEŞEHİR UNIVERSITY

FACULTY OF ENGINEERING AND NATURAL SCIENCES

AN OPTIMIZATION-BASED SUDOKU/KAKURO PLAYER

Capstone Project Final Report

Şule Akça (SEN)
Ecem Altınel (SEN)
Ecem Altınel (INE)

Advisors:
Assoc. Prof. Tankut Atan
Prof. Dr. Mehmet Alper Tunga

ISTANBUL, June 2020

STUDENT DECLARATION

By submitting this report, as partial fulfillment of the requirements of the Capstone course, the students promise on penalty of failure of the course that

- they have given credit to and declared (by citation), any work that is not their own (e.g. parts of the report that is copied/pasted from the Internet, design or construction performed by another person, etc.);
- they have not received unpermitted aid for the project design, construction, report or presentation;
- they have not falsely assigned credit for work to another student in the group, and not take credit for work done by another student in the group.

ABSTRACT

AN OPTIMIZATION-BASED SUDOKU/KAKURO PLAYER

Şule Akça (SEN)
Ecem Altınel (SEN)
Ecem Altınel (INE)

Faculty of Engineering and Natural Sciences

Advisors:

Assoc.Prof. Tankut Atan
Prof. Dr. Mehmet Alper Tunga

June 2020

This project aims to provide Sudoku and Kakuro game and solver platform for people to spend quality time and consists of two separate platforms where users can both play Sudoku and Kakuro and see the solution of the any Sudoku or Kakuro game they find on the newspaper or the internet.

Sudoku & Kakuro, which started in February, was completed in May and was created in the Windows Form application image using C # language and Gurobi optimization solver in the background.

As a result of the project, we have 150 Sudoku puzzles, easy, intermediate and hard levels, and a solver that solves any Sudoku, as well as a Kakuro puzzle with 3*3, 4*4 and 5*5 matrix and a solver.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF CONVENTIONS

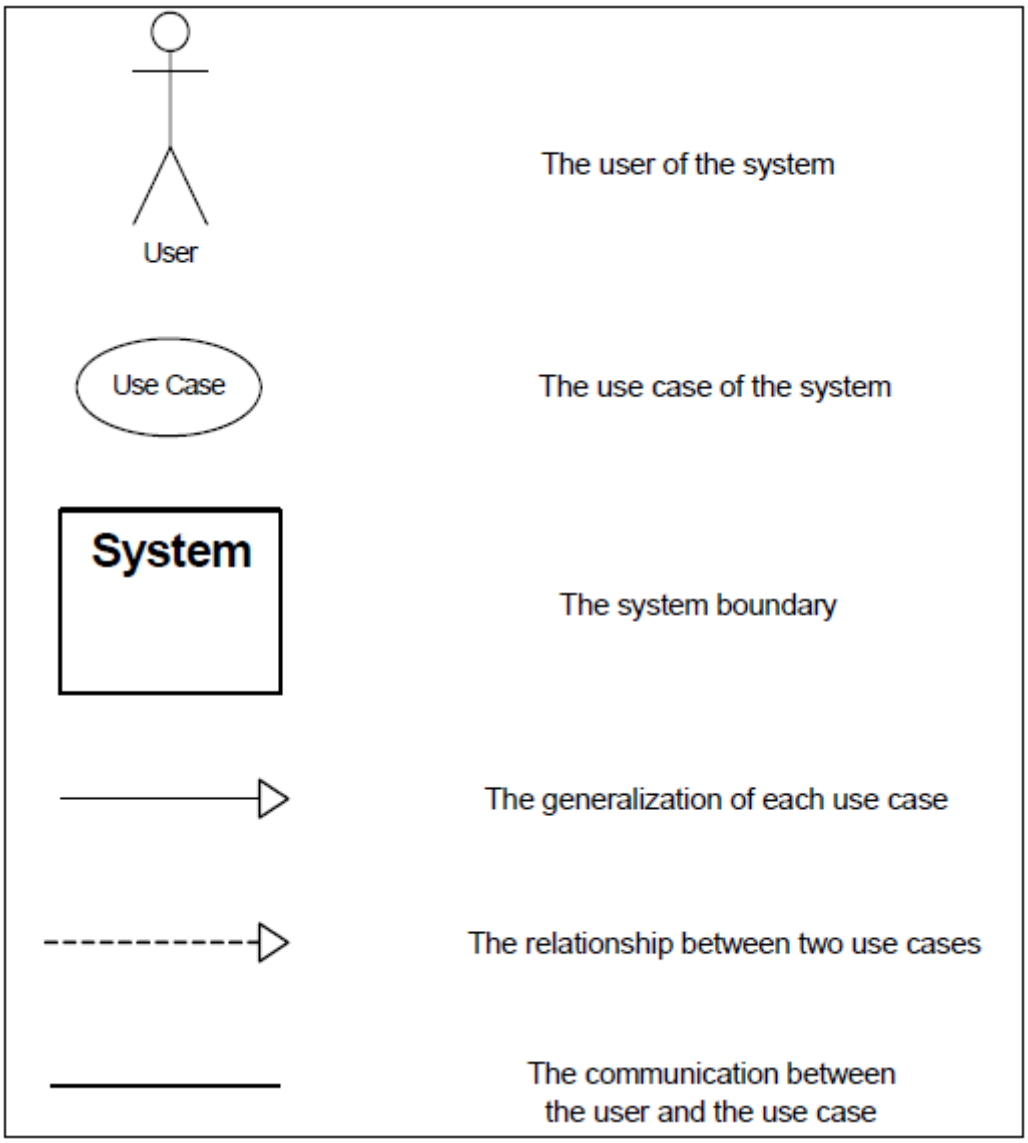


Figure 1. Notation of Use Case Diagram

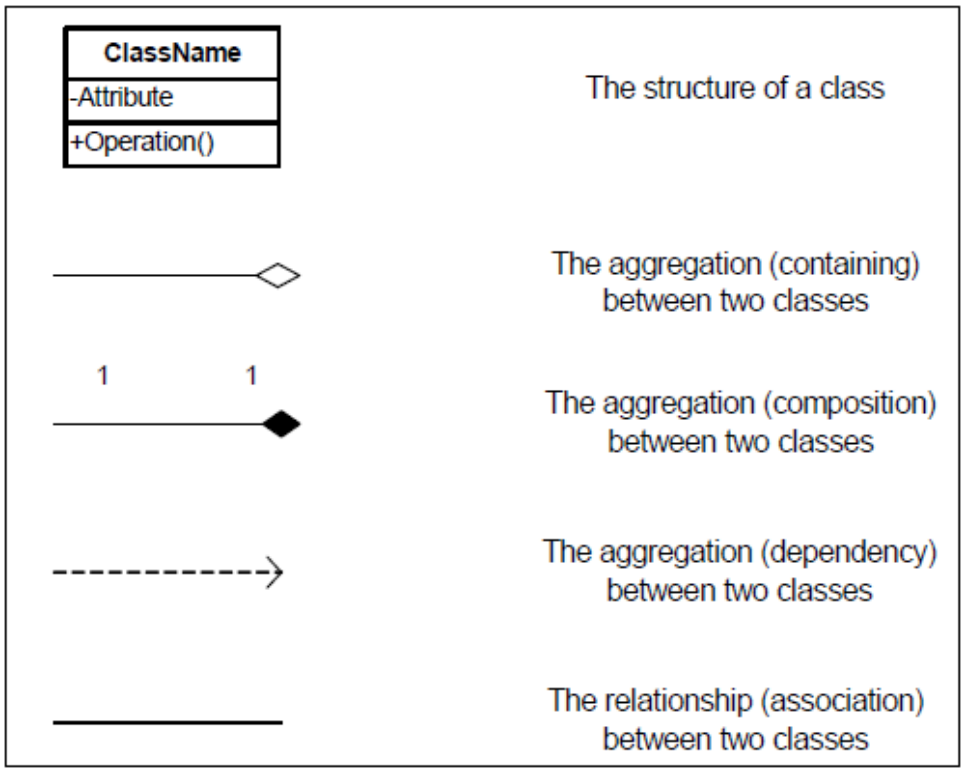


Figure 2. Notation of Class Diagram

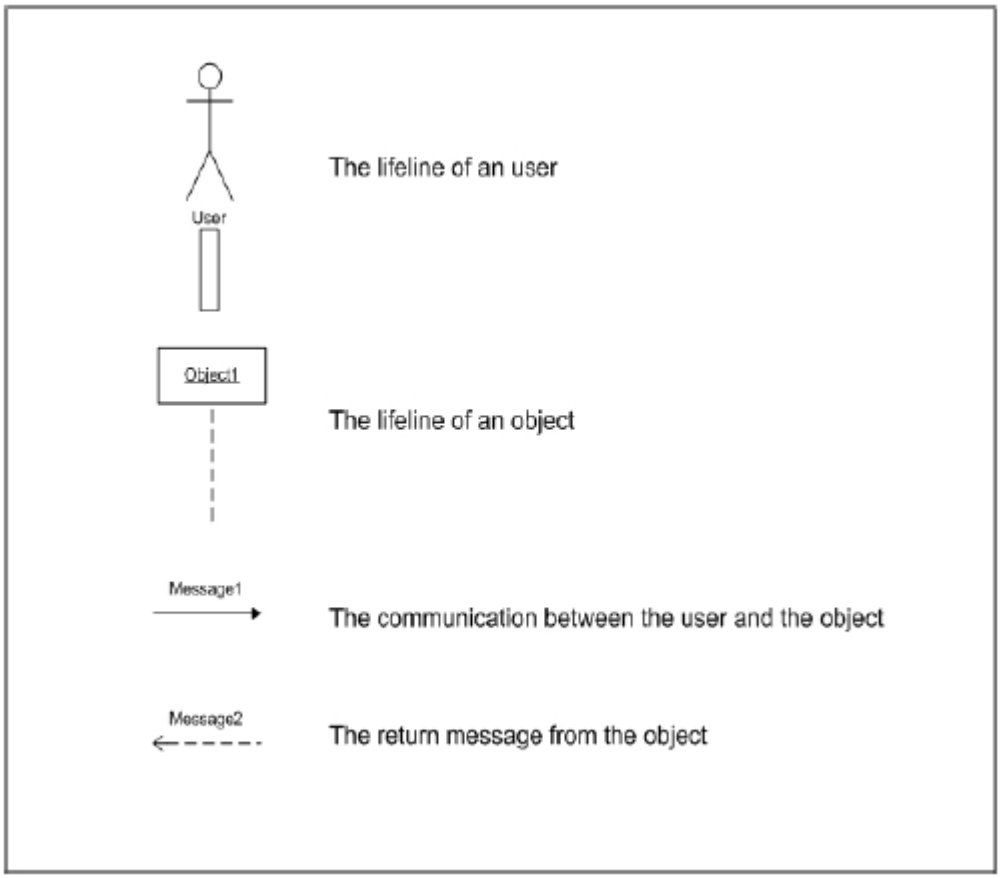


Figure 3. Notation of Sequence Diagram

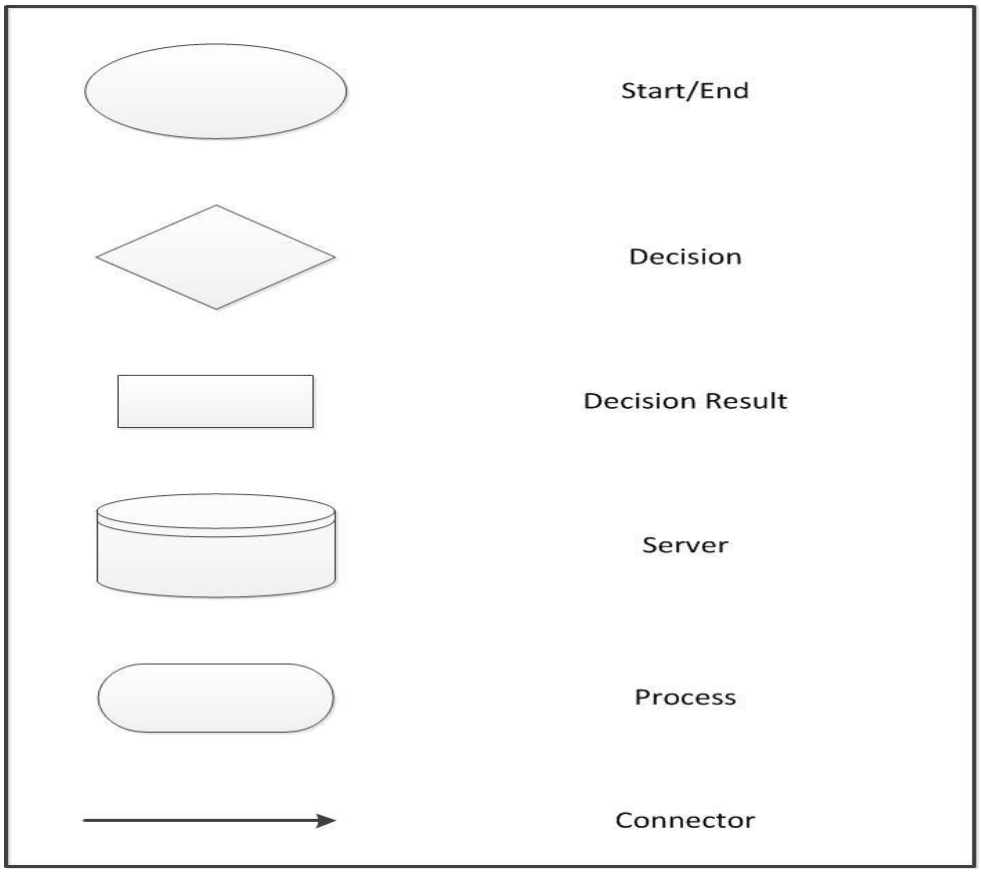


Figure 4. Notation of Data Flow Diagram

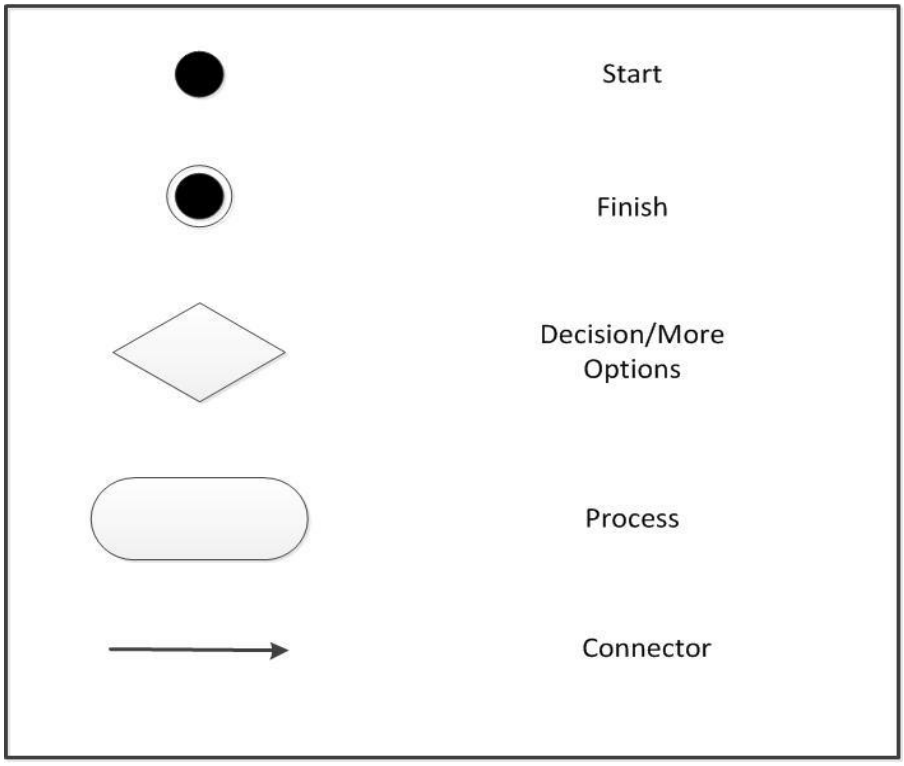


Figure 5. Notation of Process Diagram

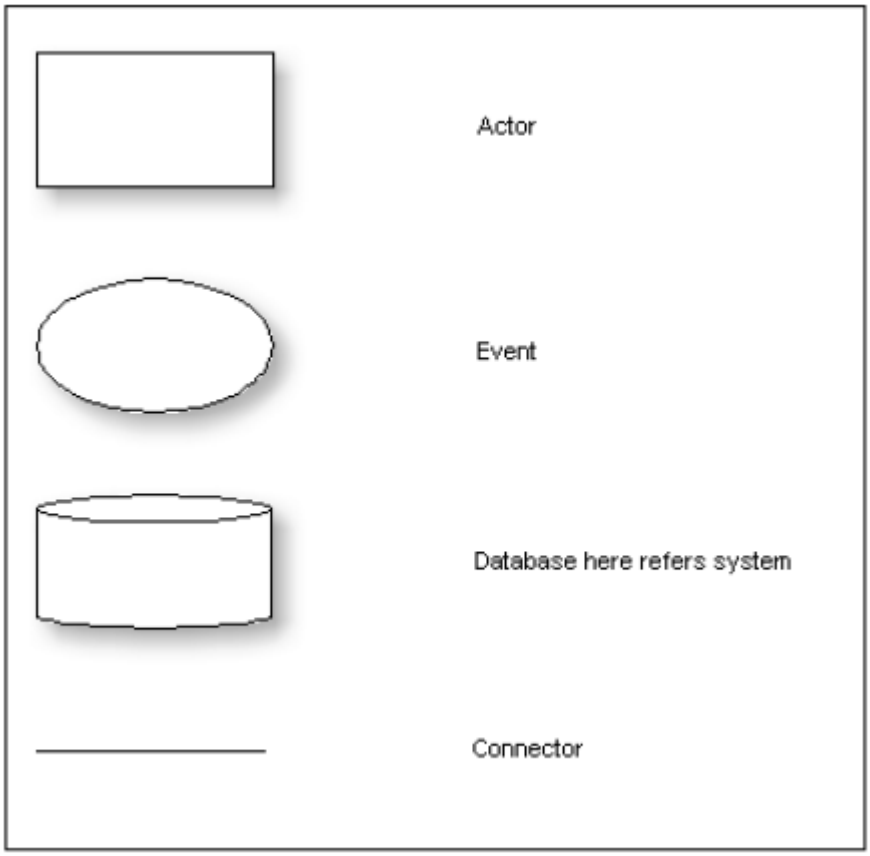


Figure 6. Notation of Design/Block Diagram

1. OVERVIEW

The general subject of this project is to prepare both a game and a solver platform for Sudoku and Kakuro. The group consists of 3 people. Students interested in the software side Şule Akça and Ecem Altınel prepared and edited Sudoku and Kakuro game in Windows form image using C # language. Ecem Altınel, the industry student, created the Sudoku solver with the optimization codes available on the internet and also prepared the Kakuro mathematical model and solver herself.

1.1. Identification Of The Need

Sudoku and Kakuro are two of the most preferred game to challenge the mind yet sometimes it can be hard to find the solution of Sudoku on newspapers within the same day or display the full solution on the website besides the hints.

Hence in this Project it is aimed to create an application which offers both the game and its solutions or the solution of any Sudoku and Kakuro.

1.2. Definition Of The Problem

It is well known that there are similar websites to this project. However, some of them provide only the game but not the solution. Although there are the ones which provide both game and the solution, they do not have any option which enables the player to solve external game. The distinctive objective and result of this project is to provide the external game solutions by using mathematical model.

1.3. Standards and Constraints

- Environmental effects: None
- Social effects: Boost self-confidence
- Economic effects: None
- Ethical issues: None
- Health and safety: Improves memory

1.4. Conceptual Solutions

There are both game and solution sites for Sudoku and Kakuro available on the Internet, but the number of sites that contain both of them is very low or not available. This platform brings both game and solver together.

While creating this project, the game sites on the internet were examined and it was determined which features were used the most. After these features were determined, extra features were added to the project if they were necessary. Therefore, this project is more advantageous than others in terms of both containing general features on the internet and providing extra optimization solver.

1.5. Physical Architecture

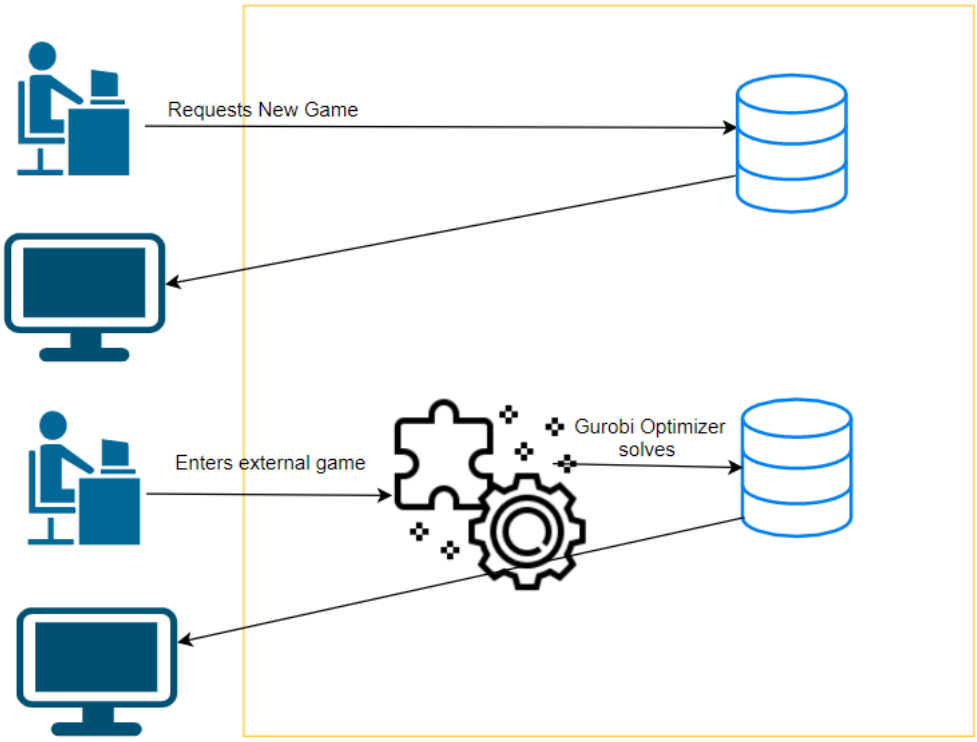


Figure 1.1 Pyhsical Architecture

2. PROJECT WORK PLAN

2.1 Major Deliverables

1. Software: Web page, game design (Visual Studio) and integrating Database (SEN Department).
2. Industrial: Mathematical model and integer programming (INE Department).

2.2 Responsibility Matrix For The Team

Table 1. Project Task Responsibility Matrix

Code	Task	Responsible
0	Final Documentation	Şule,Ecem
1	Creating game database	Şule,Ecem
1.1	Integrate database with application	Şule,Ecem
1.1.2	Mathematical model of SUDOKU	Ecem
1.1.2.a	Mathematical model of KAKURO	Ecem
1.1.2.1	Integer programming at GAMS	Ecem
1.1.2.1.2	Solving with GUROBI	Ecem
1.2	Windows application	Şule, Ecem
1.2.1	Design and features	Şule, Ecem
1.2.1.1	Application design – GridView	Şule,Ecem
1.2.1.2	Application features – Time	Şule,Ecem
1.2.1.3	Application features – Stop	Şule,Ecem
1.2.1.4	Application features – New Game	Şule,Ecem
1.2.1.4.1	Application features – Difficulty level	Şule,Ecem
1.2.1.5	Application features – Extrernal Game	Ecem
1.2.1.6	Application features – Delete	Şule,Ecem
1.2.1.7	Application features – Hint	Ecem
1.2.1.8	Application features – Submit	Şule,Ecem
1.2.1.9	Application features – Save the score	Şule,Ecem
1.2.1.10	Application features – See Solution	Şule,Ecem

2.3 Project Network

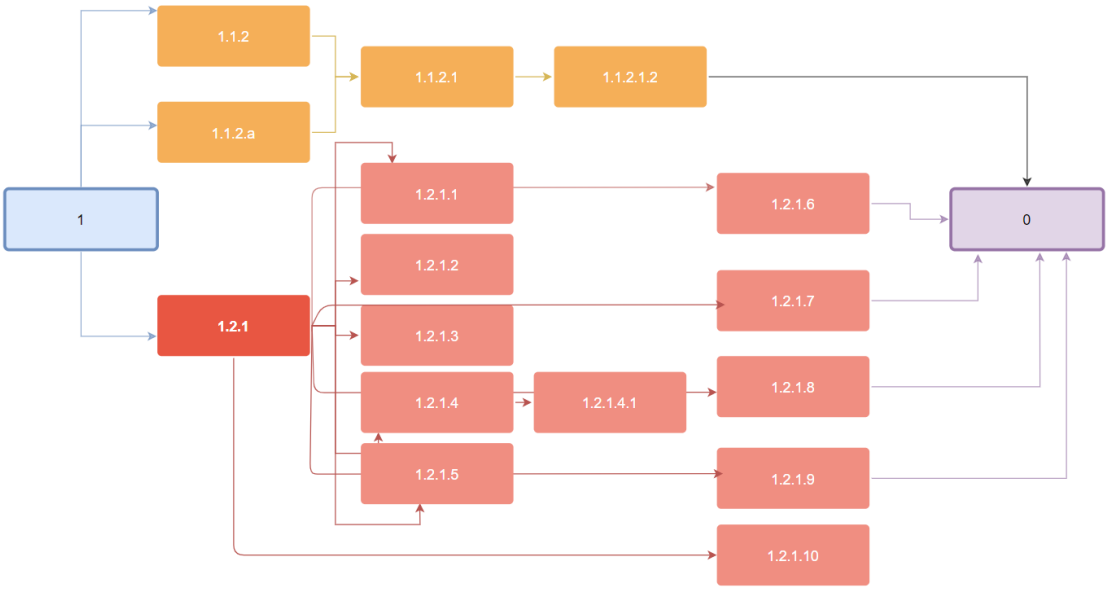
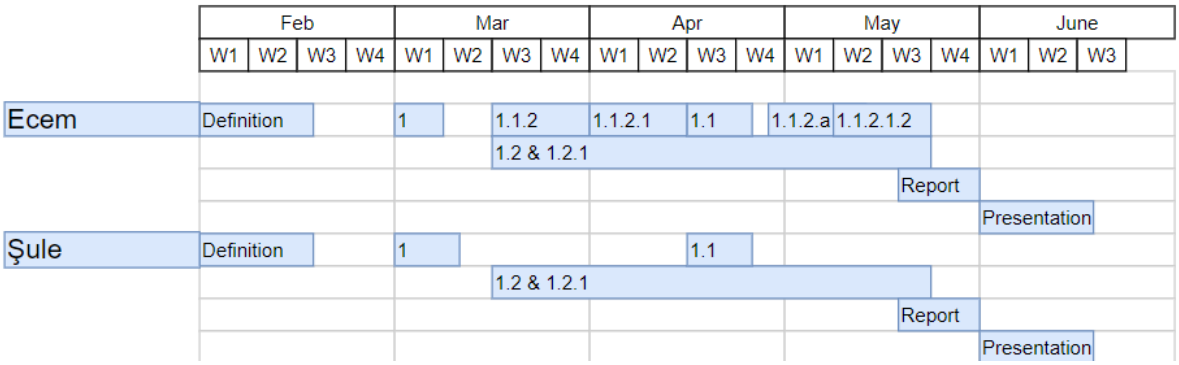


Figure 2.1 Project Network

2.4 Gantt Chart

Table 2. Project Gantt Chart



2.5 Project Interface Design

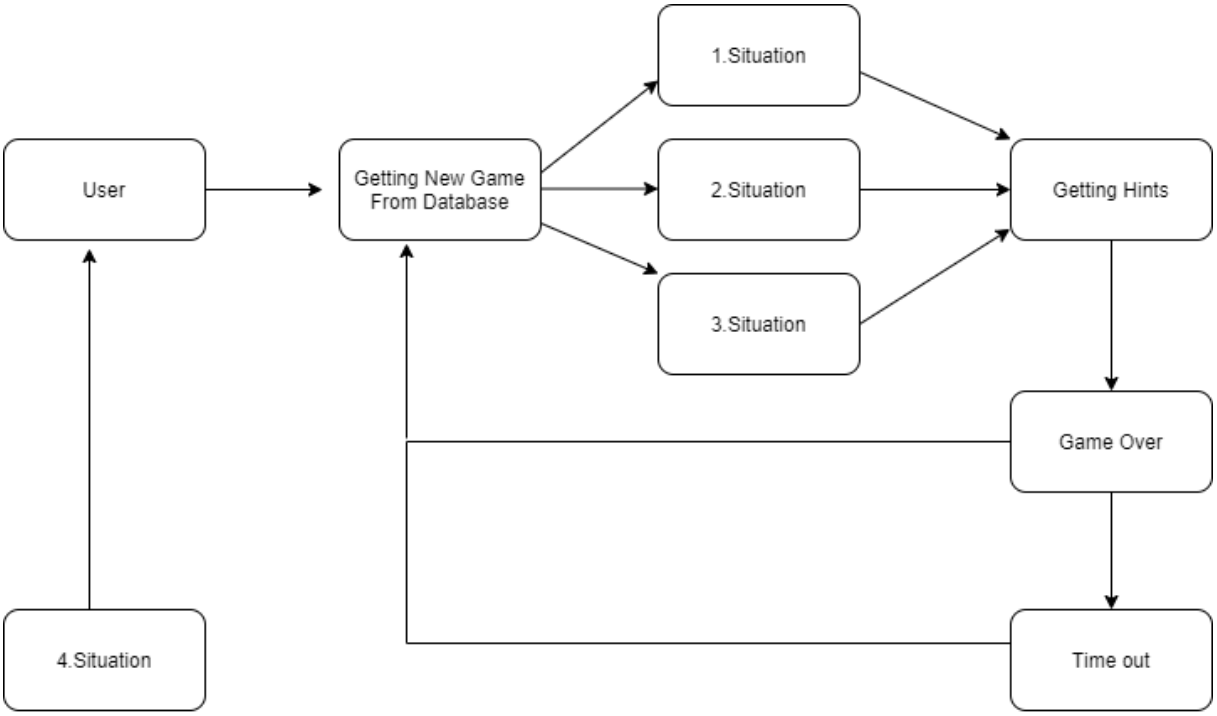


Figure 2.2. Subsystem Interface Design

Situation 1: User chooses easy level.

Situation 2: User chooses medium level.

Situation 3: User chooses high level.

Situation 4: User gets another games' solution.

3. DESIGN PROCESS

3.1 Industrial Engineering

3.1.1 Definition of the Problem

Sudoku and Kakuro are two of the most preferred logic game to challenge the mind yet sometimes it can be hard to find the solution on newspapers within the same day or display the full solution on the website besides the hints.

By implementation our project this problem will be solved and users will be able to find proper solutions to any Sudoku game and preformed 3*3, 4*4 and 5*5 Kakuro game they want.

3.1.2. Review of Technologies and Methods

Although the project was thought to proceed using Javascript before starting the, after learning that Gurobi supports languages such as Java, C # and Python, but it does not have a library for the Javascript language, it was decided to develop the games on Visual Studio using C# language.

Then, both games are created in Visual Studio development environment by using C# language and the solution part is handled using Gurobi mathematical optimization solver.

Before implementing Gurobi solver, the solution part was done by first creating a mathematical modelling and then converting this modelling into code using the Gurobi library.

3.1.3. Review the Availability of the Data

Since each website has its own database and these are difficult to access without permission, the problem data for the games are only taken from public databases such as Github and QQWing Sudoku Generator.

The fact that this site can generate the desired number of problems instantly according to the desired difficulty level for Sudoku problems has created positive effects on the pace of the project.

Even if the variety of problem data on the site is sufficient, it had to be rearranged accordingly for the project by using the editing method. Below editing method and before/after format will be seen.

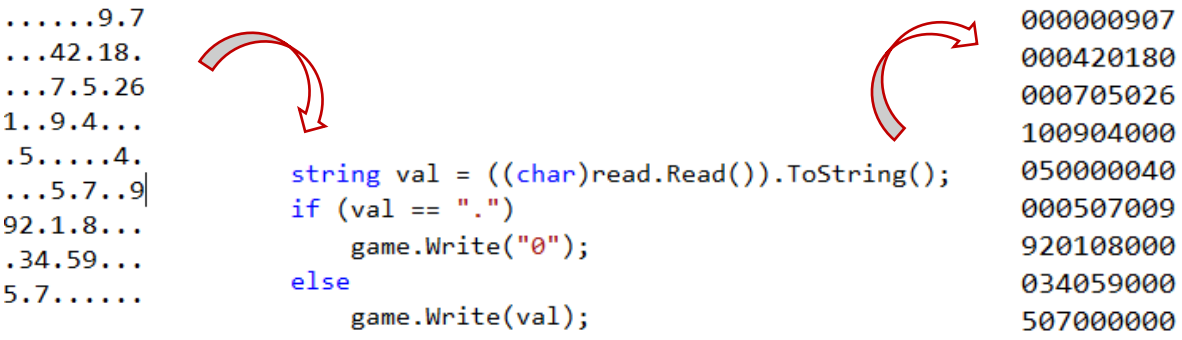


Figure 3.1. Original and Rearranged Sudoku Puzzle

3.1.4. Conceptualization

At the end of the project, both Sudoku and Kakuro games will emerge and at the same time in accordance with the purpose of the project, there will be a optimization solver in order to solve the games other than the games in database.

Game will appear after choosing the difficulty level and click start button and if user wants to solve an external game all s/he has to do will be choosing External Game section, clicking start button (not required for Kakuro game) and fill the game table in accordance with the problem that s/he wants to solve. After filling the table, solve button has to be clicked and the game will be solved.

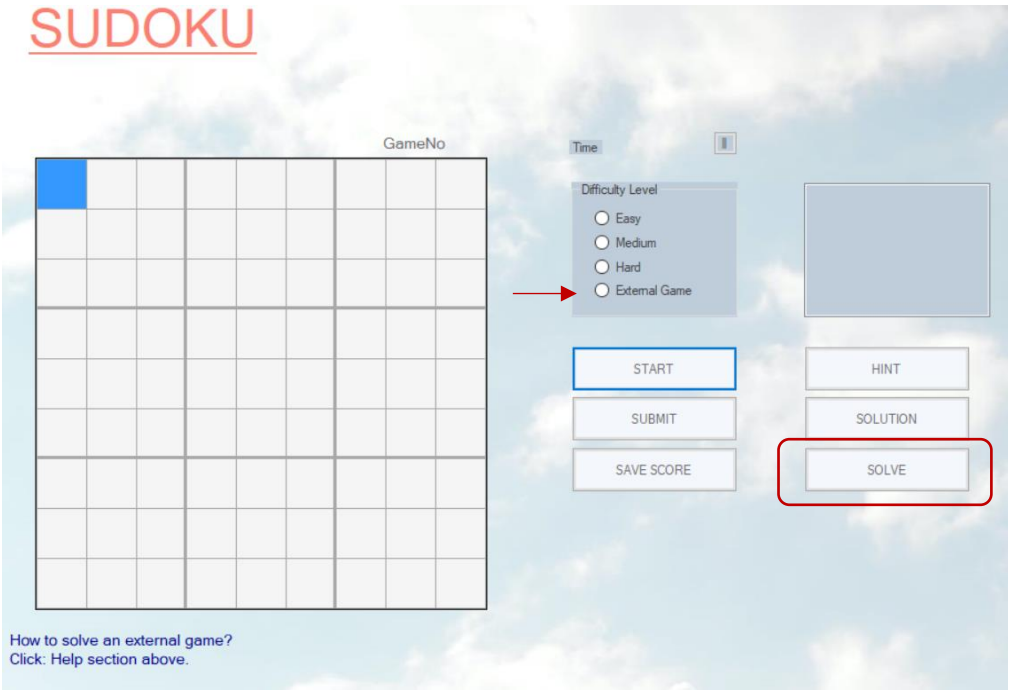


Figure 3.2. Sudoku Homepage Solve Button

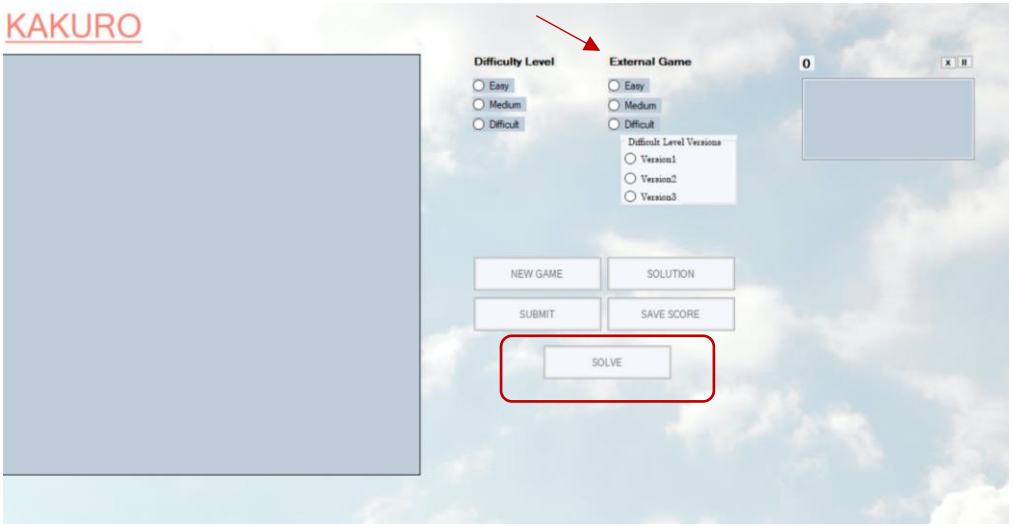


Figure 3.3 Kakuro Homepage Solve Button

3.1.5. Methodological Architecture

Solutions to Sudoku and Kakuro games can be found by solving IP models. Although there are several commercial solvers available to solve IP problems exactly such as Gurobi, FICO Xpress, IBM ILOG Cplex etc. to solve the problem Gurobi optimization solver is used by reason of the fact that instances are small and can be solved quickly. Below is the diagram that shows the inputs and outputs to the solver.

Figure 3.4. Diagram of Inputs/Outputs

3.1.6. Risk Assessment

Table 3. Project Risk Assessment

Task	Associated Risk	Severity	Probability	Risk Score	Management of the Risk
Creating Sudoku model	Wrong model will cause an error	Critical	Low	Medium(6)	Code can be checked with gams.
Creating Kakuro model	Wrong model will cause an error	Critical	Medium	High(8)	Code can be checked with gams.
Creating Sudoku optimization code	Miswritten code will give error.	Critical	Low	Medium(6)	Code can be checked with the logic of modelling algorithm.
Creating Kakuro optimization code	Miswritten code will give error.	Critical	High	High(9)	Code can be checked with the logic of modelling algorithm.
Integration of Gurobi into Visual Studio	In case of being not integrated Gurobi library will not be recognized	Catastrophic	Medium	Extreme(11)	Gurobi support forum

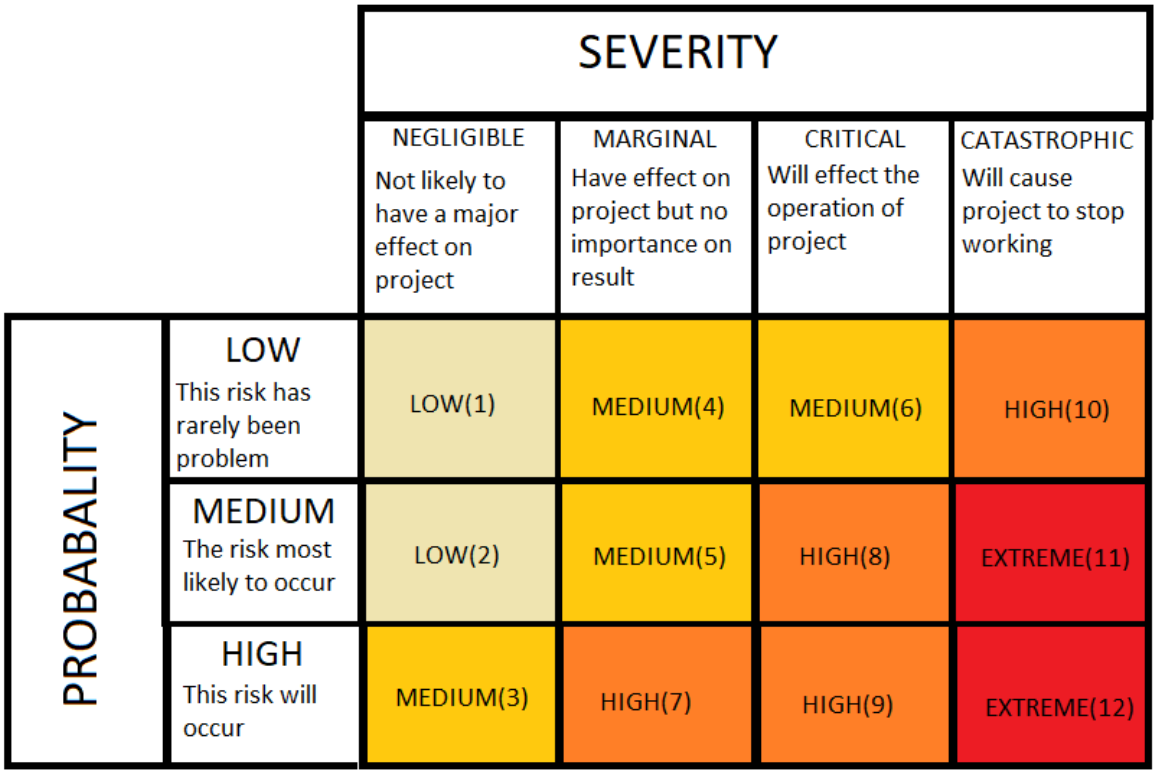


Figure 3.5. Risk Matrix of Project

3.1.7. Materialization

Since the project has been determined to be in the form of a computer program from the definition stage, there is no tangible material other than Visual Studio and Windows Form Application created using C # language.

However, apart from that, there is a game program that the user can use as long as the computer is with him/her at any time of the day, and also a solution tool to solve the games s/he sees in magazines or newspapers. It can work offline, it does not need internet. As it has an easy interface, it can be easily played by players of all ages and there is a description section especially for external games.

3.1.8. Evaluation

After finding enough problems for a game to be played for a long time, these problems were successfully arranged according to the format of the solver and each game was placed in a different file, each level in a different folder.

Sudoku and Kakuro have been converted into code after mathematical modelling and written by using the Gurobi library. After each problem is solved by using Gurobi optimization

solver, they are separated and placed according to their level in the solution folder,in this way the answer part has also been successfully created.

Since the entire problem and solution database has been created, the pre-created Sudoku and Kakuro game platform has been tested and found to be running smoothly.

3.1.9. Discussion of outcome

3.1.9.1 Theoretical

At the end of the project, it is planned to obtain a Sudoku and Kakuro game platform, written in Javascript and created using an optimization tool, that the user can play on a website. Also on this platform, users will be able to solve any Sudoku or any Kakuro game they want with one click and see the answer.

Sufficient problems will be prepared for long-term play of the game and their solution will be written to be accessible at all times. To make the game exciting, the leaderboard and timer will be added so that the user will see his/her own progress.

3.1.9.2 Practical

At the end of the project, since the Gurobi optimization tool does not support JavaScript, the project was written in C # and created using the Gurobi, also the Sudoku and Kakuro game platform that the user can play through a Windows Form was obtained. In this platform, users will be able to solve any Sudoku or 3*3 ,4*4 and 5*5 matrix Kakuro game with a single click using a solver and see the answer.

Sufficient problems have been prepared for the long-term play of the game and their solutions have been filed to be accessible at all times. To make the game exciting, the leaderboard and timer have been added so that the user can record his/her own progress. In addition, when the game is completed, users will be able to see their correct and incorrect numbers and where they did wrong.

3.1.10. Discussion of Limitations

As mentioned above, the project was not created as a website because Gurobi does not support Javascript. However, this was not an obstacle for the project. It is created not as Website but Windows Application.

While there are no restrictions for the Sudoku game, the Kakuro changing shape at all levels and had at least 10 shapes caused a glitch on the solver side. Therefore, it was decided to create and solve only the first 3 shapes instead of solving all the shapes.

The model of Sudoku:

Decision variable:

X^a_{ij}=1, if the number a is placed in row i, column j

A,i,j constraints

The rules of Sudoku:

i = 1 to 9 (in coding 0-8)

j = 1 to 9 (in coding 0-8)

a = 1 to 9 (in coding 0-8)

Each cell must take exactly one value

$$\sum_{a=1}^{9} X_{ij}^{a} = 1, \forall \text{i, j}$$

Each value is used exactly once per row

$$\sum_{j=1}^{9} X_{ij}^{a} = 1, \forall \text{i, } \forall \text{a}$$

Each value is used exactly once per column

$$\sum_{i=1}^{9} X_{ij}^{a} = 1, \forall \text{j, } \forall \text{a}$$

Each value is used exactly once per 3x3 sub-grid

$$\sum_{i=3q-2}^{3q} \sum_{j=3p-2}^{3p} X_{ij}^{a} = 1 ,\forall \text{a p,q= 1 to 3}$$

The model of Kakuro:

Decision variable:

X^k_{ij}=1, if the number a is placed in row i, column j

Else X^k_{ij}=0.

K,i,j constraints

The rules of Kakuro:

i = depends on the difficulty level of the game

j = depends on the difficulty level of the game

k = 1,2,3,4,5,6,7,8,9

V_c = depends on the game and columns

V_r = depends on the game and rows

Each cell must take exactly one value

$$\sum_{k=1}^{9} X_{ij}^{k} = 1, \forall i, j$$

Each value is used at most once per row

$$\sum_{j=1} X_{ij}^{k} \leq 1, \forall i, \forall k$$

Each value is used at most once per column

$$\sum_{i=1} X_{ij}^{k} \leq 1, \forall j, \forall k$$

Sum of column values = prespecified value (V_c)

$$\sum_{i=1} \sum_{k=1} k X_{ij}^{k} = Vc, , \forall j$$

Sum of row values = prespecified value (V_r)

$$\sum_{j=1}\sum_{k=1} kX_{ij}^{k} = \text{Vr}, ,\forall \text{i}$$

Gurobi Integration

After downloading the Gurobi optimization solver, Visual Studio should be opened to load the Gurobi library as an analyzer in the References panel. Since Gurobi supports different languages, the library should be picked properly, otherwise system will not recognize its properties.

After adding the library to the References panel, configuration manager should be reconstructed and platform should be set from ANY CPU to x64 or another platform. In the absence of this part is missing, the system will give an "BadImageFormatException" error.

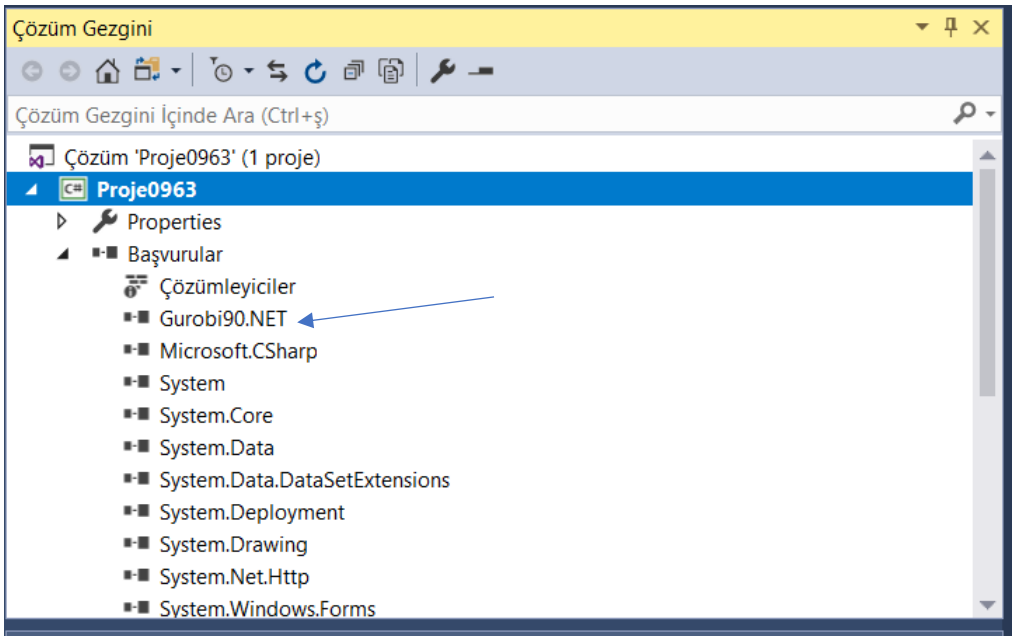


Figure 3.6. Gurobi Integration in Visual Studio

Problem Resolution Time (Kakuro)

Table 4. Resolution Time of Kakuro

Level	Version	Time (seconds)
Easy	-	0.00
Medium	-	0.01
Hard	V1	0.01
Hard	V2	0.01
Hard**	V3**	0.01**
External		~0.01

**

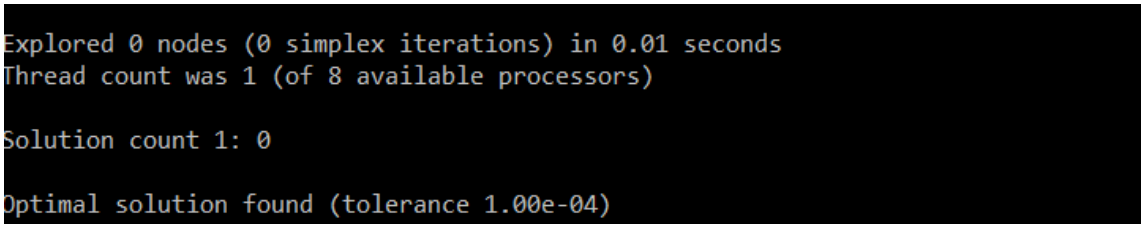
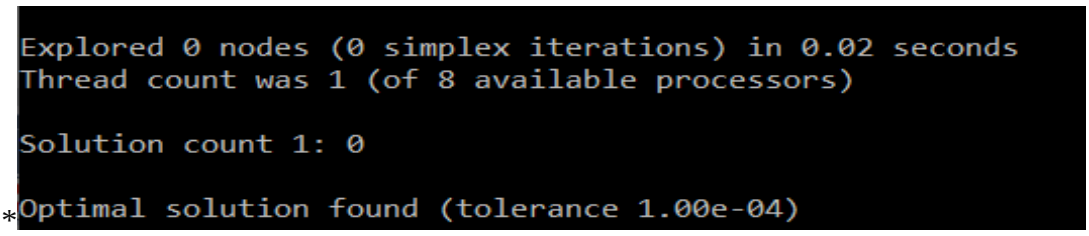


Figure 3.7. Resolution Time of Kakuro

Problem Resolution Time Sudoku

Table 5. Resolution Time of Sudoku

Level	Time (seconds)
Easy	Mostly 0.01*
Medium	Mostly 0.01*
Hard	Mostly 0.01*
External	~0.01



*

Figure 3.8. Resolution Time of Sudoku

By mostly 43 solutions from 50 problems is meant. So as it is seem above, approximately 7/50 of the problems are being solved within 0.02 seconds.

The codes of Sudoku

In the first step, the three-dimensional array of the model variables is created before moving on to the contraints.

int n=9;

int s=3; // for the 3*3 subgrid

int sıra =0 to 50;

string tür = “Easy” , “Medium” or “Hard”

```
try
{
    GRBEnv env = new GRBEnv();
    GRBModel model = new GRBModel(env);

    // Create 3-D array of model variables

    GRBVar[,,] vars = new GRBVar[n, n, n];

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            for (int v = 0; v < n; v++)
            {
                string st = "G_" + i.ToString() + "_" + j.ToString()
                                 + "_" + v.ToString();
                vars[i, j, v] = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, st);
            }
        }
    }
```

Like all other optimization codes, the Gurobi environment is created and a model is added to it. Then 81 three-dimensional variables are added to the model in a way that row will be called as i, column as j and value of cell as v. It is important that the variables are binary. This gives us ease of operation on which cell will get which value.

vars[row, column, value] = model.AddVar(LowerBound, UpperBound, Objective coefficient for new variable, GRB.BINARY (type),name of the variable);

Each cell must take exactly one value

```
// Add constraints

GRBLinExpr expr;

// Each cell must take one value

for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        expr = 0.0;
        for (int v = 0; v < n; v++)
            expr.AddTerm(1.0, vars[i, j, v]);
        string st = "V_" + i.ToString() + "_" + j.ToString();
        model.AddConstr(expr == 1.0, st);
    }
}
```

$$\sum_{a=1}^{9} X_{ij}^{a} = 1, \forall i, \forall j$$

The for loop at the top is the model turned into code.

Here,9 values that every ij cell can take are added to each other and said that it is equal to 1. Since the values are binary, this means that only one of them may be valid. Others will be 0.

Each value is used exactly once per row

```
// Each value appears once per row

for (int i = 0; i < n; i++)
{
    for (int v = 0; v < n; v++)
    {
        expr = 0.0;
        for (int j = 0; j < n; j++)
            expr.AddTerm(1.0, vars[i, j, v]);
        string st = "R_" + i.ToString() + "_" + v.ToString();
        model.AddConstr(expr == 1.0, st);
    }
}
```

$\sum_{j=1}^{9} X_{ij}^{a} = 1, \forall i, \forall a$

The for loop at the top is the model turned into code.

Here, all column values where a number (v) can be placed in a row are summed and set equal to 1. This means that in a row, for a number there is only one column to be placed.

```
Each value is used exactly once per column
```

```
// Each value appears once per column

for (int j = 0; j < n; j++)
{
    for (int v = 0; v < n; v++)
    {
        expr = 0.0;
        for (int i = 0; i < n; i++)
            expr.AddTerm(1.0, vars[i, j, v]);
        string st = "C_" + j.ToString() + "_" + v.ToString();
        model.AddConstr(expr == 1.0, st);
    }
}
```

$\sum_{i=1}^{9} X_{ij}^{a} = 1, \forall j, \forall a$

The for loop at the top is the model turned into code.

Here, all row values where a number (v) can be placed in a column are summed and set equal to 1. This means that in a column, for a number there is only one row to be placed.

```
Each value is used exactly once per 3x3 sub-grid
```

```
// Each value appears once per sub-grid

for (int v = 0; v < n; v++)
{
    for (int i0 = 0; i0 < s; i0++)
    {
        for (int j0 = 0; j0 < s; j0++)
        {
            expr = 0.0;
            for (int i1 = 0; i1 < s; i1++)
            {
                for (int j1 = 0; j1 < s; j1++)
                {
                    expr.AddTerm(1.0, vars[i0 * s + i1, j0 * s + j1, v]);
                }
            }
            string st = "Sub_" + v.ToString() + "_" + i0.ToString()
                              + "_" + j0.ToString();
            model.AddConstr(expr == 1.0, st);
        }
    }
}
```

$\sum_{i=3q-2}^{3q} \sum_{j=3p-2}^{3p} X_{ij}^{a} = 1$,∀a p,q= 1 to 3

The for loop at the top is the model turned into code.

Here, all row and column values where a number (v) can be placed in a 3*3 subgrid are summed and set equal to 1. This means that in a column, for a number there is only one row to be placed.

Fix variables associated with pre-specified cells:

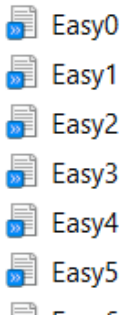
```
// Fix variables associated with pre-specified cells

StreamReader sr = File.OpenText(P+tür+"\\"+tür+sıra+".txt");

for (int i = 0; i < n; i++)
{
    string input = sr.ReadLine();

    for (int j = 0; j < n; j++)
    {
        int val = (int)input[j]-49; // 0-based

        if (val >= 0)
            vars[i, j, val].LB = 0.5; //0dan büyük olması yeter
    }
}
```



Here Sudoku games (0-50) that have been prepared before are extracted one by one and if the cell values in the 9 * 9 matrix are greater than zero, the pre-specified values added to array as val (v).

And since cell values are in the string form, 49 is substracted from to get 0 based value.

Figure 3.9. Text Files of Easy Level of Sudoku

And then model is ready to be optimized

```
}

// Optimize model

model.Optimize();
```

Writing the results into a file

```
double[,,] x = model.Get(GRB.DoubleAttr.X, vars);

StreamWriter doku = new StreamWriter(P2+ tür + "\\" + tür + sıra + ".txt");

for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        for (int v = 0; v < n; v++)
        {
            if (x[i, j, v] == 1.0)
            {
                doku.Write(v+1 ); // it was 0 based
            }
        }
    }
    doku.WriteLine();
}
doku.Close();
```

After transferring the Gurobi arrays to a new array ‘x’, the the data is printed in a proper solution file in the form of 9 * 9 matrix.

This code is generated to solve approximately 150 problems. After solving the problems, solutions are also stored in a different database and during the game the solutions will be extracted from database. During the game only external problems will be solved. Codes for the external game is nearly same as the code above. Only ‘tür’ name is changing.

The model is ready to be closed.

```
// Dispose of model and env
model.Dispose();
env.Dispose();
```

And if any error occurs during the execution, it will be seen with catch method.

```
}
catch (GRBException e1)
{
    Console.WriteLine("Error code: " + e1.ErrorCode + ". " + e1.Message);
    errorLbl.Text = "Error code: " + e1.ErrorCode + ". " + e1.Message;
}
```

Unsolved

```
530070000
600195000
098000060
800060003
400803001
700020006
060000280
000419005
000080079
```

Solved

```
534678912
672195348
198342567
859761423
426853791
713924856
961537284
287419635
345286179
```

Figure 3.10. Unsolved/Solved Sudoku Puzzle

KAKURO CODES:

//The codes to solve an external problem will be explained.

```
int n = 9;
int s; //matrix-1
int version = 3;//for diff
if (radioButtonExEasy.Checked)
{
    s = 2;
}
else if (radioButtonExMdm.Checked)
{
    s = 3;
}
else
{
    s = 4;
    if (radioButtonV1.Checked)
        version = 1;
    else if (radioButtonV2.Checked)
        version = 2;
    else
        version = 3;
}
```

First pre-specified cell values are extracted from problem and hold in an array.

```
////// Fix variables associated with pre-specified cells
    string P = @"C:\PROJE-ECEM\Proje0963\Proje0963\KakuroTxt\";

    StreamReader sr = new StreamReader(P + "kakuroGame.txt");
    double[,] array = new double[s + 1, s + 1];
    string str;
    string[] strArray;
    for (int i = 0; i < s + 1; i++)
    {
        str = sr.ReadLine();
        strArray = str.Split(',');

        for (int j = 0; j < s + 1; j++)
        {

            array[i, j] = Convert.ToDouble(strArray[j]);
            // Console.Write(array[i, j]);
        }
        // Console.WriteLine();

    }
    // Console.ReadLine();
```

Here it is said s,s,n+1 because the maximum value which a cell can get must be 9 and arrays start from 0. Since n is set to 9 at the beginning, in order to get the value 9 too, size of value should be set to n+1;

```
    try
    {
        GRBEnv env = new GRBEnv();
        GRBModel model = new GRBModel(env);

        // 3-D array
        GRBVar[,,] vars = new GRBVar[s, s, n + 1];
        // vars(row,col,value)


        for (int i = 0; i < s; i++)
        {
            for (int j = 0; j < s; j++)
            {
                for (int v = 0; v <= n; v++)
                {
                    string st = "G_" + i.ToString() + "_" + j.ToString()
                                     + "_" + v.ToString();
                    vars[i, j, v] = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, st);
                }
            }
        }
```

Each cell must take exactly one value

$$\sum_{k=1}^{9} X_{ij}^{k} = 1, \forall i, \forall j$$

```
// Add constraints

GRBLinExpr expr;

//// Each cell must take one value

for (int i = 0; i < s; i++)
{
    for (int j = 0; j < s; j++)
    {
        expr = 0.0;
        for (int v = 1; v <= n; v++)
            expr.AddTerm(1.0, vars[i, j, v]);
        string st = "V_" + i.ToString() + "_" + j.ToString();
        model.AddConstr(expr == 1.0, st);
    }
}
```

Each value is used at most once per row

$$\sum_{j=1} X_{ij}^{k} \leq 1, \forall i, \forall k$$

```
// Each value appears once per row

for (int i = 0; i < s; i++)
{
    for (int v = 1; v <= n; v++)
    {
        expr = 0.0;


        for (int j = 0; j < s; j++)
        {
            expr.AddTerm(1.0, vars[i, j, v]);
        }
        string st = "R_" + i.ToString() + "_" + v.ToString();

        model.AddConstr(expr <= 1.0, st);
    }
}
```

Each value is used at most once per column

$$\sum_{i=1} X_{ij}^{k} \leq 1, \forall j, \forall k$$

```
////Each value appears once per column

for (int j = 0; j < s; j++)
{
    for (int v = 1; v <= n; v++)
    {
        expr = 0.0;
        for (int i = 0; i < s; i++)
            expr.AddTerm(1.0, vars[i, j, v]);
        string st = "C_" + j.ToString() + "_" + v.ToString();
        model.AddConstr(expr <= 1.0, st);
    }
}
```

Sum of column values = prespecified value (V_c)

$$\sum_{i=1} \sum_{k=1} kX_{ij}^{k} = Vc, \forall j$$

```
}
for (int j = 0; j < 1; j++)
{
    GRBLinExpr expr3 = 0.0;

    for (int i = 0; i < s; i++)
    {
        for (int v = 1; v <= n; v++)
        {
            double w = (double)v;

            expr3.AddTerm(w, vars[i, j, v]);


        }

    }
    model.AddConstr(expr3 == array[0, 1], "g");


}
```

**this step should be repetead for each pre-specified column for each difficulty level.

E.g: This step is only for the column number 0. For 3*3 matrix it should be repeated once more(j=1):

```
for (int j = 1; j < 2; j++)
{
    GRBLinExpr expr4 = 0.0;

    for (int i = 0; i < s; i++)
    {
        for (int v = 1; v <= n; v++)
        {
            double w = (double)v;

            expr4.AddTerm(w, vars[i, j, v]);


        }


    }
    model.AddConstr(expr4 == array[0, 2], "f");


}
```

for 4*4 matrix twice more(j=1,j=2) and for 5*5 matrix whether it depends on the version, usually this step should be repeated three more times (j=1,j=2, j=3).

Sum of row values = prespecified value (V_r)

$$\sum_{j=1}\sum_{k=1} kX_{ij}^{k} = \text{Vr}, \forall \text{i}$$

```
////sum of prespecified vals

for (int i = 0; i < 1; i++)
{
    GRBLinExpr expr1 = 0.0;

    for (int j = 0; j < s; j++)
    {
        for (int v = 1; v <= n; v++)
        {
            double w = (double)v;

            expr1.AddTerm(w, vars[i, j, v]);

        }

    }
    model.AddConstr(expr1 == array[1, 0], "d");
}
```

**this step should be repetead for each pre-specified row for each difficulty level.

E.g: This step is only for the row number 0. For 3*3 matrix it should be repeated once more(i=1):

```
for (int i = 1; i < 2; i++)
{
    GRBLinExpr expr2 = 0.0;

    for (int j = 0; j < s; j++)
    {
        for (int v = 1; v <= n; v++)
        {
            double w = (double)v;

            expr2.AddTerm(w, vars[i, j, v]);

        }
    }
    model.AddConstr(expr2 == array[2, 0], "e");
```

for 4*4 matrix twice more(i=1,i=2) and for 5*5 matrix whether it depends on the version, usually this step should be repeated three more times (j=1,j=2, j=3).

After getting pre-specified values from constructed array, model is ready to be optimized:

```
model.Optimize();

// Write model to file
model.Write("kakuro.lp");
double[,,] x = model.Get(GRB.DoubleAttr.X, vars);
if (s == 2 || s == 3)
{
    for (int i = 0; i < s; i++)
    {
        for (int j = 0; j < s; j++)
        {
            for (int v = 1; v <= n; v++)
            {
                if (x[i, j, v] == 1)
                {
                    array[i + 1, j + 1] = v;
                    //Console.Write(array[i + 1, j + 1]);

                }



            }
        }
        //Console.WriteLine();
    }
}
```

Once optimization is done, freshly calculated values will be inserted into arrays. At this part arraj values (i,j) will be written as (i+1 and j+1) because matrix has to be return is original size. (at the beginning it is said that s=matrixSize-1)

Since situation is different with matrix 5*5 (which means s=4); it should be written for each version of difficult level.

```
for (int i = 1; i <= 1; i++)
{
    for (int j = 0; j <= 2; j++)
    {
        if (x[i, j, v] == 1)
        {
            array[i + 1, j + 1] = v;

        }

    }

}

for (int i = 2; i <= 2; i++)
{
    for (int j = 1; j <= 3; j++)
    {
        if (x[i, j, v] == 1)
        {
            array[i + 1, j + 1] = v;

        }

    }

}
for (int i = 3; i <= 3; i++)
{
    for (int j = 2; j <= 3; j++)
    {
        if (x[i, j, v] == 1)
        {
            array[i + 1, j + 1] = v;

        }
```

```
for (int j = 0; j <= 2; j++)
{
    if (x[0, j, v] == 1)
    {
        array[0 + 1, j + 1] = v;

    }
}

for (int j = 0; j <= 2; j++)
{
    if (x[1, j, v] == 1)
    {
        array[1 + 1, j + 1] = v;

    }

}

for (int j = 1; j <= 3; j++)
{
    if (x[2, j, v] == 1)
    {
        array[2 + 1, j + 1] = v;

    }

}

for (int j = 1; j <= 3; j++)
{
    if (x[3, j, v] == 1)
    {
        array[3 + 1, j + 1] = v;
```

```
for (int j = 0; j <= 1; j++)
{
    if (x[0, j, v] == 1)
    {
        array[0 + 1, j + 1] = v;
    }
}
for (int j = 0; j <= 3; j++)
{
    if (x[1, j, v] == 1)
    {
        array[1 + 1, j + 1] = v;

    }

}
for (int j = 0; j <= 3; j++)
{
    if (x[2, j, v] == 1)
    {
        array[2 + 1, j + 1] = v;

    }

}
for (int j = 2; j <= 3; j++)
{
    if (x[3, j, v] == 1)
    {
        array[3 + 1, j + 1] = v;

    }
}
```

For version 1,2,3. (All is in the for loop v=1 to 9)

After inserting freshly calculated values into array, program is ready to write solution into a file.

```
    StreamWriter wr = new StreamWriter(P + "kakuroSolution.txt");


    for (int i = 0; i < s + 1; i++)
    {
        for (int j = 0; j < s + 1; j++)
        {
            if (j < s)
                wr.Write(array[i, j] + ",");
            else
                wr.Write(array[i, j]);

        }
        wr.WriteLine();
    }


    wr.Close();

    // Dispose of model and env
    model.Dispose();
    env.Dispose();

}
catch (GRBException e1)
{
    Console.WriteLine("Error code: " + e1.ErrorCode + ". " + e1.Message);
}
```

Unsolved

```
-1,9,29,16,-1
24,0,0,0,-1
9,0,0,0,17
-1,21,0,0,0
-1,17,0,0,0
```

Solved

```
-1,9,29,16,-1
24,8,9,7,-1
9,1,5,3,17
-1,21,8,4,9
-1,17,7,2,8
```

Figure 3.11. Unsolved/Solved Kakuro Puzzle

Table 6. Comparison of Solvers and The Reason of Choosing Gurobi

Free and Open Source Solvers	**Commercial Solvers**
GLPK	Cplex
SoPlex	Xpress
CLP	Gurobi
SCIP	
LP_Solve	

Short Explanations of the Solvers

GLPK: Created to solve large-scale linear programming (LP), mixed integer programming (MIP). It is written in ANSI C and also has a callable library.

SoPlex: Linear programming solver implemented in C ++. It can be used standalone or embedded into program using C++ library

CLP: An open source programming solver written in C++. At first it is created to be callable library but standalone version is also available.

SCIP: Available for solving integer and constraint programs. Has open LP Solver support, so that it is possible to call a range of open source and commercial LP solvers.

LP_Solve: Also an open source solver that can be used to solve linear (mixed integer) programs and is written in ANSI C.

Cplex: Large-scale (mixed integer) linear problems. Actively developed by IBM.

Xpress: Desinged to solve LP. Provides callable library and standalone interface.

Gurobi: Modern solver for LP as well as NLP.

The table below is a comparison chart from Matthias Templ's study in 2012.

While making this comparison, the latest versions of the programs were used and 87 problems were tried to be solved. In the second column of the table, there is the average solution time of the problems, in the third column, how many of the 87 problems are solved is written, and in the last column, the percentage of the problems solved is shown.

Table 7. The Percentage of the Problems

Solver	**Running Time**	**Instances solved**	**Solved(%)**
GLPK	22.11	3	3.45
SCIP-S (using SoPlex)	5.33	57	65.52
SCIP-C (using CLP)	3.76	63	72.41
LP_Solve	19.40	5	5.75
CPlex	1.45	73	83.91
Xpress	1.29	74	85.06
Gurobi	1.00	77	88.51

The chart below is a visual demonstration of program comparisons.

As it is written in the table, the orange areas represent the number of problems solved and it can be observed that the program with the most solutions is Gurobi. And again, as seen in the table, the blue parts show the average solution times of the questions. Gurobi solved 87 problems in an average of 1 second and is the fastest optimization tool in comparison.

Gurobi has been used in this project since Gurobi is the optimization solver, which solves the most problems in the shortest time compared to all other optimization solvers.

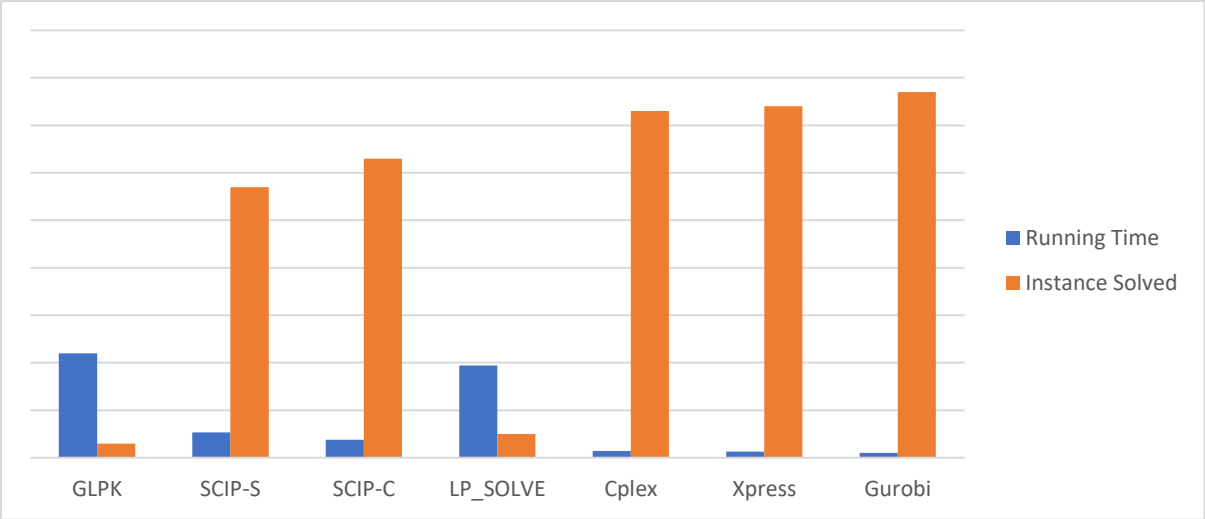


Figure 3.12. Comparision of Solvers

3.2 Software Engineering

3.2.1 Interface Requirements

3.2.1.1 User Interfaces

This Project have a dashboard as homepage of form. In that dashboard, user can choose the game, Sudoku or Kakuro. Users should click on the image which game she/he wants to play. Then, related game will be opened. Figure 3.8. represents dashboard of Project.

Figure 3.13. Project Dashboard

In Sudoku page, users will see game and can see the level of the game, and after choosing level, the puzzle will be shown in the screen according to related level. Also, there are six buttons for users(Start, Submit, Solution, Save Score, Hint and Solve). Users will see a timer, timer stop button and a listbox in order to write/save their game score.

Additionlly, there are a menu bar in the page above which names are ‘Go To’, ‘Help’, and ‘Developers’. Using Go To, users can go another game or dashboard back. Using Help, user can get help about solving external games, and finally, in Developers tab, game developers are shown.

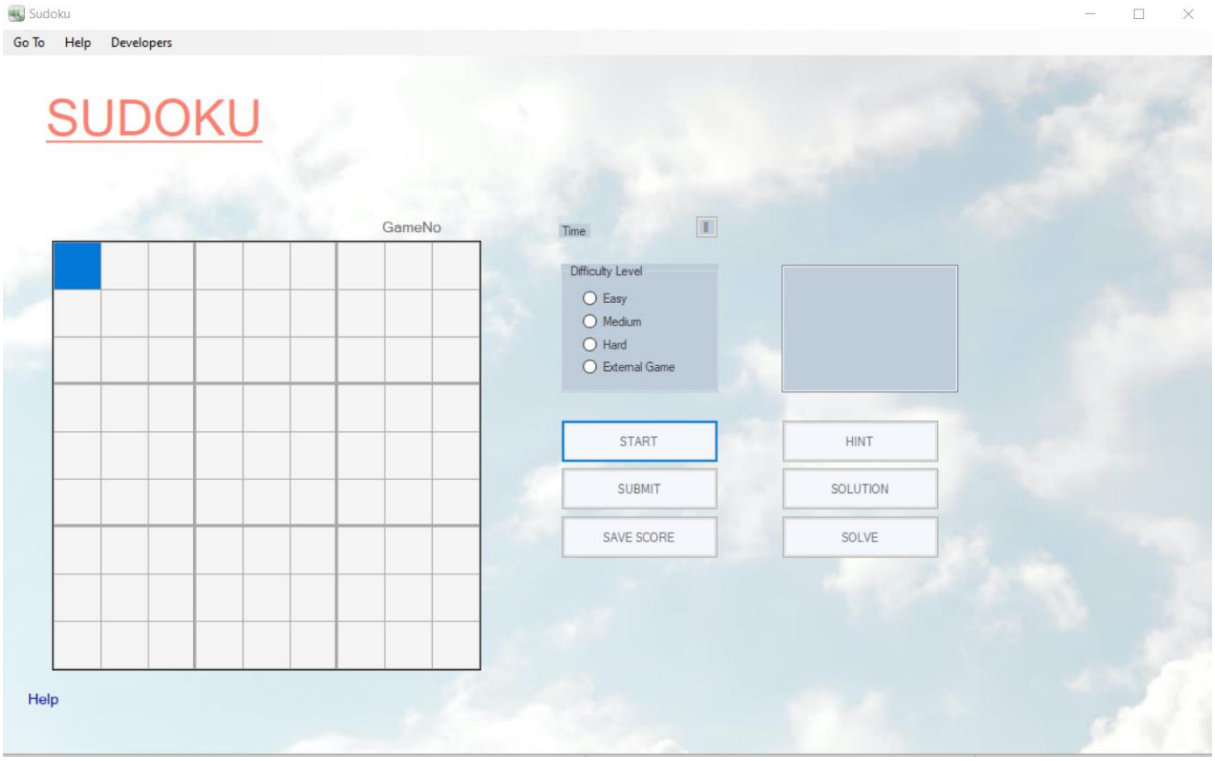


Figure 3.14. Sudoku Homepage

In Kakuro page, users will see default game puzzle and level of the game, after user choose the level, puzzle will be automatically changed. Also, there are 5 buttons for users(New Game, Submit, Solution, Save Score,Solve) and user will see a timer and a listbox to save /write their game score. In addition for that, there are stop/resume button for the game.

There are a menu bar in the page above which names are ‘Go To’, ‘Help’, and ‘Developers’. Using Go To, users can go another game or dashboard back. Using Help, user can get help about solving external games, and finally, in Developers tab, game developers are shown.

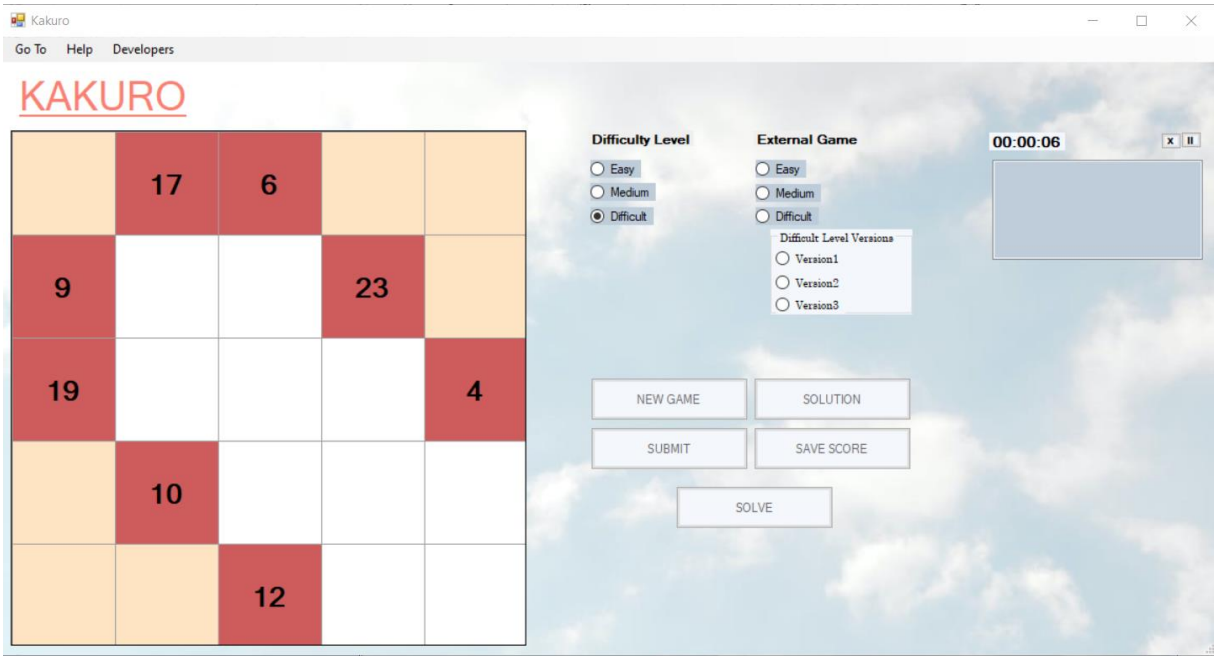


Figure 3.15. Kakuro Homepage

3.2.1.2 Software Interfaces

This project was done using C# & .Net framework with Microsoft Visual Studio 2019 IDE. Also, there are puzzle databases, solutions databases, and external games&solutions databases as txt file. Most of these databases are kept in C: of a computer, Kakuro puzzles&solutions are directly kept in a Project file as 'PuzzleType'. The libraries of this Project are listed in below:

- `using System.Collections.Generic: Using for data structures, data collections and arrays.`
- `using System.IO: Using for classes that are for input and outputs applications.`
- `using System.Data; Using for databases processes.`
- `using System.Linq: Using for multiple databases processes.`
- `using System.Drawing: Using for graphics/images.`
- `using System.Text: Using for string&char type variables.`
- `using System.Threading: Using for creating/checking a thread.`
- `using System.Windows.Forms: Using for GUI design.`
- `using Gurobi: Using for processes of Gurobi.`

3.2.2 Functional Requirements

3.2.2.1 Behaviors of the Software Application

Table 8. Behaviors of Project

Actor Name	**Name of Behavior**	**Description of Behavior**
User	*ChooseLevelofGame()*	*User can choose the level of the game as easy, medium or difficult, then starts game.*
User	*StopResumeGame()*	*User can stop the game and resume.*
User	*ResetGame()*	*User can reset the game and starts the new game or diffirent level game.*
User	*GetSolution()*	*User can see the solution of the game.*
User	*GetHints()*	*User can get hints for that game but max 3 times for one game.*
User	*SaveScore()*	*User can save score in the listbox.*
User	*SubmitGame()*	*User submits the game and can see the corrects/wrongs.*
User	*GetAnotherGameSolution()*	*User can solve another game using external game button.*

3.2.2.2 Attributes of the Software Application

Table 9. Attributes of Project

Object Name	**Name of Attribute**	**Description of Attribute**
GamePuzzle	*cColWidth*	*Equals width of a cell for Sudoku.*
	cRowHeight	*Equals height of a cell for Sudoku.*
	cMaxCell	*Equals total cell of Sudoku.*
	cSidelenght	*Equals whole Sudoku grid length.*
	sayi	*Equals a random number.*
	oncekisayi	*Equals random number for checking different games.*
	countTrue	*Equals correct number of whole cells of Sudoku.*
	countFalse	*Equals wrong number of whole cells of Sudoku.*
	row	*Equals number of rows of Kakuro.*
	col	*Equals number of columns of Kakuro.*
	index	*Equals index of a cell of Kakuro.*
	rowNumber	*Equals current row number of Kakuro.*
	P	*Equals database of Sudoku for getting numbers.*
	P2	*Equals database of Sudoku solutions for getting all numbers.*
	groupOfPuzzle	*Equals database of Kakuro for getting numbers.*
	groupOfPuzzleSolution	*Equals database of Kakuro solutions for getting numbers.*
	currentPuzzleType	*Equals current puzzle of Kakuro game.*
	selectedIndexNumber	*Equals selected index of selected cell of Kakuro.*
Time	*saat*	*Equals spend hours for game.*
	dakika	*Equals spend minutes for game.*
	saniye	*Equals spend seconds for game.*

3.2.2.3 Design and Implementation

Database Management System

In this Project, txt files are used as a Database. For Sudoku game puzzles & solutions and Kakuro external game puzzles & solutions are kept in the C: of computer. When a user start the game, numbers directly come from that txt files according to levels. Additionally, Kakuro puzzles of all levels(easy, medium, difficult) are kept in the Project as a file which name is “PuzzleType”. Because in the code, dictionary structures are used for the game numbers.

Database Physical Model

The whole database consists of numbers and its structure is 'int', there is no string or char variables in database. For Kakuro, the numbers of a game puzzle&solutions are splitted with comma(,) and whole puzzles are separated from each other using ‘#’ in all txt files. However, for Sudoku, for each game puzzle, there are separated txt files for each game and their solutions.

In addition, for both games, '0' in the databases shows empty cells where the user must enter the number.

Completed Database Screenshots

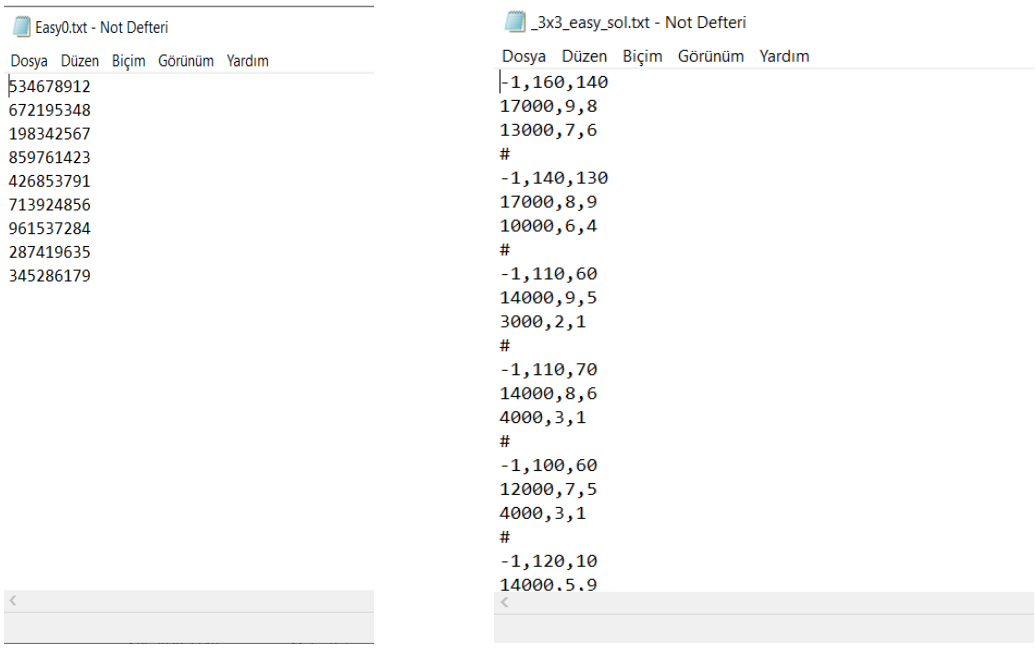


Figure 3.16. Example Database of Sudoku and Kakuro Solutions

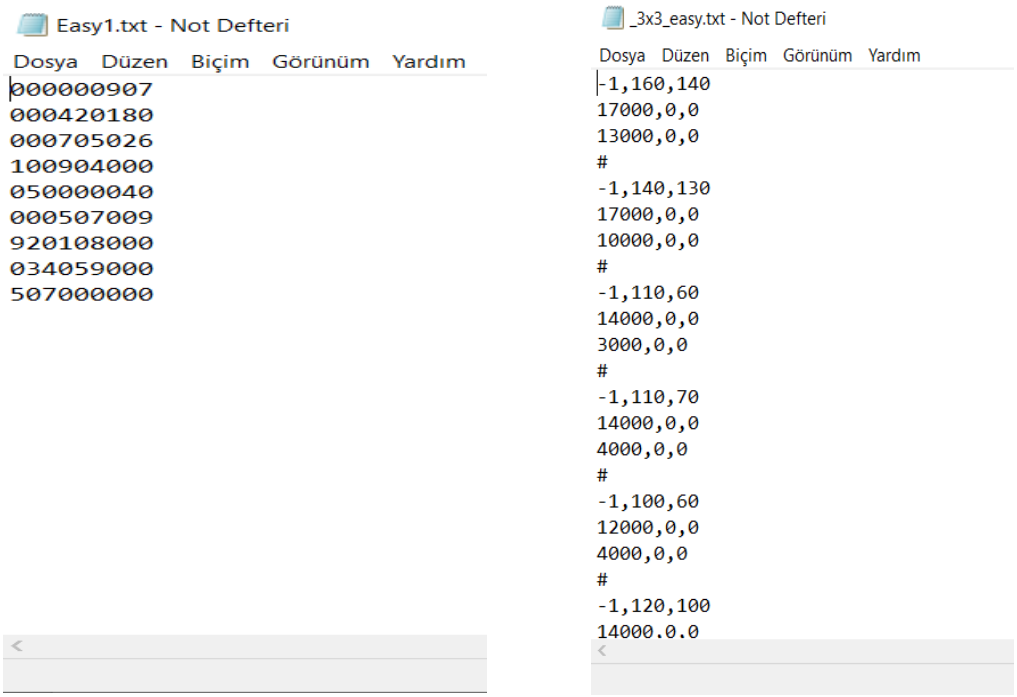


Figure 3.17. Example Database of Sudoku and Kakuro

3.2.3 Nonfunctional Requirements

3.2.3.1 Software Quality Attributes

Availability: The system/form shall run smoothly at any time of the day and the user can easily access it.

Testability: This system/form shall be tested without any problem in any condition with internet connection.

Usability: The system/form shall be simple to use, understand and learn.

Portability: The system/form shall be functional in mobile devices, tablet modes or PC.

Reliability: The system/form shall be able to execute without error for a certain period of time or under conditions where the internet connection is weak.

Maintainability: New features shall be easily added to this system/form and the system should be updated.

Flexibility: The system/form shall be able to run under conditions such as disconnection of the internet connection while the system is running.

3.2.4 Use-case Modeling

3.2.4.1 User Requirements

First of all, the user enters the web page, then user will see a Sudoku game which is unresolved. Before user starts the game, she/he can choose level of game as easy, medium or difficult. If user wants to get help about game's solution, she/he can get solution of that game or another game which is entered by user. Also, user can stop or reset the game. After game is over, user is able to start new game.

3.2.4.2 Use Case Scenarios

3.2.4.2.1 Use Case Scenario for the Entering Web Page

Use-case Name	*Enter web page*
Use-case Description	*User enter the web page.*
Actors	*Users*
Pre-Condition	*User should go to the site.*
Post-Condition	*User see game' interface/page.*
Normal Flow	*Step1: User enters the web page.*
Alternate Flow	*Alt-step 1: Page does not load.* *-Reload the page.*
Business Rules	*No business rule*

3.2.4.2.1 Use Case Scenario for the Starting New Game

Use-case Name	*Start New Game*
Use-case Description	*Game is coming from database and user starts the new game.*
Actors	*Users*
Pre-Condition	*User should enter the page and choose the level of game.*
Post-Condition	*User plays game.*
Normal Flow	*Step1: User enters the web page.* *Step2: Database sends a game as easy level automatically.* *Step 3: User see the game.* *Step 4: User starts to play.*
Alternate Flow	*Alt-step 2: User can change level of the game.*
Business Rules	*If user does not choose any level, the game starts as easy level.*

3.2.4.2.2 Use Case Scenario for the Starting New Game

Use-case Name	*Choose Level of Game*
Use-case Description	*Game is coming from database and user choose the game's level.*
Actors	*Users*
Pre-Condition	*User should enter the page and click the level of game box.*
Post-Condition	*User plays game with related level.*
Normal Flow	*Step1: User enters the web page.* *Step2: Database sends a game as easy level automatically.* *Step 3: User sees the game.* *Step 4: User changes the level.*
Alternate Flow	*Alt-step 2: User can change level of the game.*
Business Rules	*If user does not choose any level, the game starts as easy level.*

3.2.4.2.3 Use Case Scenario for the Stopping Game

Use-case Name	*Stop Game*
Use-case Description	*User stop the game while he/she is away from game.*
Actors	*Users*
Pre-Condition	*User should start the game.*
Post-Condition	*Game stops.*
Normal Flow	*Step1: User enters the web page.* *Step2: Game is coming from database automatically as a easy level.* *Step 3: User should choose the level of game if wants to change.* *Step 4: User clicks the stop button.* *Step 5: The game is stopped.*
Alternate Flow	*No alternative flow*
Business Rules	*After the game is reset after a long period of play.*

3.2.4.2.4 Use Case Scenario for the Getting Hint

Use-case Name	*Get Hint*
Use-case Description	*User gets hint about the game solution.*
Actors	*Users*
Pre-Condition	*User should start the game.*
Post-Condition	*User gets one hint.*
Normal Flow	*Step1: User enters the web page.* *Step2: Game is coming from database automatically as a easy level.* *Step 3: User should choose the level of game if wants to change.* *Step 4: User clicks the hint button.* *Step 5: A hint comes from database to user.*
Alternate Flow	*Alt-step 5: If clicked more than 5 times, the user cannot get hint anymore.*
Business Rules	*The hint button can be clicked up to 5 times.*

3.2.4.2.5 Use Case Scenario for the Resetting Game

Use-case Name	*Reset Game*
Use-case Description	*User reset the game and can start the new game.*
Actors	*Users*
Pre-Condition	*User should start the game.*
Post-Condition	*The new game comes from database.*
Normal Flow	*Step1: User enters the web page.* *Step2: Game is coming from database automatically as a easy level.* *Step 3: User should choose the level of game if wants to change.* *Step 4: User clicks the reset button.* *Step 5: Existing game is over.* *Step 6: The new game comes from database.*
Alternate Flow	*Alt-step 6: User can change the level of the game.*
Business Rules	*If the user does not play for 15 minutes, the game resets itself.*

3.2.4.2.6 Use Case Scenario for the Solution of Existing Game

Use-case Name	*See Solution of Existing Game.*
Use-case Description	*User see the whole solution of the game.*
Actors	*Users*
Pre-Condition	*User should start the game.*
Post-Condition	*Solution of the game is coming from database.*
Normal Flow	*Step1: User enters the web page.* *Step2: Game is coming from database automatically as a easy level.* *Step 3: User should choose the level of game if wants to change.* *Step 4: User clicks the See Solution button.* *Step 5: Solution of existing game comes from database to user.*
Alternate Flow	*No alternative flow*
Business Rules	*No business rule*

3.2.4.2.7 Use Case Scenario for the Solution of Another Game

Use-case Name	*See Another Game Solution*
Use-case Description	*User monitorizes different solution of game which is in another source.*
Actors	*Users*
Pre-Condition	*User should start the game.*
Post-Condition	*Solution of the game is coming from database.*
Normal Flow	*Step1: User enters the web page.* *Step2: Game is coming from database automatically as a easy level.* *Step 3: User should choose the level of game if wants to change.* *Step 4: User clicks the See another game solution button.* *Step 5: The user enters the numbers of the game he/she wants the solution.* *Step 6: Game solution is coming from database.*
Alternate Flow	*Alt-step 3: The user can click the 'See another game solution' button even if she/he does not start any new game.*
Business Rules	*No business rule*

3.2.4.2.8 Use Case Scenario for the Solution of Another Game

Use-case Name	*Send Games and Solutions*
Use-case Description	*Database send game and solution to the user page.*
Actors	*Database*
Pre-Condition	*User should start the game.*
Post-Condition	*Database sends game or solution.*
Normal Flow	*Step1: User enters the web page.* *Step2: Database sends a game related with level automatically.* *Step 3: User see the game.*
Alternate Flow	*Alt-step 2: Database send the solution according to user's choice.*
Business Rules	*No business rule*

3.2.4.3 Use-Case Diagram

Use-case Diagram of the project is shown below.

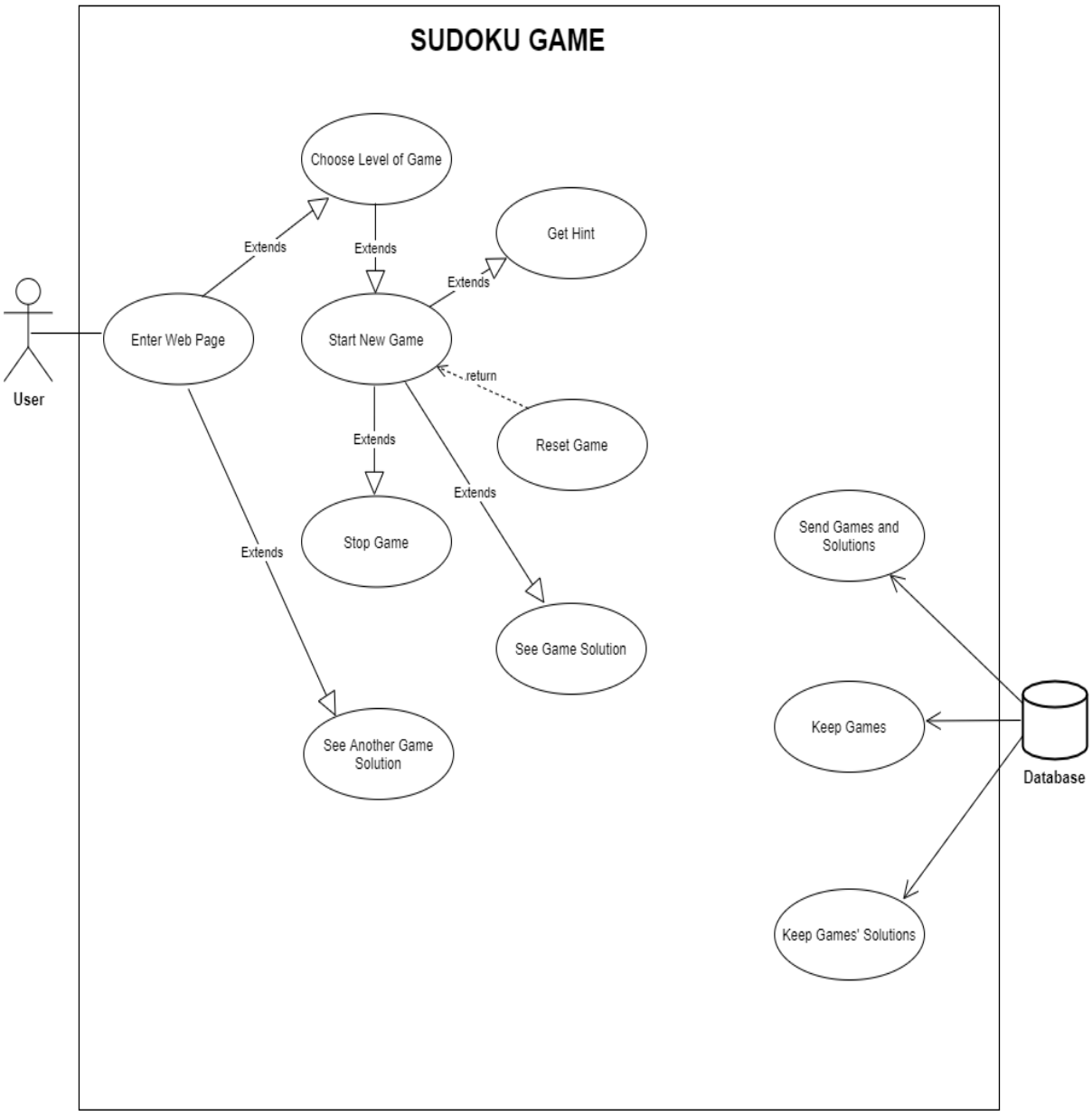


Figure 3.18. Use-Case Diagram of Game Project

3.2.5 Data Modeling

3.2.5.1 Activity Diagram

Activity Diagram of the project is shown below.

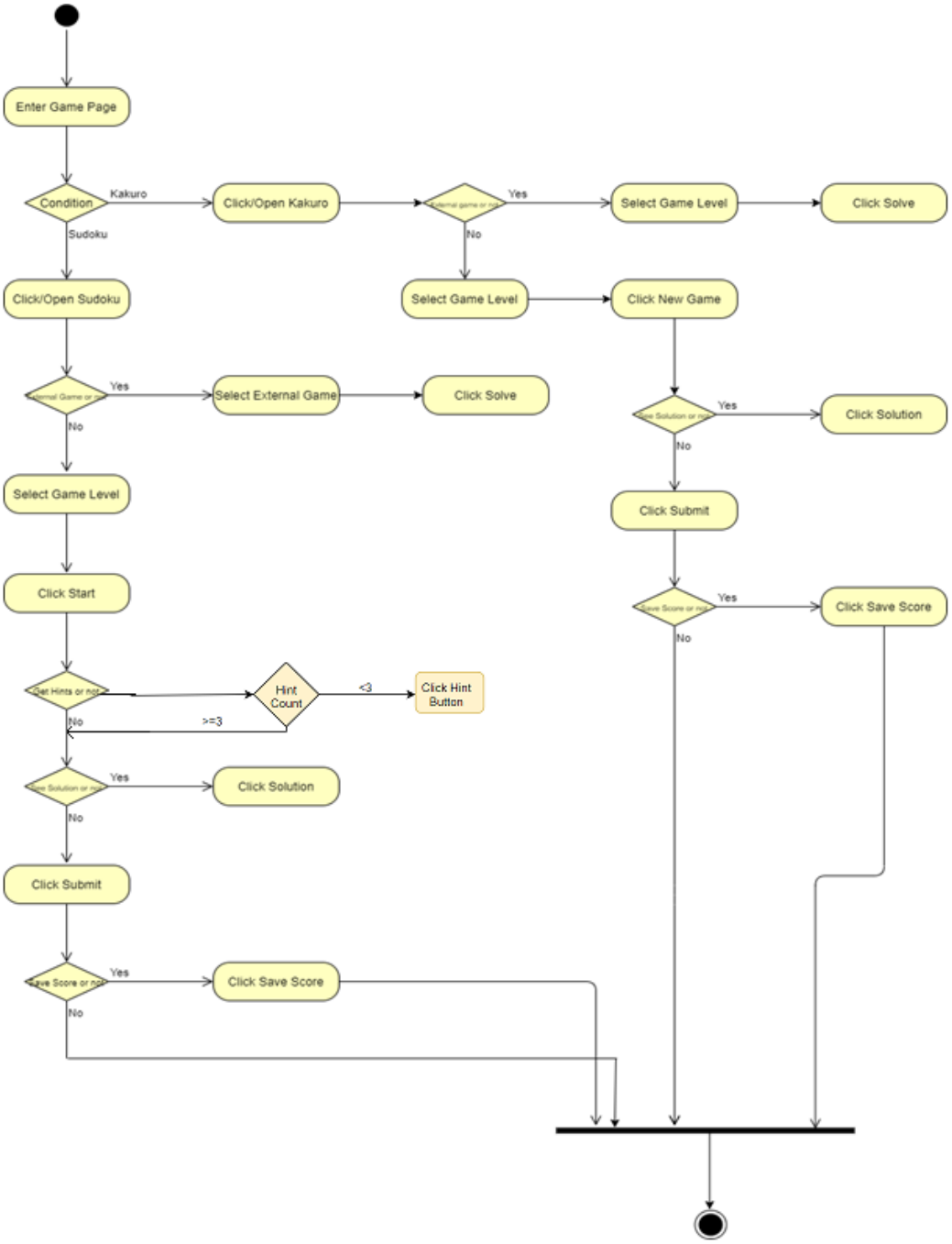


Figure 3.19. Activity Diagram of Game Project

3.2.5.2 Sequence Diagram

Sequence Diagram of the project is shown below.

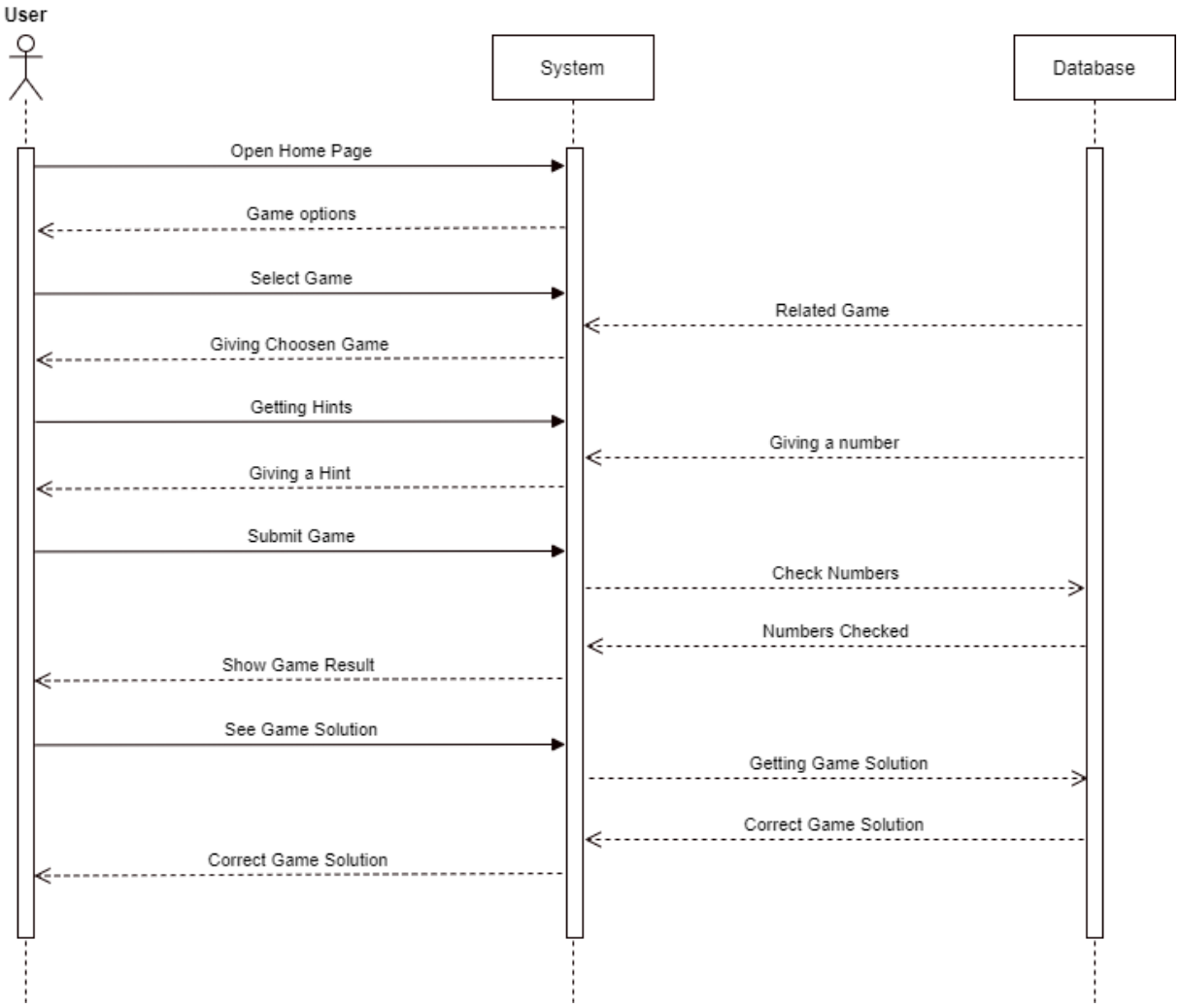


Figure 3.20. Sequence Diagram of Game Project

3.2.5.3 Data Flow Diagram

Data Flow Diagram of the project is shown below.

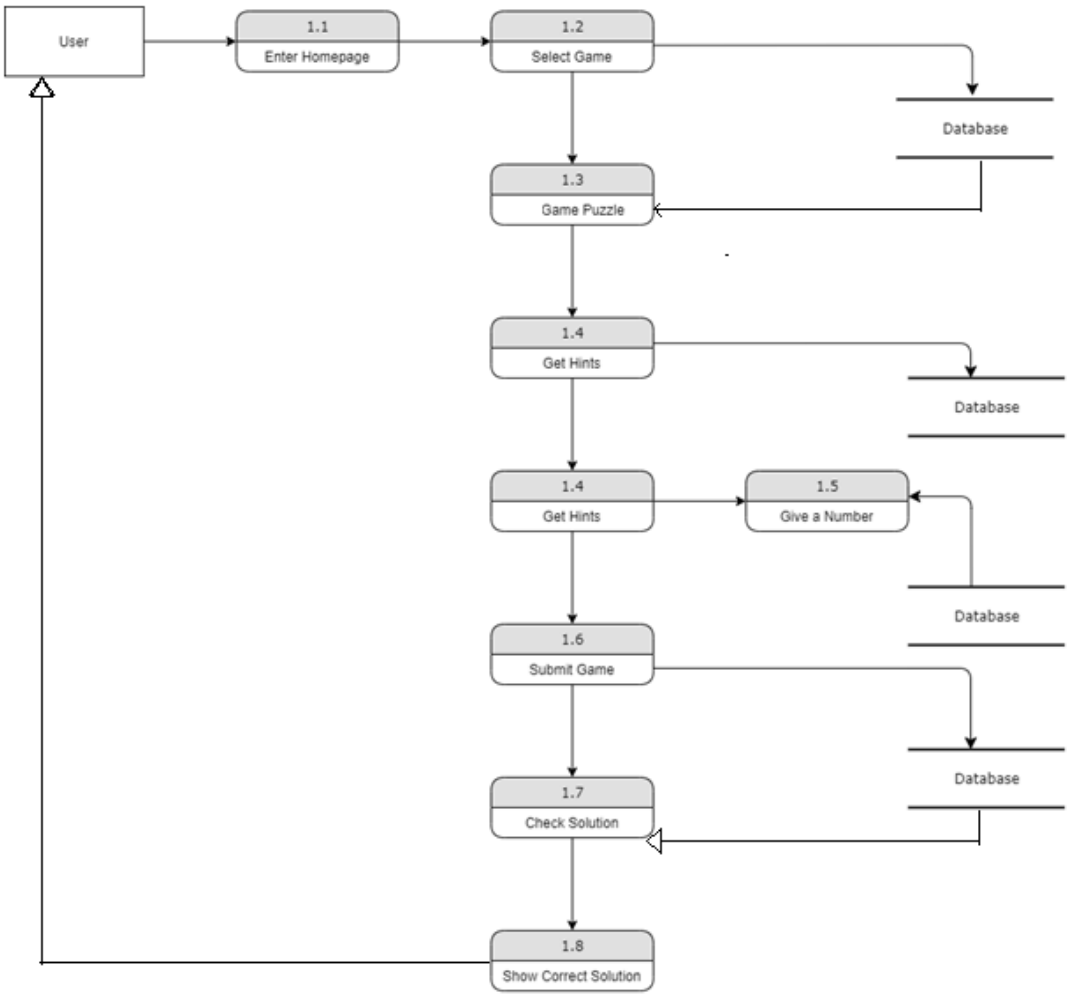


Figure 3.21. Data Flow Diagram of Game Project

3.2.5.4 Design/Block Diagram

Design/Block Diagram of the project is shown below.

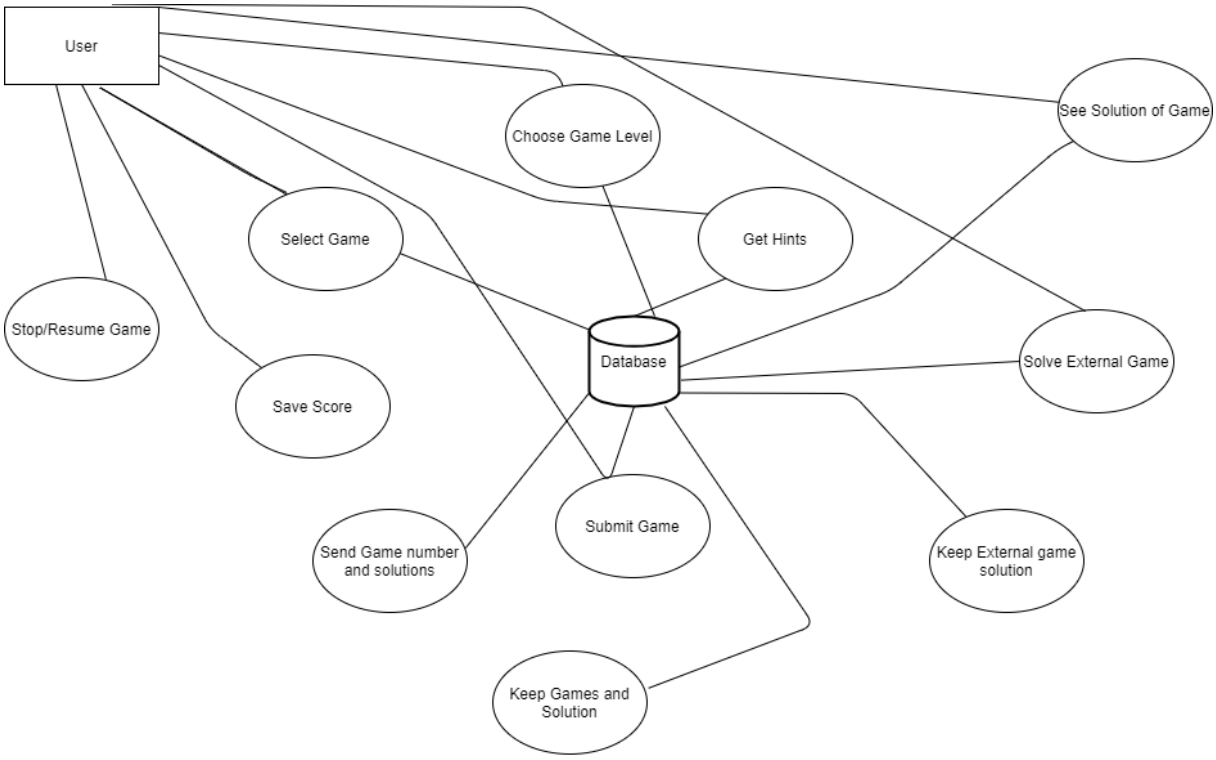


Figure 3.22. Design/Block Diagram of Game Project

3.2.5.5 Object Diagram

Object Diagram of the project is shown below.

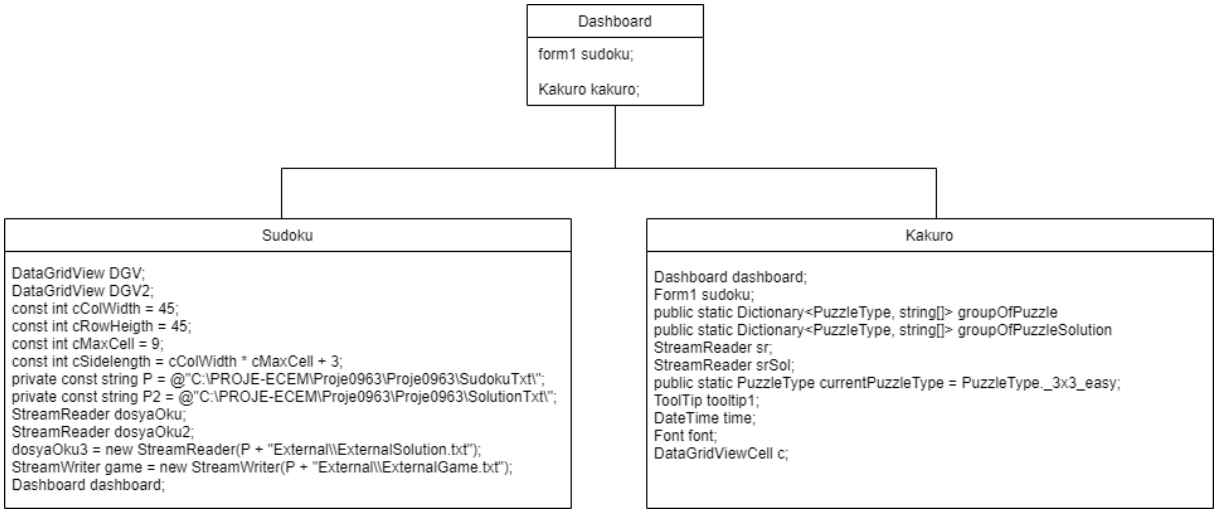


Figure 3.23. Object Diagram of Game Project

3.2.5.6 Class Diagram

Class Diagram of the project is shown below.

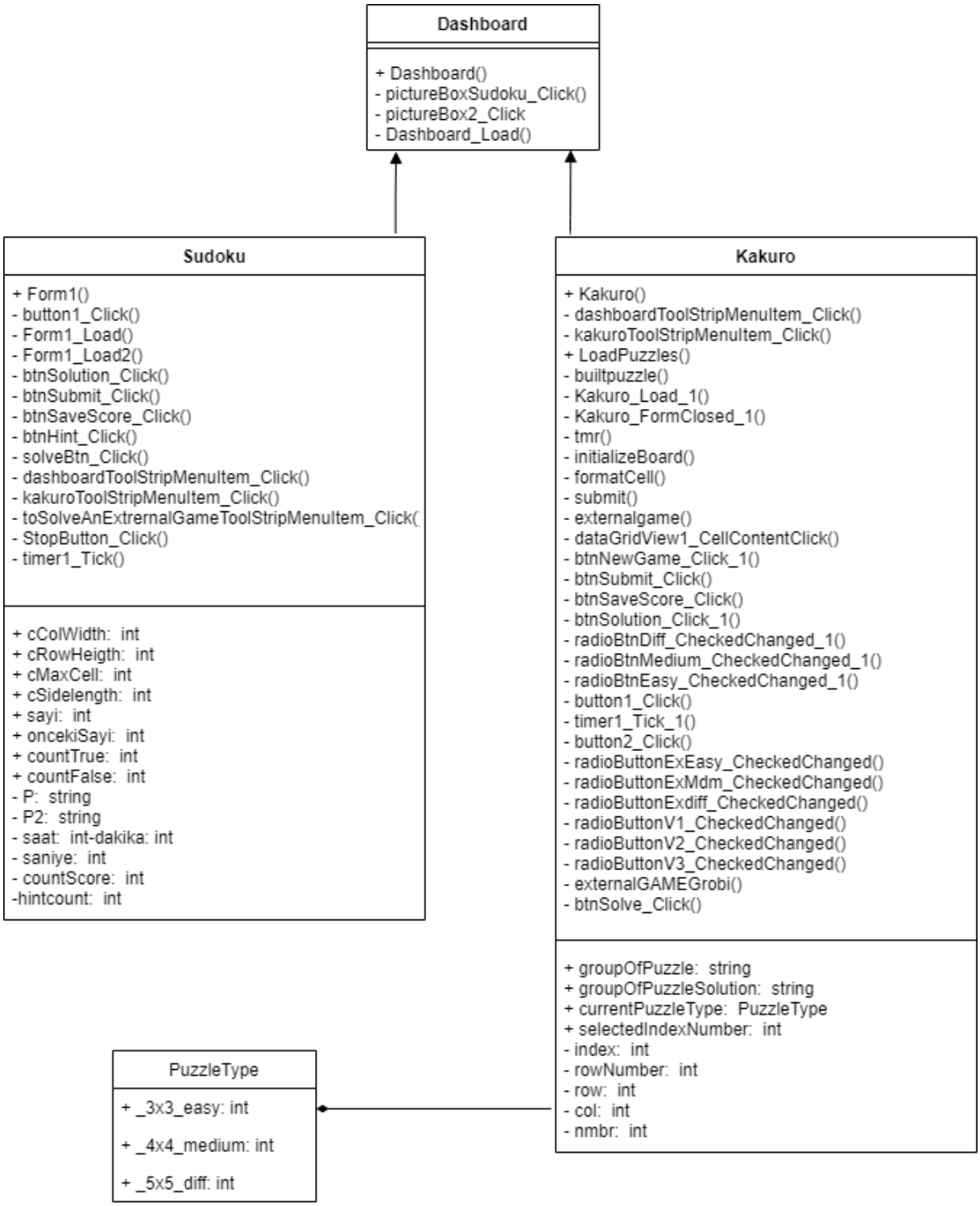


Figure 3.24. Class Diagram of Game Project

3.2.6 Test Results

To make sure that the project runs succesfully, functional and non-functional testing methods are operated. While constructing test class, at first unit testing project is added and then it is attached to the main project so that the properties to test can be seen. After it is decided which özellikler to test, methods are created accordingly. As functional testing, unit testing is implemented into project to see whether program is giving the exact desired puzzle when a RadioButton is chosen.

Also another unit testing method implemented to make sure that Sudoku puzzle grid is in the original form which consist of 9*9 rows and columns. For the non-functional testing part usability testing is done by both developers and users. As the conclusion of this usability test, it is decided that both games are appropriate to their original game format and external game section is working as intended.

4. RESULTS

4.1 Project Budget

In this project, there is no cost.

Component	Cost (TL)
TOTAL	**0**

4.2 Project Communication Matrix

Project developers and advisers' information is given below.

	Name and Surname	Email
STUDENTS	Şule Akça	sule.akca@bahcesehir.edu.tr
	Ecem Altınel	ecem.altinel@bahcesehir.edu.tr
	Ertuğrul Duman	ertugrul.duman@bahcesehir.edu.tr
ADVISERS	Doç. Dr. Tankut Atan	sabritankut.atan@eng.bau.edu.tr
	Prof. Dr. Mehmet Alper Tunga	alper.tunga@eng.bau.edu.tr

5. CONCLUSION

In this project, it is aimed to create a website which offers both the game and its solutions or the solution of any Sudoku and Kakuro game. The important advantage of the project is that the user can easily find the solution of a game that s/he sees in any newspaper or magazine, but cannot find a solution. To do that, it is sufficient to integrate the numbers on the form. In addition, the fact that the games are separated according to the difficulty level is important for the user to play the game at the desired level.

As stated above, since the main purpose of the project was build an application which allows users to find the solutions of external games besides playing Sudoku and Kakuro, an optimization solver had to be used as an analyzer.

It is already mentioned in the comparison phase that Gurobi has been used in this project since it can solve the problems in a shorter time. Gurobi was added externally as a parser to the Windows Form Application, and the Gurobi library was used to translate mathematical modeling into code for Sudoku and Kakuro game. Even though it was not complicated to implement Sudoku into codes, converting Kakuro modelling into codes was challenging. Since in this project only 3*3,4*4,5*5 matris puzzles are used, to improve the game the other shapes can be implemented, as well.

Although this project was originally intended to be created as a website, since Javascript is not supported by Gurobi, which is the language used to create the website, it was created using C#, one of the languages supported by Gurobi. Being created as Windows Form Application provides an advantage to play these game without an internet connection.

Since this application is designed for users to play the classic Sudoku and Kakuro game, it includes time display and leaderboard. Users can get hints when they get stuck on solving puzzle, and if they want to solve an external game, they will also have a help section explaining it.

While finding and using problems for the games Sudoku and Kakuro, two different method was followed. One of them is to keep Kakuro problems, which are approximately 50, in the same txt file and pass them to the game table using the dictionary method. The other is to store 150 problems as easy, medium and hard in separate folders and to pass those problems to game table by reading from the txt files. However, the number of games that the user plays can be increased, so more games can be provided to the user.

To sum up, for this project an easy to play and user friendly game platform is constructed, so that users will be able to reach Sudoku and Kakuro game any time, anywhere without internet connection and will be able to solve external games just with one click, as well.

ACKNOWLEDGEMENT

We wish to thank our advisers Prof. Dr. Mehmet Alper TUNGA and Assoc. Prof. Dr Tankut ATAN for their all helps and attentions.

This work was partly/wholly funded by Bahçeşehir University.

REFERENCES

1. https://sudoku.com/
2. https://www.gurobi.com/documentation/
3. https://www.kakuros.com/
4. https://www.gurobi.com/documentation/9.0/examples/sudoku_cs_cs.html
5. http://www.baskent.edu.tr/~bkececi/userfiles/file/SUDOKU/SudokuILPModel.pdf
6. https://support.gurobi.com/hc/en-us
7. https://support.gurobi.com/hc/en-us/articles/360025135272-How-to-handle-a-BadImageFormatException-
8. https://github.com/dimitri/sudoku/blob/master/sudoku.txt
9. https://qqwing.com/
10. https://www.daniweb.com/programming/software-development/code/453445/making-a-datagridview-look-like-a-sudoku-board
11. https://docs.microsoft.com/tr-tr/dotnet/core/tutorials/library-with-visual-studio?tabs=csharp
12. https://www.tutorialspoint.com/software_engineering/software_analysis_design_tools.htm
13. https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/