Name: Qi Li **Email**: Qi5@kth.se

Name: Hussam Hassanein **Email**: Hussamh@kth.se

The Spread Pattern of Disease Infection in Fixed Population

Table of Content

Problem description

When a disease infection alert has been triggered in WHO(World Health Organization), the most valuable information that researchers need to figure out is the the spread pattern of the disease with the damage that might occurs in future days. The goal of this project is to create a simulation application that simulate the disease spreading process and pattern with reasonable mathematical model and system. This can provide researchers a preview about what might be happen in the future days.
The simulation is base on the scientific fact that currently already known by the world. Base on these facts,one need to define the best way to simulate the the infection. The data that simulator produced should be accurate and precise with proper method of verification and validation.

Model description

Contamination is modeled as follows: The population is modeled as an NxN matrix in which each element represents an individual in the population. The disease to be simulated is spread by direct contact between individuals. An individual is ill for a random number of days during which time there is a probability that the individual infects each of its direct neighbors (normally 8) in the population. An individual that gets infected one day you cannot infect other individuals the same day. Every day an individual is ill, there is also a probability that the individual dies. An individual who dies is not contagious. A sick person cannot be infected. An individual who has been ill but recovered, i.e. who is healthy, is immune and cannot be infected again. A run of the simulation should last until no ill individuals remain in the population.

Input
The user should be able to set values for the following input parameters:
• Size of population (N)
• Probability (per day) that a sick individual infects a healthy neighbor who is not immune (S)
• Length of time an individual is ill, rectangular distributed in an interval [minDays, maxDays]
• The probability that an individual dies a day the individual is ill (L)
• The number of individuals that are ill initially and where they are located

Output from the simulation
Output from the simulation should be (the user should be able to turn on/off output):
• The number of individuals that becomes infected per day
• The number individuals that died per day
• The number of individuals that have recovered per day
• The number of ill individuals per day
• The accumulated number of infected every day
• The accumulated number of deaths per day

The user should be able to specify values for all inputs on the command line when the execution is initiated (on the Linux servers), but all parameters should also have a default value. Similarly, one should be able to select which output is to be displayed/printed. Validation of the simulation is simplified if the program can output a graphical representation of the population with information on which individuals are healthy, ill and deceased.[1]

The pros and cons of the model

The simulator provide a representation of the damage and spread pattern in a ongoing disease infection within a fixed population. The simulator allow user to control the desired input and the output value. During the simulation process, the simulator will go through each person and terminate the simulation process when all the people inside the fixed population has been scanned through by the simulator.

Pros

The model that simulator used is the basic fact of all the disease. This makes the model can be really useful on the disease that has similar biological structure before in the database. The model is also useful in extensive study of already known disease for more efficient cure. The simulator that use this model can provide all the known fact to the user without start from beginning.
Instead only focus on the result of the simulation, the simulator focus more about the process of the infection spreading. To achieve a ‘process focus’ simulator, the simulator provide a active GUI(Graphic User Interface) that output more informative and visually result to the user. By using GUI, the simulator can be used by wider user group. The user group can be professionals who focus more and the process of the spreading or student who more interested on the result of the ‘aftermath’ . The simulator also allow user to control the input and output value base on the need of the user. This provide a open platform for different user and increase the accessibility of the simulator.

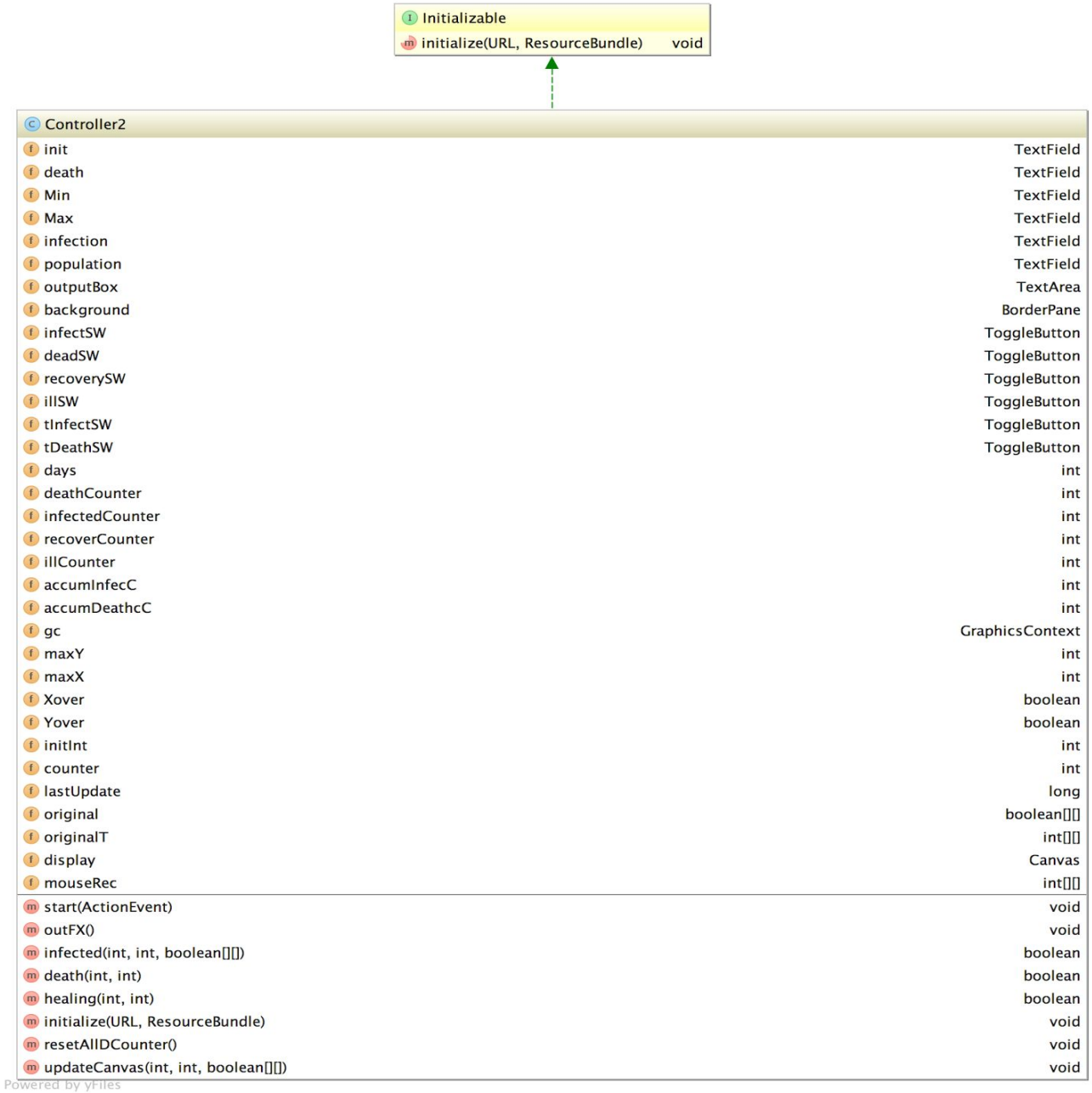
Cons

The model only include the general scenarios of all the disease. In reality all the disease has different property based on their biological structural. If the disease is mutant into a unknown area that some unknown pattern and input can occur during the infection, the simulator will be useless by than.
The model that current simulator use should be the basic guideline of the program. User should be base on this guideline create their own extensive model. This will make the true open platform for solving the problem and product more precise and accurate result. The trade of here is restrict the user group into professionals.

Solution description

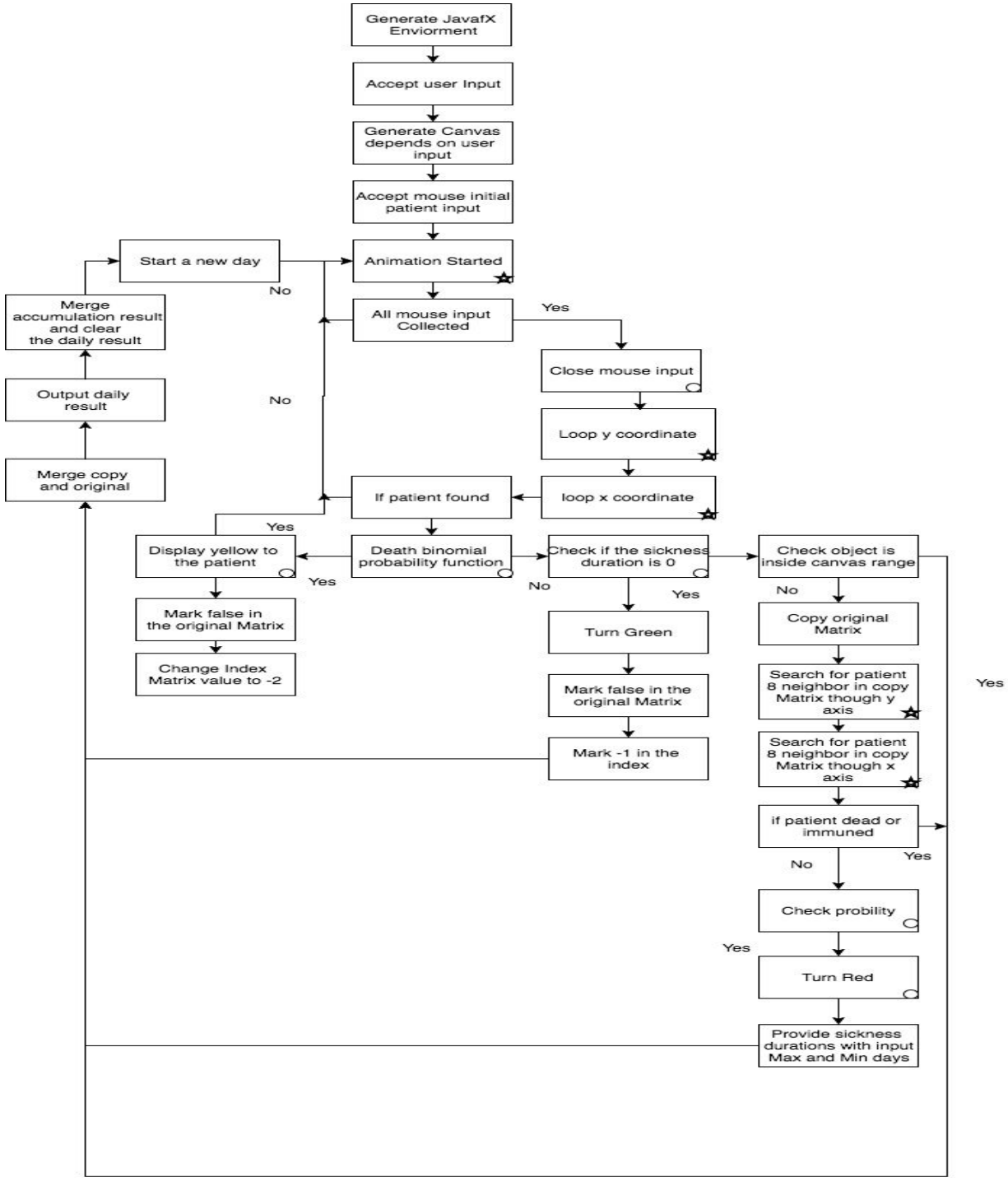
Diagrams

UML diagram



Flow chart

Both Diagram can be found under AFS/HOME/q/i/qi5/untitled2/Diagram and AFS/HOME/h/u/untitled2/Diagram folder for higher resolution file



Methodology

The simulator will start by require obligated input value, and then detect if the initial input is valid. The input check has been done by an if statement with maximum of total 70*74 pixel of the whole canvas. Each patient match to 1 pixel without duplicate. After user has click the start button the program will start counting if the initial patient count has been reached.
The output data has been separated into 2 categories, daily value and accumulated value. The simulation will be terminated when all the daily value has been set to 0 which mean no more action will happen during day after. There is a exception for the termination standard that the probability calculation module can decide that all the person around the initial patient did not get infected like figure 1 showed. In another word, the infection has stopped after the first day. In reality, this case is rare because evolution make most disease quite strong and if this case happen than most likely there is no use for the simulator. To prevent this rare situation happens, an counter has been implemented to monitor the number of times the initial termination situation has happened. If this situation has happened 10 times, the system will seems the program has done with the simulation. Because the startup phase has max 3 to 4 infection. Use 10 as exception limit is safe enough to judge the program has done or not.When the simulator has done with its current session, the program will show a message to inform user that the simulation has been done.

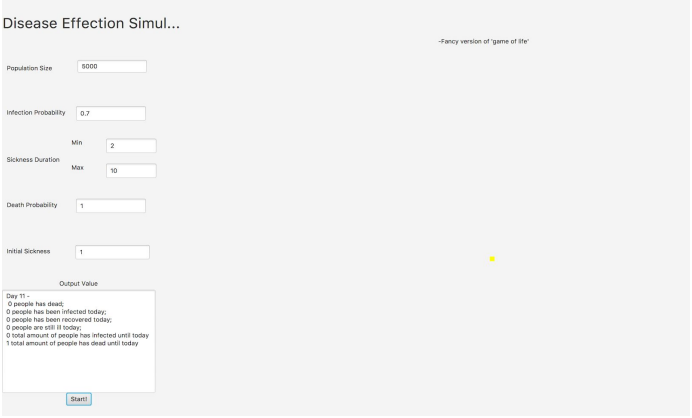


Figure 1. 1 spawn

To be able to limit the population, the 2 backend matrices and front end canvas should has certain boundary. There is a monitor module to check that If there is a spawn outside the boundary. When monitor has detected that there is a out of boundary spawn, the spawn process will be terminated and dropped. The monitor will initialized before all the spawn process started which is during the coordinate calculation phase.
The simulator has set the length of a day as 1000ms. The methodology for implement this functionality is by using animation timer, timer will tick every 1000ms. The most of the validation for each function has been done by java own debug functionality.
During the first infection day for a patient, the ‘day one’ infected person should not infect other person inside the fixed population.To prevent the ‘day one’ patient infect other person,the

probability check has global property that the probability check module will active once per day per person. The simulator will be terminated after each of the person has been through probability check once per day.
The validation during modelling has been done by constant testing the answer before implementation and after implementation. Before implementation, each step has been planned and pre-calculated on the paper. Theoretically each model should return the desired answer. The answer do not have to be precise but the answer should be accurate. The detail tuning can be done during implementation of the code. Because by than the validation can be done by java debug tool and most of unnecessary tuned calculation can be done by the computer.
Most of the validation during implementation of the planning for each function of the module used in the simulator has been done by java own debug functionality.

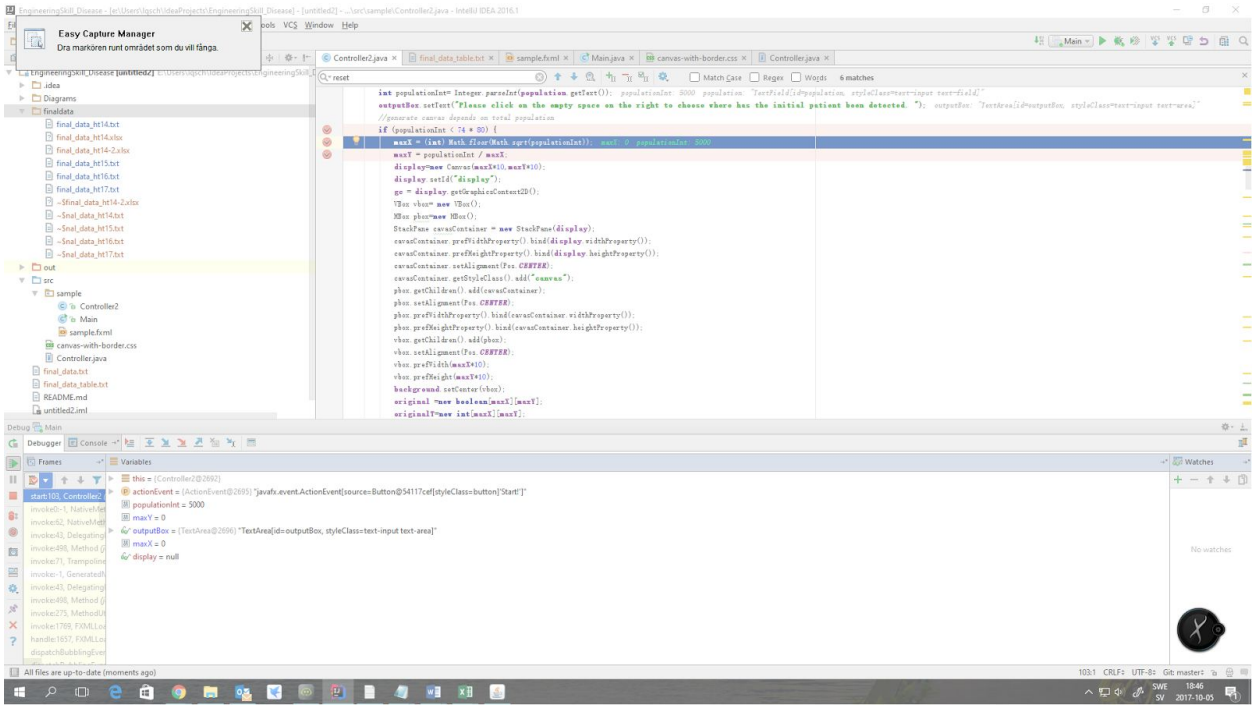


Figure 2. Java debug function

With setting end point at the LOC(Lines of Code) that need to be debugged. The program will run into the end point and start step into the function. This will show the variable and process number of current step in function as shown in figure 2. For example,the input 10000 as population, but at population checking step it is showing that the if statement still providing a true, that program has implemented wrong and the function is returning false result. At same time if function return false as answer, the program is implemented correctly.

Input parameters[1]

Semester	N	minDays	maxDays	L	Initial number of ill individuals	Placement
ht17	40	6	9	0	1	20,20

ht16	**50**	**1**	**12**	**0**	**1**	**25,25**
ht15	**50**	**3**	**9**	**0**	**1**	**25,25**
ht14	**50**	**4**	**8**	**0**	**1**	**25,25**

Total amount or sickness and length of disease[1]

Term	Population	Sickness duration	The length of Disease
HT14	40	6-->9	33
HT15	50	1-->12	39
HT16	50	3-->9	31
HT17	50	4-->8	48

Validation Methodology

Test cases

- Test the output in normal conditions.
- Observe the change in results, given that there are random values involved in the process.
- Test the extreme values where the probabilities are either 0 or 100% and see how does that differ from any random value in between
- Test increasing or decreasing ratios and see how that reflects on the graph and to make sure if it gives stable results.
- Change one of the requirements and observe the results of that.
- Since in the input data there is not a specification of the infection rate. The simulator use constant 0.1 as infection probability for all of the simulation.
- Test each function independently (module testing).
- Test how each function work together (integration testing).
- Try to combine each test result with common sense to see if it make sense
- Hand the software to some medical professional people and see how the test result goes.
- Consult disease specialist to see if the graphical and data representation make sense for them.
- Execute all test cases again each time a new functionality has been implemented, after some basic function development.

Output Validation

- During the whole testing phases the program only change 1 variable at a time and make other variable constant for more accurate test result.
- Each function result has been validation through java integrated debug function. The program has been validate through step into each function and validate if the variable returning right answer. The validation has been done by input normal value and extreme values.
- There is a slight difference between the outputs under the same circumstances due to the Disease spread is randomized as figure 3 shown below. For precise and accurate simulation. The simulator use binomial distribution during simulation. A single person can be sick or not is depends on the probability user input.The simulator will determine the probability according to binomial distribution module. The module has been implemented as random generate a number and set the input probability as a critical area. If this number falls inside the critical area, the module will return a hit otherwise this person survived.

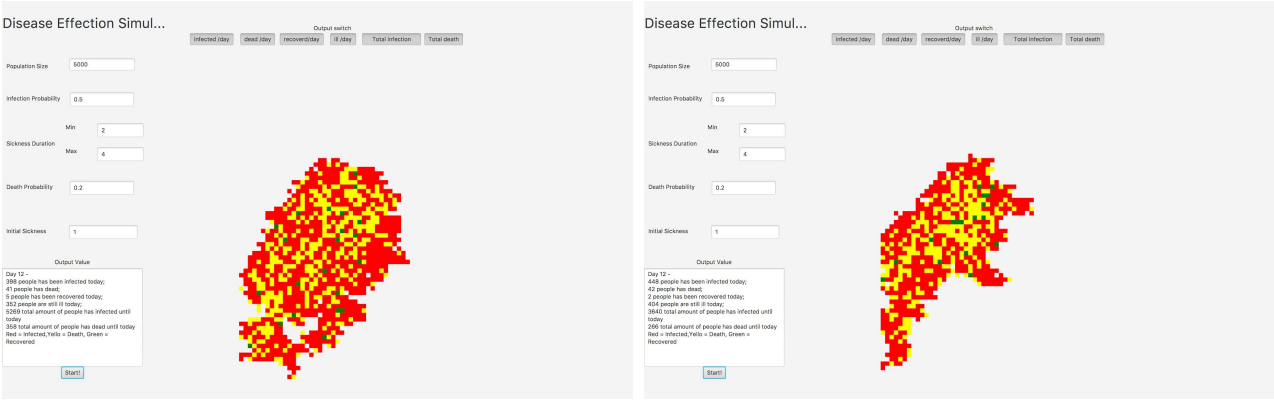


Figure 3 randomized output

- The extreme value testing will occur in both of the probability input which are infection probability and death probability. The validation result will decide base on if the visualization of the cavas output fits the real life cases. During minimum of infection probability, nobody got sick and during maximum probability everybody got sick. Same case goes for the death probability. Nobody is dying death when the probability is 0 as figure 5 and all the infected people are dead when death probability goes to maximum. These result are obtained from GUI interface of simulation process. The GUI has set grey as did not get involved during this infection, yellow as dead, red as infected and green as immuned/recovered. GUI can provide active support to verify each cases in a more visualized and obvious way. The GUI representation has been shown in figure 4 and 5 below. For example during the maximum infection probability testing,the program show that every person turned red when the program is terminated with few person dead and nobody is in grey which is survived the disease.

Figure 4. Death probability as 0

- The test case for each function is built on the fact that the result of each step is correct in a theoretical, mathematical and human sense point of view.
- The test for integration include, input all the possible cases include all the extreme possible one can thought of.
- The program has handed to one of the biology student and observe the feedback. The feedback contain few problems including the data inaccuracy and the spreading speed. The general data output and visualization of disease spreading pattern is correct. This

can be shown by the more the disease spreading gets close to the edge of the canvas, the more people are either immune or dead. The feedback also contain the fact that the spreading goes to one side of the canvas first before spreading to the other side of the canvas. This symptom is due to those people are in connection to each other.The symptom also explain why disease controller always start quarantine certain group of people that has connection with initial infection patient. Because quantain the people around the patient can effectively cut down the disease spread to the other side of the population.

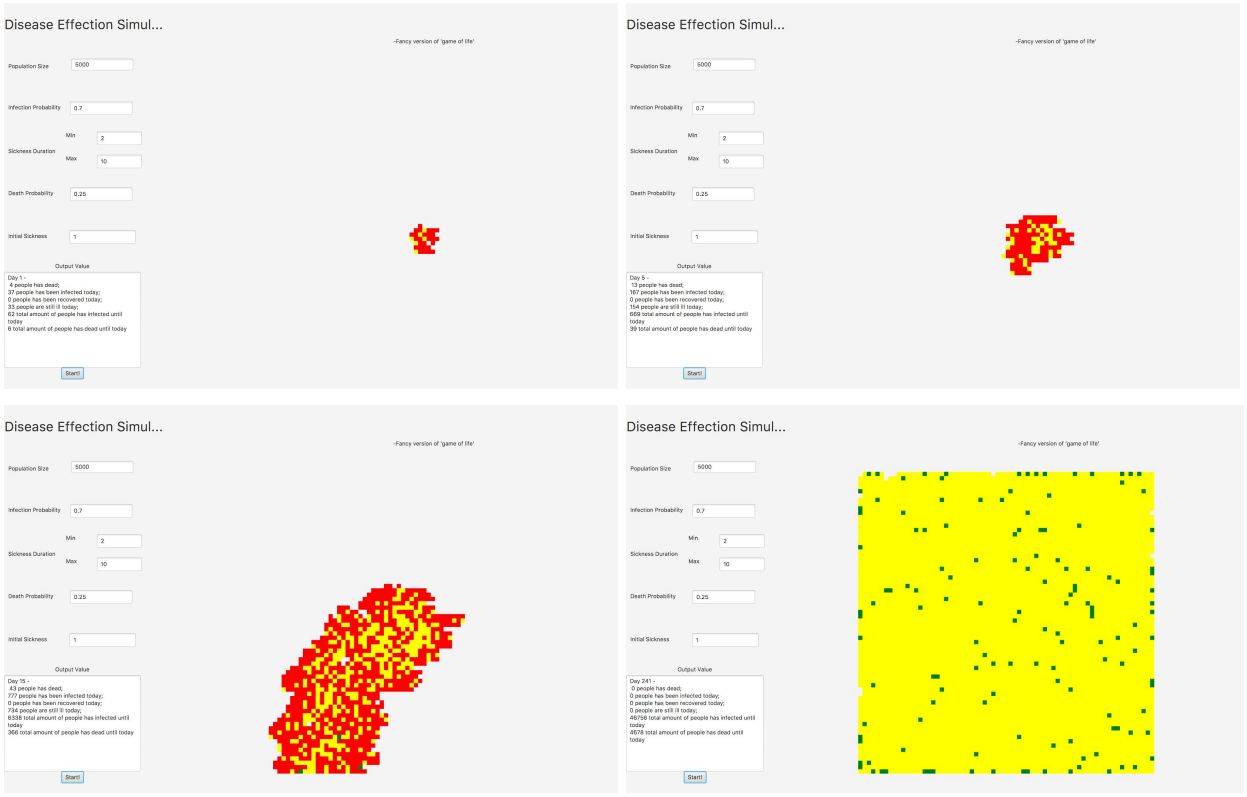


Figure 5. Max death rate

- After a basic function has been implemented, the simulator has been implemented some extra data collection features and correcting few bugs that can cause user make mistakes. After each of these functions has been implemented, all the test case has been going through again the beginning of the test case with the help of java debug tool.

Results and conclusions

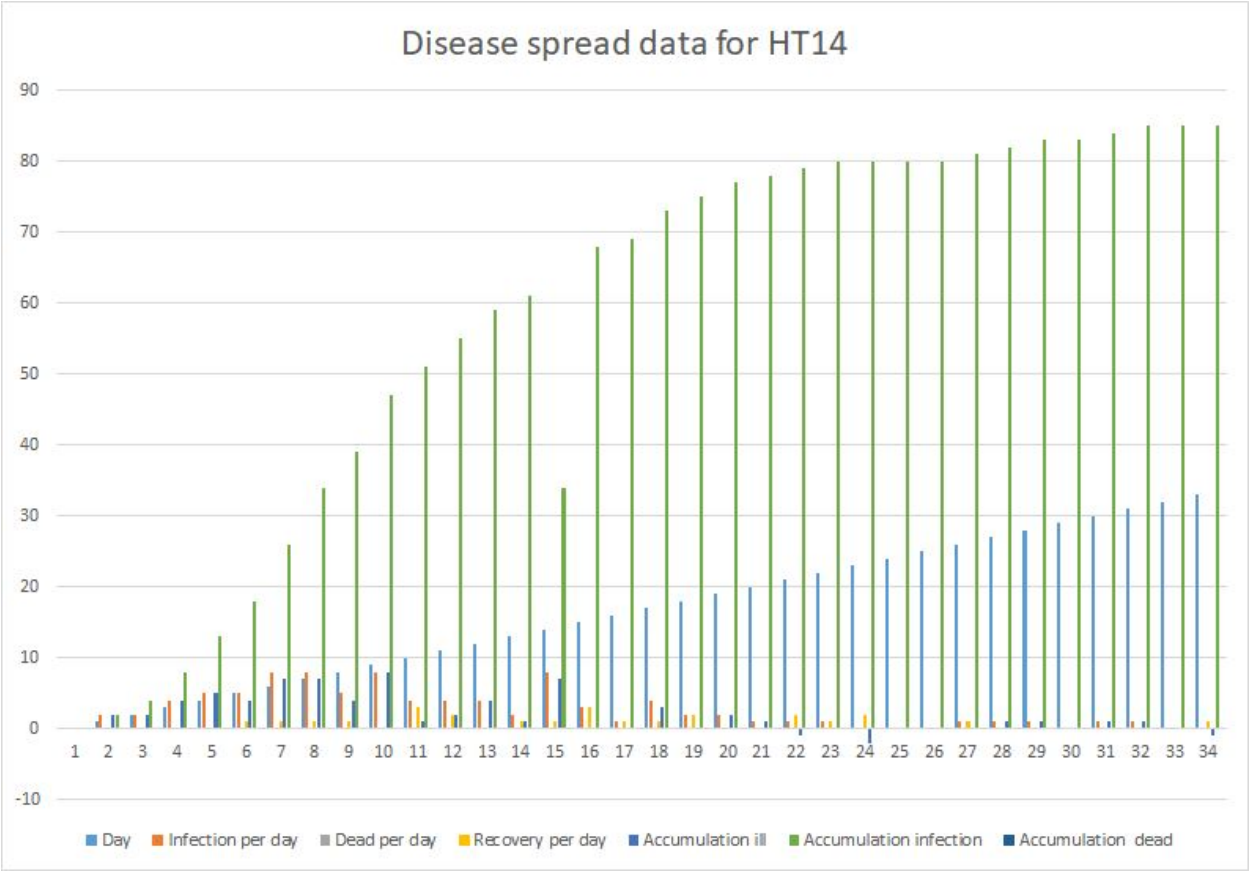


Figure 6.Graph for HT14

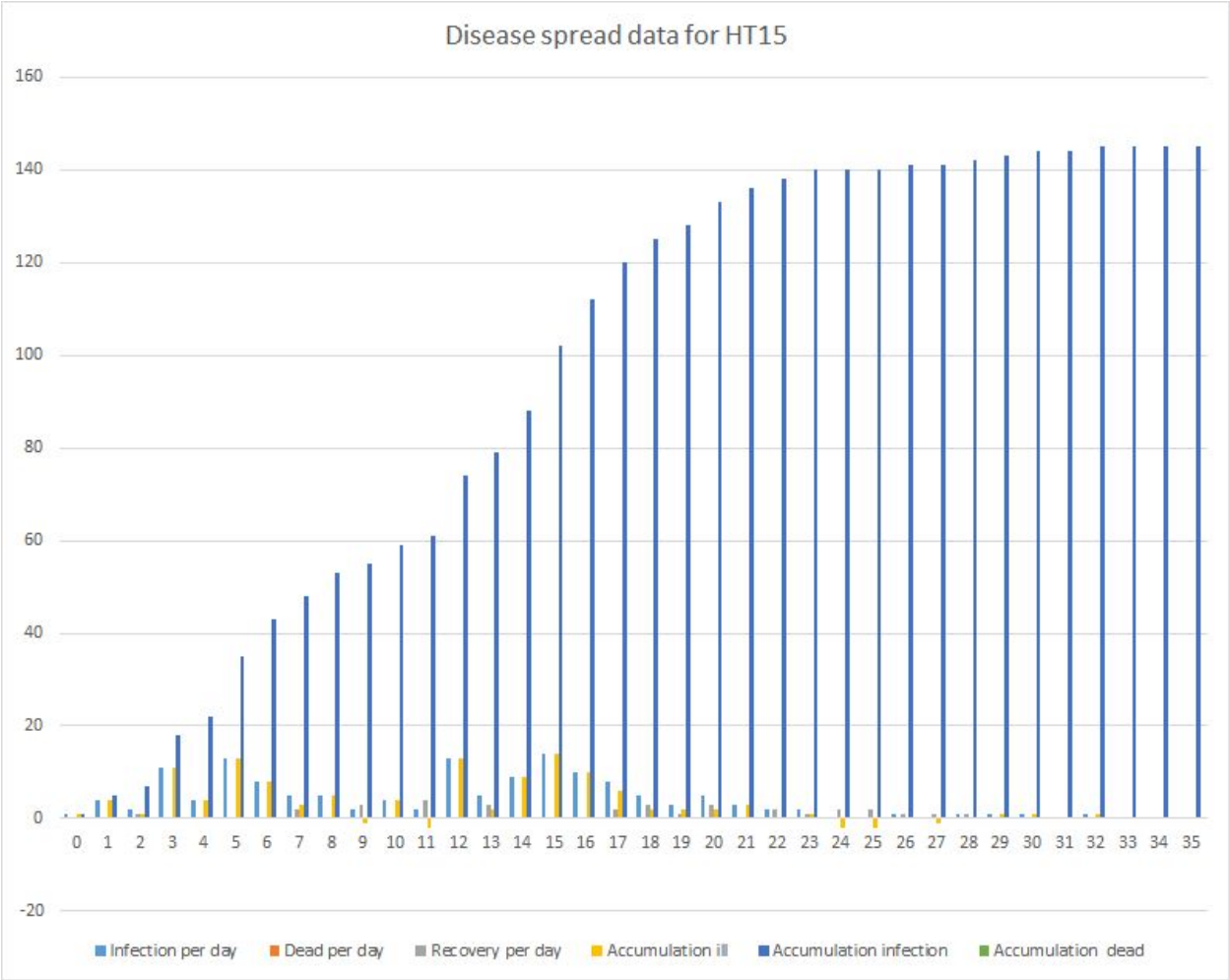


Figure 7.Graph for HT15

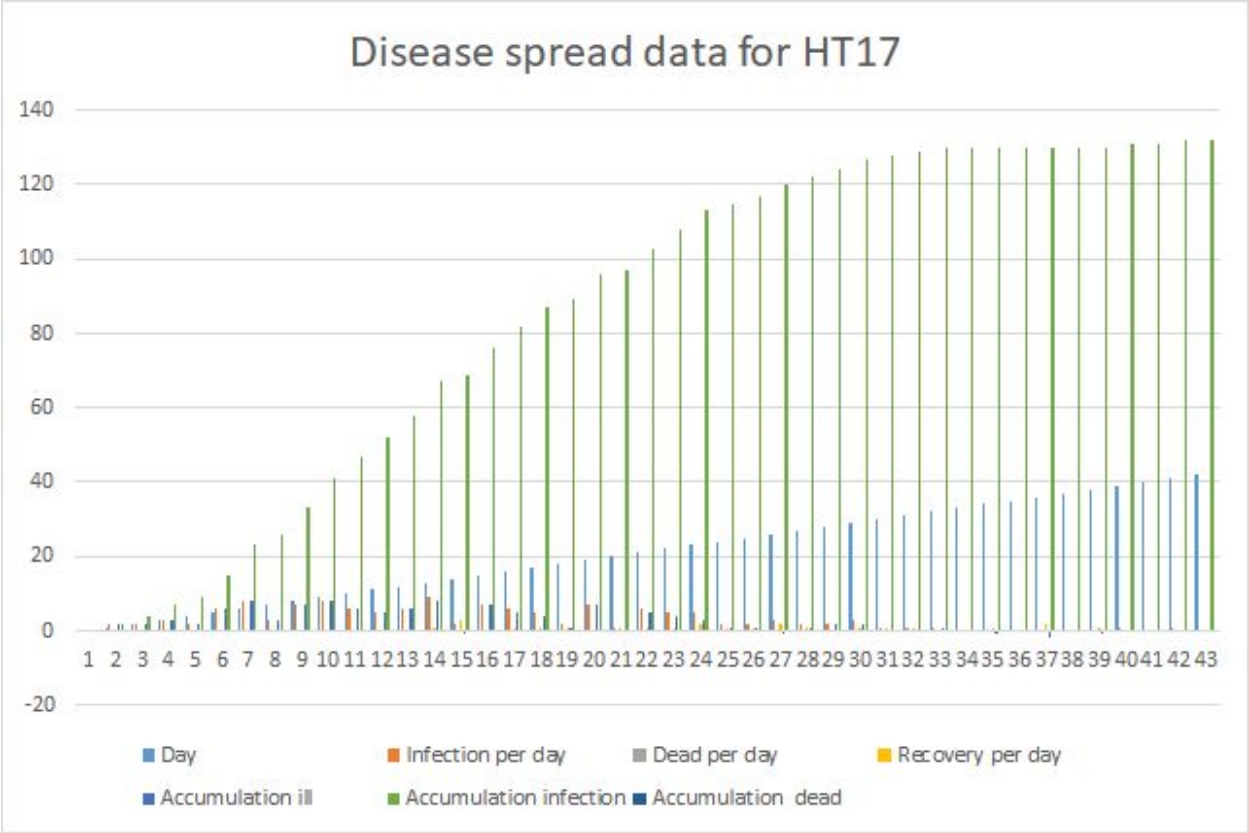


Figure 7.Graph for HT17

The figure above present with X=Days, Y= population in current category.In general, the graph has perform a log n pattern with a smooth population increment on accumulation infection. The infection per day has shown as a parabola with maximum at day 8 at ht14 and day 14 at ht 17. The maximum of infection per day for ht 16 has two maximum at day 12 and day 5. Accumulation dead and ill in all 3 graph has been shown as increase with a straight line.

During the simulation between 4 different input values, the output data shows a pattern that the bigger the population and sickness duration is, the longer time the disease spreads. The sickness duration has also effect on the average recovery time. The pattern has been shown on the simulator that the green spot on the window is clearly getting less or is taking a longer time to show up on the canvas when one increase the duration time. The pattern can also happen when the minimum sickness days is increased. One exception has happen during the ht17 when result of the sickness duration is getting lower but the total days of disease spread is getting higher. The reason of the exception is unknown. The exception remain in different trial of the simulation.

The threshold of the infection can exist and the threshold is depends on the infection probability,sickness duration and death probability. When a patient is dead, they will no longer be able to infect other people. This state is basically same as immuned but the

simulation stated as dead. So the death rate is depend on sickness duration.The shorter the sickness duration the higher possibility that one can be survive before infect other people.
A test has performed that input infection probability as a fixed value and constantly increasing death probability or lower the sickness duration. The test showed us that in the most extreme circumstances, only 1 or 2 spawned with dead or immuned right before infected person spread the disease to anybody else. The test result are completely independent to the population too. The purpose of this test is to verify if the system is working properly. Because the symptom of the test looks like the simulator crashing and only spawn one or two patients before simulation terminated. The debug tool by java has been chosen as validation tool for the test. During the validation, one step in the function to see if the function is doing the right thing . After the test,one define the symptom from the test as extreme circumstances and has been categorized in exception during developing, data collection and data validation.

How to compile and run/use the program

To be able to compile the program, it is required to have to Official Oracle Java JDK 1.8.0 installed with JavaFX plugin. OpenJDK will not work unless user build OpenJFX on the machine. For how to do build OpenJFK please visit website:https://wiki.openjdk.java.net/display/OpenJFX/Building+OpenJFX.

User should be able to compile this program in all IDE but Intellij is strongly recommended because all the setting has been preset well by the Intellij.
Input the initial Sickness population, Probability of infection , minimum and maximum sickness days, the probability of death and how many initial sickness people have found before simulation start. Than press the start button in the end of the window. Click inside the black boundary to choose where user want the initial sickness people spawn. The software will automatically start the simulation session after the total amount of initial sickness people has successfully spawned. The final data can be found at final_data.txt file. The file has been set with good formatting and record the result per day. Daily results can also been found in the output window at the left corner of the program.

Location of the source code

KTH AFS:

The 2 file below are the same, but belong to the each team member.

Qi Li

\\AFS\kth.se\home\q\i\qi5\untitled2

Hussam hassanein

\\AFS\kth.se\home\h\u\hussamh\untitled2

Git

SSH:git@github.com:GiantPanda0090/EngineeringSkill_Disease.git

Reference

[1]Simulation how an infection spreads in a population v1.1. (2018). https://www.kth.se/social/files/59bb84e956be5b2bc7ce4268/Labb%20-%20Simulering%20av%20smittspridning%20english.pdf, [online] pp.1-2. Available at: https://www.kth.se/social/files/59bb84e956be5b2bc7ce4268/Labb%20-%20Simulering%20av%20smittspridning%20english.pdf [Accessed 17 Jan. 2018].

Appendices

Requirements Analysis

The general requirement is to create a system that simulates the process of the spread of an infection in a population.
We build a system to help solving the problem, according to some requirements, which will be specified in the following table.

Description of requirement	**What is required to satisfy the requirement**	**To what degree the requirement is fulfilled**
The population is modeled as an NxN matrix	Population is represented in a table with equal number of rows and columns	Completely fulfilled
The disease to be simulated is spread by direct contact between individuals	The infection process is to occur only when there is a direct contact between individuals, meaning that you can only infect your neighbors.	Completely fulfilled
An individual is ill for a random number of days during which time there is a probability that the individual infects each of its direct neighbors (normally 8) in the population	The duration of infection is randomized. There is a probability of infection to 8 neighbors in 3x3 grid.	Completely fulfilled
Every day an individual is ill, there is also a probability that the individual dies.	There should a probability assumed for the ill individual to be dead, which gets updated everyday.	Completely fulfilled
An individual who dies is not contagious	If an individual dies, they can’t infect others anymore.	Completely fulfilled
A sick person cannot be infected	If an individual has been infected they can’t be infected again. In other words you either recover or die.	Completely fulfilled

An individual who has been ill but recovered, is immune and cannot be infected again.	If an individual is recovered they can't change their state at all. They will stay recovered for ever.	Completely fulfilled
Some input values are to be decided by the user	An input is to be required in the start of the process	Completely fulfilled

Outline

We have created a system, that simulates how an infection spreads in population. The system had to follow the requirements set by the client, which outlined the basics of this project. On the other hand, the way to solve the problem has been improving during the whole process of building the system. Building the system from scratch was a challenge in the start, because of us using modelling, which provides a different perspective to the problem. The validation and verification stage has helped a lot in deepen the understanding of the problem and made sure that everything works as intended. In general, The problem we have handled could have a lot more depth into it in real life, meaning that there are a lot more attributes and requirements that the problem could be based on. The model we have built deals with infection problems, according to the requirements specified by the client.

Source code

JavaFX Main Class:

```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Main extends Application {

   @Override
   public void start(Stage primaryStage) throws Exception{
       Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
       primaryStage.setTitle("Hello World");
       Scene scene =new Scene(root, 1124, 845);
       scene.getStylesheets().add("canvas-with-border.css");
       primaryStage.setScene(scene);
       primaryStage.show();
   }


   public static void main(String[] args) {
       launch(args);
   }
}
```

JavaFX Controller Class:

```
package sample;

/*
Controller Class
```

```
*/

import javafx.animation.AnimationTimer;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.control.ToggleButton;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;

import java.io.PrintWriter;
import java.net.URL;
import java.util.Random;
import java.util.ResourceBundle;

/**
* Created by lqsch on 2017-10-01.
*/
public class Controller2 implements Initializable {
    @FXML
    public TextField init;

    @FXML
    public TextField death;

    @FXML
    public TextField Min;
    @FXML
    public TextField Max;

    @FXML
    public TextField infection;

    @FXML
    public TextField population;

    @FXML
    public TextArea outputBox;

    @FXML
    public BorderPane background;

    @FXML
    public ToggleButton infectSW;
```

```
    @FXML
    public ToggleButton deadSW;
    @FXML
    public ToggleButton recoverySW;
    @FXML
    public ToggleButton illSW;
    @FXML
    public ToggleButton tInfectSW;
    @FXML
    public ToggleButton tDeathSW;

    /* non fxml init*/
    PrintWriter writer;
    PrintWriter tableWriter;


    //output
    private int days;
    private int deathCounter;
    private int infectedCounter;
    private int recoverCounter;
    private int illCounter;
    private int accumInfecC;
    private int accumDeathcC;


    public GraphicsContext gc;
    public int maxY;
    public int maxX;
    public boolean Xover;
    public boolean Yover;

    public int initInt;
    public int counter;

    private long lastUpdate;
    boolean[][] original;
    int[][] originalT;
    Canvas display;
    int[][] mouseRec;
    public int stopcounter;
    @FXML
    public void start(ActionEvent actionEvent) {
        //initialization for start button
        int populationInt= Integer.parseInt(population.getText());
        outputBox.setText("Please click on the empty space on the right to choose where has the
initial patient been detected. ");
        //generate canvas depends on total population
        if (populationInt < 74 * 80) {
            maxX = (int) Math.floor(Math.sqrt(populationInt));
```

```
            maxY = populationInt / maxX;
            display=new Canvas(maxX*10,maxY*10);
            display.setId("display");
            gc = display.getGraphicsContext2D();
            VBox vbox= new VBox();
            HBox pbox=new HBox();
            StackPane cavasContainer = new StackPane(display);
            cavasContainer.prefWidthProperty().bind(display.widthProperty());
            cavasContainer.prefHeightProperty().bind(display.heightProperty());
            cavasContainer.setAlignment(Pos.CENTER);
            cavasContainer.getStyleClass().add("canvas");
            pbox.getChildren().add(cavasContainer);
            pbox.setAlignment(Pos.CENTER);
            pbox.prefWidthProperty().bind(cavasContainer.widthProperty());
            pbox.prefHeightProperty().bind(cavasContainer.heightProperty());
            vbox.getChildren().add(pbox);
            vbox.setAlignment(Pos.CENTER);
            vbox.prefWidth(maxX*10);
            vbox.prefHeight(maxY*10);
            background.setCenter(vbox);
            original =new boolean[maxX][maxY];
            originalT=new int[maxX][maxY];
        } else {
            outputBox.setText("Population size is too big. it should be less that 70*77 ");
            return;
        }
        int initInt= Integer.parseInt(init.getText());
        mouseRec= new int[initInt][2];
        //initialize people get sick and where are they
        final int[] xAxis = new int[1];
        final int[] yAxis = new int[1];
        int[][] record=new int[initInt][2];

        display.setOnMouseClicked(event -> {
             xAxis[0] =(int) event.getX()/10;
             yAxis[0] =(int) event.getY()/10;
             xAxis[0]=xAxis[0]*10;
             yAxis[0]=yAxis[0]*10;
            boolean[][] copy =original;
            infectedCounter++;
            updateCanvas(xAxis[0], yAxis[0],copy);
            original=copy;
            counter++;
        });
        //end of initialization
        //log input value into log file
        writer.print("Result for Testing value: ");
        writer.println("Total Population: "+ population.getText() +" Infected Probability: "+
infection.getText() + " Days of sickness duration: From "+ Min.getText()+" till "+
Max.getText() +" days.");
```

```
        writer.println("Death Probability: "+ death.getText()+" Initial Sickness Porpulation:
"+ init.getText());
        //start simulation
        AnimationTimer timer = new AnimationTimer() {
            @Override
            public void handle(long now) {
                display.setOnMouseEntered(event -> {
                    outputBox.setText("Mouse X: "+(int) event.getX()/10+"Mouse Y: "+(int)
event.getY()/10);
                });
                //if total number of initilized people are successfully been input and accepted
                if (counter>=initInt) {
                    display.setOnMouseClicked(null);
                    display.setOnMouseEntered(null);
                    //start a new day per second
                    if (now - lastUpdate >= 1000_000_000) {
                        resetAllDCounter();
                        for (int y = 0; y < maxY; y++) {
                        for (int x = 0; x < maxX; x++) {
                            if (original[x][y] == true) {

                                //death
                                if(death(x,y)){
                                    break;
                                }

                                //healing
                               if (healing(x,y)){
                                    break;
                                }
                                //probability go though neighbor cell(8 of them)
                                if (x > 0 + 1 && y > 0 + 1 && x < maxX - 1 && y < maxY - 1) {
                                    //8 neighbors
                                    boolean[][] copy = original;//copy
                                    for (int diffy = -1; diffy <= 1; diffy++) {
                                        int Ny = y + diffy;
                                        for (int diffx = -1; diffx <= 1; diffx++) {
                                            int Nx = x + diffx;
                                            if (originalT[Nx][Ny] >= 0) {
                                                //infected
                                                stopcounter=0;
                                                infected(Nx,Ny,copy);
                                            }
                                        }
                                    }
                                    original = copy;
                                }
                            }
                        }
                    }
```

```
                        lastUpdate = now ;
                        illCounter=illCounter+infectedCounter-deathCounter-recoverCounter;
                        accumInfecC=accumInfecC+infectedCounter;
                        accumDeathcC=accumDeathcC+deathCounter;
                        //output
                        outFX();
                        days++;

if(infectedCounter==0&&deathCounter==0&&recoverCounter==0&&illCounter==0) {
                            stopcounter++;//quickfix
                        }
                        if (stopcounter> 10){
                           writer.close();
                           tableWriter.close();
                           resetAllDCounter();
                           resetAllOth();
                           this.stop();
                           outputBox.setText(outputBox.getText()+"\n Simulation has done!!");

                        }
                }
                    }
                }


        };

            timer.start();

    }
    public void outFX(){
        outputBox.setText("Day "+ days+" - \n");
        tableWriter.println(" - ");
        tableWriter.print(" | ");
        tableWriter.println(days+" | ");

        if(infectSW.isSelected()){
            outputBox.setText(outputBox.getText()+infectedCounter+" people has been infected
today; \n");
            tableWriter.println(" - ");
            tableWriter.println(infectedCounter+" | ");
        }
        if(deadSW.isSelected()){
            outputBox.setText(outputBox.getText()+deathCounter+" people has dead today; \n");
            tableWriter.println(" - ");
            tableWriter.println(deathCounter+" | ");
        }
        if(recoverySW.isSelected()){
            outputBox.setText(outputBox.getText()+recoverCounter+" people has been recovered
today; \n");
```

```
            tableWriter.println(" - ");
            tableWriter.println(recoverCounter+" | ");

        }
        if(illSW.isSelected()){
            outputBox.setText(outputBox.getText()+illCounter+" people are still ill today; \n"
);
            tableWriter.println(" - ");
            tableWriter.println(illCounter+" | ");


        }
        if(tInfectSW.isSelected()){
            outputBox.setText(outputBox.getText()+accumInfecC+" total amount of people has
infected until today \n" );
            tableWriter.println(" - ");
            tableWriter.println(accumInfecC+" | ");

        }
        if(tDeathSW.isSelected()){
            outputBox.setText(outputBox.getText()+accumDeathcC+" total amount of people has
dead until today \n");
            tableWriter.println(" - ");
            tableWriter.println(accumDeathcC+" | ");

        }
        outputBox.setText(outputBox.getText()+"Red = Infected,Yello = Death, Green = Recovered
\n");
        //log output data
        writer.println(outputBox.getText());

writer.println("================================================================================
========");
    }
    //infected method
    public boolean infected(int Nx,int Ny,boolean[][]copy){
        Random rand = new Random();
        double test = rand.nextDouble();
        if (test < Double.parseDouble(infection.getText())) {
            infectedCounter++;
            updateCanvas(Nx * 10, Ny * 10, copy);
            return true;
        }
        return false;
    }

    //death method
    public boolean death(int x,int y){
        //death
        Random dRand = new Random();
```

```
    double dTest = dRand.nextDouble();
    if (dTest < Double.parseDouble(death.getText())) {
        originalT[x][y] = -2;
        original[x][y] = false;
        deathCounter++;
        gc.setFill(Color.YELLOW);
        gc.fillRect(x * 10, y * 10, 10, 10);
        return true;
    }
    return false;
}

//healing method
public boolean healing(int x, int y){
    originalT[x][y] = originalT[x][y] - 1;
    if (originalT[x][y] == 0) {
        original[x][y] = false;
        originalT[x][y] = -1;
        recoverCounter++;
        gc.setFill(Color.GREEN);
        gc.fillRect(x * 10, y * 10, 10, 10);
        //gc.clearRect(x* 10,y* 10,10,10);
        return true;
    }
    return false;
}

//before pressing the start button
//initial state
@Override
public void initialize(URL location, ResourceBundle resources) {
    //geneneral initialization for overall application
    stopcounter=0;
    days=0;
    deathCounter=0;
    infectedCounter=0;
    recoverCounter=0;
    illCounter=0;
    accumInfecC=0;
    accumDeathcC=0;
    lastUpdate = 0;
    infectSW.setSelected(true);
    deadSW.setSelected(true);
    recoverySW.setSelected(true);
    illSW.setSelected(true);
    tInfectSW.setSelected(true);
    tDeathSW.setSelected(true);
    try {
        writer = new PrintWriter("final_data.txt", "UTF-8");
        tableWriter= new PrintWriter("final_data_table.txt", "UTF-8");
```

```
        }catch(Exception e ){
            System.err.println(e);
        }
        //default value
        population.setText("5000");
        infection.setText("0.5");
        Min.setText("2");
        Max.setText("4");
        death.setText("0.2");
        init.setText("1");
        //user mistake elimination
        outputBox.setText("Please input the initial value for each box. \n"+"For probability
please use decimal number instead of percentage.\nFor example 0.75 instead of 75 percent. ");
    }
    public void resetAllOth(){
        original=null;
        originalT=null;
        mouseRec=null;
        maxX=0;
        maxY=0;
        lastUpdate=0;
        counter=0;
        stopcounter=0;
        accumDeathcC=0;
        accumInfecC=0;


    }
    //reset the counters for yesterday
    public void resetAllDCounter(){
        deathCounter=0;
        infectedCounter=0;
        recoverCounter=0;
        illCounter=0;


    }

    //gui update per change
    public void updateCanvas(int x,int y,boolean[][]copy){
            final int xReal = x;
            final int yReal = y;
            gc.setFill(Color.RED);
            gc.fillRect(xReal, yReal, 10, 10);
            copy[x / 10][y / 10] = true;
       int minT= Integer.parseInt(Min.getText());
       int maxT=Integer.parseInt(Max.getText());
       Random durationR= new Random();
        originalT[x/10][y/10]=durationR.nextInt(maxT-minT) + minT;
```

```
    }

}
```

JavaFX FXML xml file:

```
<?xml version="1.0" encoding="UTF-8"?>

<!--Authors lqsch Qi Li, Hussam Hassanein-->
<!--Front end InterfaceXML file-->
<!--Download JavaFX 8.0 plus SceneBuilder before edit it-->
<!--suggested to edit it in Intellij with SceneBuilder-->
<!--Right click the fxml file on the project list in the right and choose Open in Scene
Builder-->

<?import javafx.scene.canvas.*?>
<?import javafx.scene.shape.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.text.*?>
<?import java.lang.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="845.0" prefWidth="1124.0" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller2">
  <left>
     <VBox prefHeight="845.0" prefWidth="321.0" BorderPane.alignment="CENTER">
        <children>
           <HBox prefHeight="77.0" prefWidth="321.0">
              <children>
                 <VBox prefHeight="75.0" prefWidth="151.0">
                    <children>
                       <Label prefHeight="24.0" prefWidth="133.0" text="Population Size">
                          <VBox.margin>
                             <Insets left="10.0" top="35.0" />
                          </VBox.margin>
                       </Label>
                    </children>
                    <HBox.margin>
                       <Insets />
                    </HBox.margin>
                 </VBox>
                 <VBox prefHeight="75.0" prefWidth="152.0">
                    <children>
                       <TextField fx:id="population" onAction="#start">
```

```
                        <VBox.margin>
                            <Insets left="3.0" top="30.0" />
                        </VBox.margin>
                    </TextField>
                </children>
            </VBox>
        </children>
    </HBox>
    <HBox layoutX="10.0" layoutY="10.0" prefHeight="100.0" prefWidth="200.0">
        <children>
            <VBox prefHeight="100.0" prefWidth="152.0">
                <children>
                    <Label prefHeight="16.0" prefWidth="142.0" text="Infection
Probability">
                        <VBox.margin>
                            <Insets left="10.0" top="35.0" />
                        </VBox.margin>
                    </Label>
                </children>
                <HBox.margin>
                    <Insets />
                </HBox.margin>
            </VBox>
            <VBox prefHeight="100.0" prefWidth="149.0">
                <children>
                    <TextField fx:id="infection" prefHeight="31.0" prefWidth="167.0">
                        <VBox.margin>
                            <Insets left="3.0" top="30.0" />
                        </VBox.margin>
                    </TextField>
                </children>
            </VBox>
        </children>
    </HBox>
    <HBox layoutX="10.0" layoutY="110.0" prefHeight="100.0" prefWidth="200.0">
        <children>
            <VBox prefHeight="94.0" prefWidth="138.0">
                <children>
                    <Label prefHeight="21.0" prefWidth="135.0" text="Sickness Duration">
                        <VBox.margin>
                            <Insets left="10.0" top="35.0" />
                        </VBox.margin>
                    </Label>
                </children>
                <HBox.margin>
                    <Insets />
                </HBox.margin>
            </VBox>
            <VBox layoutX="10.0" layoutY="10.0" prefHeight="94.0" prefWidth="184.0">
                <children>
```

```
            <HBox prefHeight="54.0" prefWidth="184.0">
               <children>
                  <Label prefHeight="16.0" prefWidth="142.0" text="  Min" />
                  <TextField fx:id="Min" prefHeight="31.0" prefWidth="167.0" />
               </children>
            </HBox>
            <HBox prefHeight="41.0" prefWidth="57.0">
               <children>
                  <Label prefHeight="16.0" prefWidth="142.0" text="  Max" />
                  <TextField fx:id="Max" prefHeight="31.0" prefWidth="167.0" />
               </children>
            </HBox>
         </children>
      </VBox>
   </children>
</HBox>
<HBox layoutX="10.0" layoutY="210.0" prefHeight="100.0" prefWidth="200.0">
   <children>
      <VBox prefHeight="100.0" prefWidth="152.0">
         <children>
            <Label prefHeight="16.0" prefWidth="142.0" text="Death Probability">
               <VBox.margin>
                  <Insets left="10.0" top="35.0" />
               </VBox.margin>
            </Label>
         </children>
         <HBox.margin>
            <Insets />
         </HBox.margin>
      </VBox>
      <VBox prefHeight="100.0" prefWidth="149.0">
         <children>
            <TextField fx:id="death" prefHeight="31.0" prefWidth="167.0">
               <VBox.margin>
                  <Insets left="3.0" top="30.0" />
               </VBox.margin>
            </TextField>
         </children>
      </VBox>
   </children>
</HBox>
<HBox layoutX="10.0" layoutY="310.0" prefHeight="100.0" prefWidth="200.0">
   <children>
      <VBox prefHeight="94.0" prefWidth="151.0">
         <children>
            <Label prefHeight="16.0" prefWidth="142.0" text="Initial Sickness">
               <VBox.margin>
                  <Insets left="10.0" top="35.0" />
               </VBox.margin>
            </Label>
```

```
                    </children>
                    <HBox.margin>
                        <Insets />
                    </HBox.margin>
                </VBox>
                <VBox prefHeight="94.0" prefWidth="158.0">
                    <children>
                        <TextField fx:id="init" prefHeight="31.0" prefWidth="131.0">
                            <VBox.margin>
                                <Insets left="3.0" top="30.0" />
                            </VBox.margin>
                        </TextField>
                    </children>
                </VBox>
            </children>
        </HBox>
        <VBox prefHeight="249.0" prefWidth="321.0">
            <children>
                <HBox alignment="CENTER" prefHeight="29.0" prefWidth="321.0">
                    <children>
                        <Label text="Output Value" />
                    </children>
                </HBox>
                <HBox alignment="CENTER" prefHeight="225.0" prefWidth="321.0">
                    <children>
                        <TextArea fx:id="outputBox" prefHeight="223.0" prefWidth="324.0"
wrapText="true" />
                    </children>
                </HBox>
            </children>
        </VBox>
        <HBox alignment="CENTER" layoutX="10.0" layoutY="406.0" prefHeight="29.0"
prefWidth="321.0">
            <children>
                <Button mnemonicParsing="false" onAction="#start" text="Start!" />
            </children>
        </HBox>
    </children>
    <BorderPane.margin>
        <Insets />
    </BorderPane.margin>
</VBox>
</left>
<top>
<HBox prefHeight="100.0" prefWidth="200.0" BorderPane.alignment="CENTER">
    <children>
        <HBox prefHeight="100.0" prefWidth="1126.0">
            <children>
                <HBox alignment="CENTER" prefHeight="51.0" prefWidth="385.0">
                    <children>
```

```
                        <Label prefHeight="48.0" prefWidth="384.0" text="Disease Effection
Simulator " textAlignment="CENTER">
                           <font>
                              <Font size="31.0" />
                           </font>
                        </Label>
                     </children>
                  </HBox>
                  <HBox alignment="BOTTOM_RIGHT" prefHeight="100.0" prefWidth="738.0">
                     <children>
                        <BorderPane prefHeight="100.0" prefWidth="749.0">
                           <bottom>
                              <HBox alignment="CENTER" prefHeight="24.0" prefWidth="738.0"
BorderPane.alignment="CENTER">
                                 <children>
                                    <HBox prefHeight="31.0" prefWidth="115.0">
                                       <children>
                                          <ToggleButton fx:id="infectSW"
mnemonicParsing="false" text="infected /day" />
                                       </children>
                                    </HBox>
                                    <HBox prefHeight="31.0" prefWidth="90.0">
                                       <children>
                                          <ToggleButton fx:id="deadSW" mnemonicParsing="false"
text="dead /day" />
                                       </children>
                                    </HBox>
                                    <HBox prefHeight="31.0" prefWidth="115.0">
                                       <children>
                                          <ToggleButton fx:id="recoverySW"
mnemonicParsing="false" text="recoverd/day" />
                                       </children>
                                    </HBox>
                                    <HBox prefHeight="31.0" prefWidth="73.0">
                                       <children>
                                          <ToggleButton fx:id="illSW" mnemonicParsing="false"
text="ill /day" />
                                       </children>
                                    </HBox>
                                    <HBox prefHeight="31.0" prefWidth="142.0">
                                       <children>
                                          <ToggleButton fx:id="tInfectSW"
mnemonicParsing="false" prefHeight="31.0" prefWidth="134.0" text="Total infection" />
                                       </children>
                                    </HBox>
                                    <HBox prefHeight="31.0" prefWidth="115.0">
                                       <children>
                                          <ToggleButton fx:id="tDeathSW"
mnemonicParsing="false" text="Total death" />
                                       </children>
```

```
                                    </HBox>
                                </children>
                            </HBox>
                        </bottom>
                        <center>
                            <HBox alignment="BOTTOM_CENTER" prefHeight="1.0"
prefWidth="738.0" BorderPane.alignment="CENTER">
                                <children>
                                    <Label alignment="CENTER" contentDisplay="CENTER"
prefHeight="24.0" prefWidth="169.0" text="Output switch" />
                                </children>
                            </HBox>
                        </center>
                    </BorderPane>
                </children>
            </HBox>
        </children>
    </HBox>
  </top>
  <center>
    <BorderPane fx:id="background" prefHeight="200.0" prefWidth="200.0"
BorderPane.alignment="CENTER" />
  </center>
</BorderPane>
```

Raw data

Output data

Simulation data for HT14

Day	Infection per day	Dead per day	Recovery per day	Accumulation ill	Accumulation infection	Accumulation dead
0	0	0	0	0	0	0
1	2	0	0	2	2	0
2	2	0	0	2	4	0
3	4	0	0	4	8	0
4	5	0	0	5	13	0
5	5	0	1	4	18	0
6	8	0	1	7	26	0
7	8	0	1	7	34	0
8	5	0	1	4	39	0
9	8	0	0	8	47	0
10	4	0	3	1	51	0
11	4	0	2	2	55	0
12	4	0	0	4	59	0
13	2	0	1	1	61	0
14	8	0	1	7	34	0
15	3	0	3	0	68	0
16	1	0	1	0	69	0
17	4	0	1	3	73	0
18	2	0	2	0	75	0
19	2	0	0	2	77	0
20	1	0	0	1	78	0
21	1	0	2	−1	79	0
22	1	0	1	0	80	0
23	0	0	2	−2	80	0
24	0	0	0	0	80	0
25	0	0	0	0	80	0
26	1	0	1	0	81	0
27	1	0	0	1	82	0
28	1	0	0	1	83	0
29	0	0	0	0	83	0
30	1	0	0	1	84	0
31	1	0	0	1	85	0
32	0	0	0	0	85	0
33	0	0	1	−1	85	0

Simulation data for HT15

Day	Infection per day	Dead per day	Recovery per day	Accumulation ill	Accumulation infection	Accumulation dead
0	1	0	0	1	1	0
1	4	0	0	4	5	0
2	2	0	1	1	7	0
3	11	0	0	11	18	0
4	4	0	0	4	22	0
5	13	0	0	13	35	0
6	8	0	0	8	43	0
7	5	0	2	3	48	0
8	5	0	0	5	53	0
9	2	0	3	−1	55	0
10	4	0	0	4	59	0
11	2	0	4	−2	61	0
12	13	0	0	13	74	0
13	5	0	3	2	79	0
14	9	0	0	9	88	0
15	14	0	0	14	102	0
16	10	0	0	10	112	0
17	8	0	2	6	120	0
18	5	0	3	2	125	0
19	3	0	1	2	128	0
20	5	0	3	2	133	0
21	3	0	0	3	136	0
22	2	0	2	0	138	0
23	2	0	1	1	140	0
24	0	0	2	−2	140	0
25	0	0	2	−2	140	0
26	1	0	1	0	141	0
27	0	0	1	−1	141	0
28	1	0	1	0	142	0
29	1	0	0	1	143	0
30	1	0	0	1	144	0
31	0	0	0	0	144	0
32	1	0	0	1	145	0
33	0	0	0	0	145	0
34	0	0	0	0	145	0
35	0	0	0	0	145	0

Simulation data for HT16

Day	Infection per day	Dead per day	Recovery per day	Accumulation ill	Accumulation infection	Accumulation dead
0	3	0	0	3	3	0
1	0	0	1	−1	3	0
2	4	0	0	4	7	0
3	2	0	2	0	9	0
4	0	0	3	−3	9	0
5	3	0	0	3	12	0
6	3	0	1	2	15	0
7	0	0	1	−1	15	0
8	1	0	0	1	16	0
9	3	0	0	3	19	0
10	0	0	1	−1	19	0
11	4	0	0	4	23	0
12	5	0	1	4	28	0
13	3	0	1	2	31	0
14	3	0	2	1	34	0
15	3	0	0	3	37	0
16	2	0	1	1	39	0
17	0	0	1	−1	39	0
18	0	0	0	0	39	0
19	1	0	1	0	40	0
20	2	0	0	2	42	0
21	3	0	0	3	45	0
22	0	0	2	−2	45	0
23	1	0	1	0	46	0
24	1	0	0	1	47	0
25	0	0	0	0	47	0
26	0	0	0	0	47	0
27	0	0	0	0	47	0
28	0	0	1	−1	47	0
29	0	0	1	−1	47	0
30	0	0	0	0	47	0
31	0	0	1	−1	47	0
32	0	0	0	0	47	0
33	0	0	0	0	47	0
34	0	0	0	0	47	0
35	0	0	0	0	47	0

Simulation data for HT17

Day	Infection per day	Dead per day	Recovery per day	Accumulation ill	Accumulation infection	Accumulation dead
0	0	0	0	0	0	0
1	2	0	0	2	2	0
2	2	0	0	2	4	0
3	3	0	0	3	7	0
4	2	0	0	2	9	0
5	6	0	0	6	15	0
6	8	0	0	8	23	0
7	3	0	0	3	26	0
8	7	0	0	7	33	0
9	8	0	0	8	41	0
10	6	0	0	6	47	0
11	5	0	0	5	52	0
12	6	0	0	6	58	0
13	9	0	1	8	67	0
14	2	0	3	−1	69	0
15	7	0	0	7	76	0
16	6	0	1	5	82	0
17	5	0	1	4	87	0
18	2	0	1	1	89	0
19	7	0	0	7	96	0
20	1	0	1	0	97	0
21	6	0	1	5	103	0
22	5	0	1	4	108	0
23	5	0	2	3	113	0
24	2	0	1	1	115	0
25	2	0	1	1	117	0
26	3	0	2	−1	120	0
27	2	0	1	1	122	0
28	2	0	0	2	124	0
29	3	0	1	2	127	0
30	1	0	1	0	128	0
31	1	0	1	0	129	0
32	1	0	0	1	130	0
33	0	0	0	0	130	0
34	0	0	1	−1	130	0
35	0	0	0	0	130	0
36	0	0	2	−2	130	0
37	0	0	0	0	130	0
38	0	0	1	−1	130	0
39	1	0	0	0	131	0
40	0	0	0	0	131	0
41	1	0	0	0	132	0
42	0	0	0	0	132	0

Individual Reflection

Individual reflection 1

-Hussam

As engineers, we must be able to solve problems. And these solutions must be correct, effective and good. We often look for the most optimal way to solve problems. Hence, we have to look at the bigger picture to be able to find the fastest, cheapest and most efficient solution. Engineers should be able to master these skills, because it is part of their daily routine in solving problems and getting involved in different activities and phases of designing products. It is great to be able to invent something that works, but it is important to be able to show that it works and serves the purpose and also that it is tested in a scientific way.

First of all, we need to be able to identify the assumption that this module is based on. After we have these assumption clear, we need to analyze this data and see in what circumstances it can give us different results. When we figure out different solutions to one problem, we need to select the most efficient one according to the resources that we have. And we do that, by comparing the advantages and the disadvantages of each of the solutions.

It is important to consider the validation and verification step, because we need to prove that our system is valid to be able to present a system that provides a solution to the problem.
It might be a bit difficult to go from a problem to model, because we are moving from a small step to generalize the problem and outline the requirement. The process takes longer, because we don’t approach problems directly or don’t go with the instinct solution that comes to mind, but we follow a scientific structure that requires a different mindset and some patience to be able to cope with what is going on and what is to be done.

I have learnt a lot about building a system from only knowing the problem, and I could see the changes during the whole process of designing the system. I can see the difference between the very early stages and the late ones. There is however a lot to be learned in the future about this topic. The more I work with such problems, the better I get.

Individual reflection 2

-Qi Li

In out future career life as a engineer to be able to identify what the customer need and identify what they want is the necessary step. This can help us clearly define our goal and the best way to reach it. During this assignment i realize i have issue towards the need and want which usually not the case for me. Long term study life has make myself focus too much on the theoretical level knowledge. At first of few days i start using multithreading to trying to achieve the maximum performance and forgot we can easily solve this issue with with simple multidimensional array. The mistake on define the wrong model has waste a lot of time. During the verification for each step, I have also realize that I am think stuff too straight and forget how computer thinks. Few of the function took me a while because of the returning result and process number is correct but i think it is wrong because it does not make sense in normal life. This is mostly due to the mental model of each function is not clear before i write it. A piece of paper and pen to model each step before i start coding is probably a good idea for future assignment. There are always possibility to find the shortcut if the mental model is correct before implementation. 50 LOC can always become 5 when different people are doing the same assignment. During the test, verification and validation phase, i feels like i start getting fuzzy for some complicated function. Especially when several loop happend together with different function calling. I should be more careful with those step and patient with them. Because this is the place that most likely result will go wrong. Even a little uncertainty here might lead to chain react and cause huge effect to the final result.