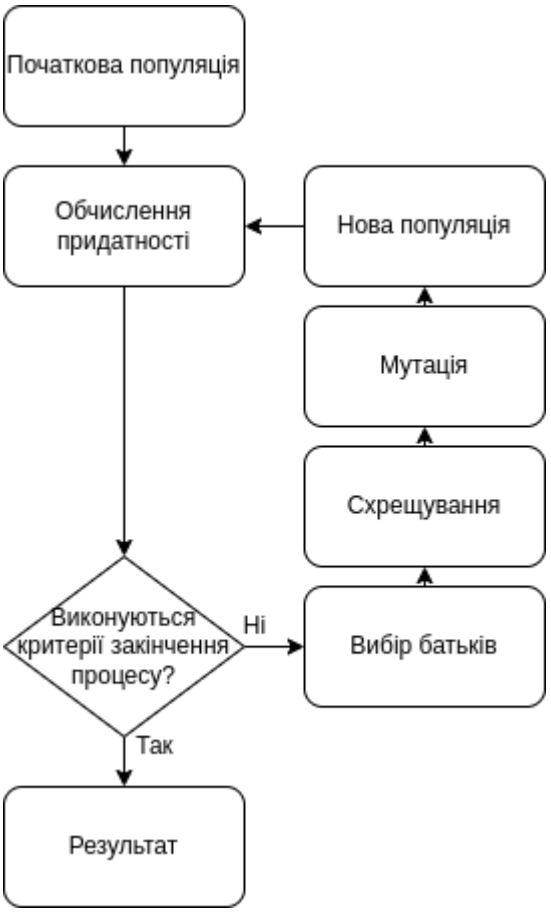
ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

ДІАГРАМА СТРУКТУРИ ІСМЗ

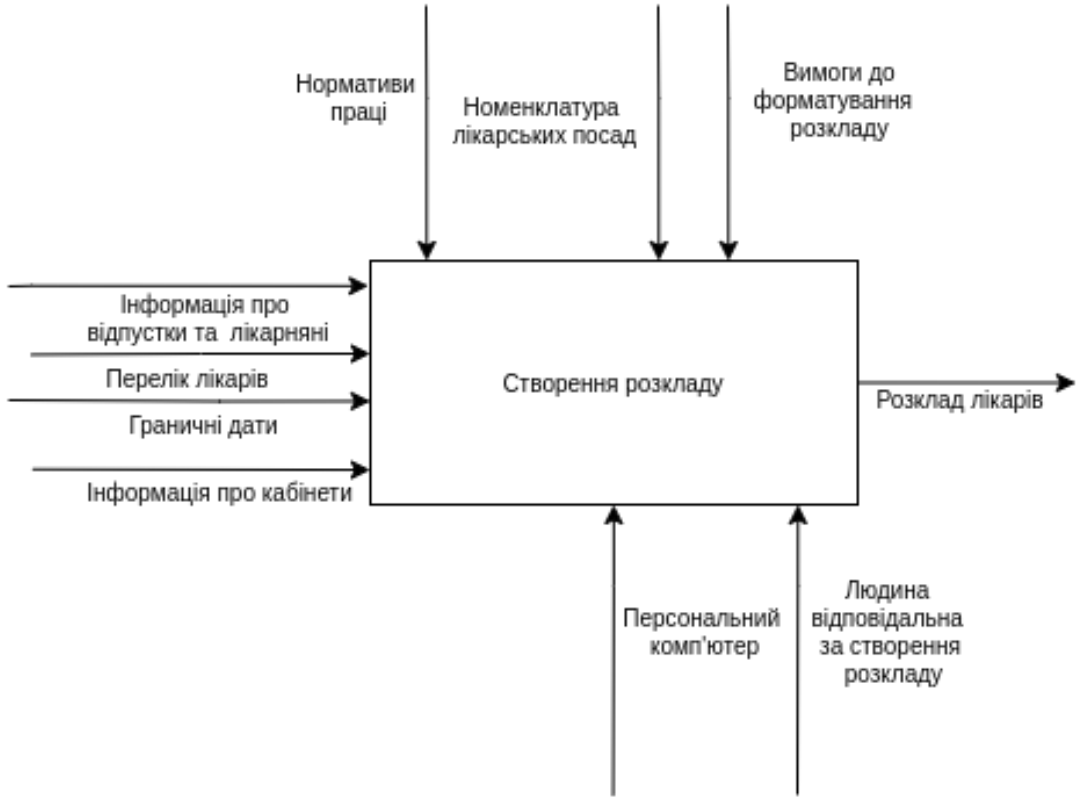
Розроб.	Гончаренко О.І.			Розробка та дослідження компоненту автоматизованого створення розкладу інформаційної системи медичного закладу	
Перевірив	Решетнік В.М.				
Н.Контроль	Решетнік В.М.				
				ІТПм-20-1	Лист 1
Утвердив	Гребеннік І.В.			СТ	Листів 1

ГЮИК. 506190.002 С10



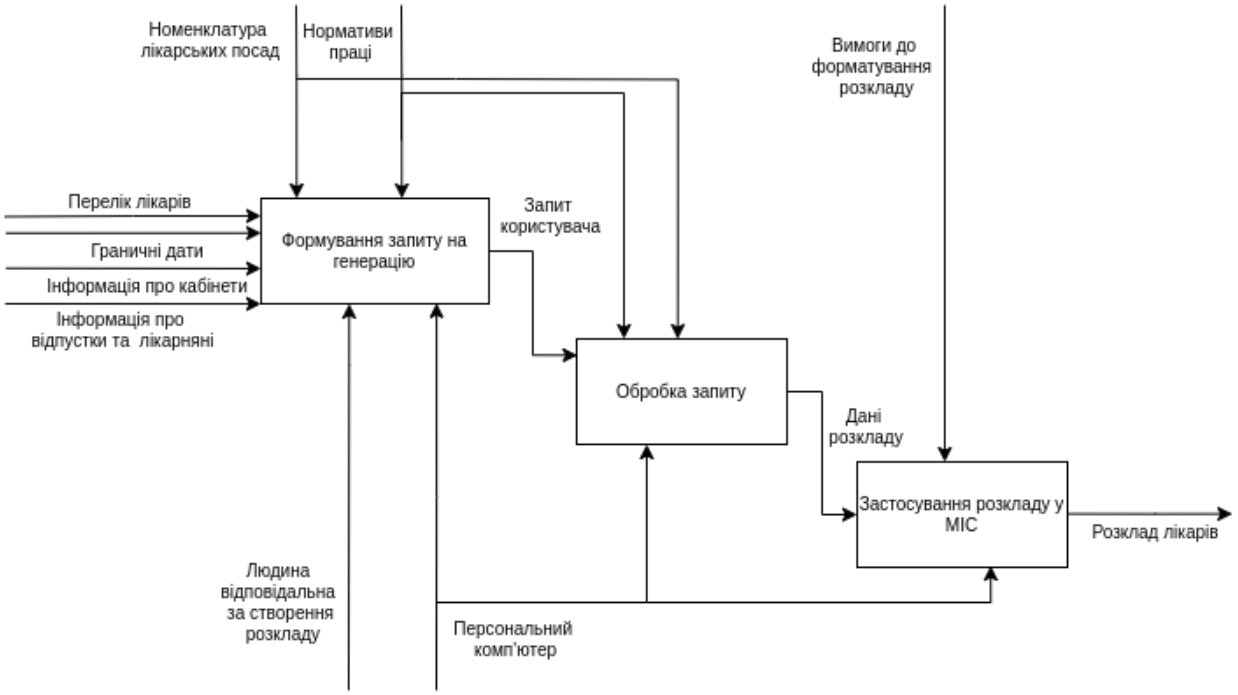
					ГЮИК. 506190.002 С10			
					Розробка та дослідження компоненту автоматизованого стоврення розкладу інформаційної системи медичного закладу	Лит.	Маса	Масштаб
Ізм.	Лист	№ докум.	Підпис	Дата				
Розроб.		Гончаренко О.І.				Лист 1	Листів 1	
Перевірив		Решетнік В.М.						
Т. Контр.								
Реценз.					Схема генетичного алгоритму	ХНУРЕ Кафедра СТ		
Н.Контр.		Решетнік В.М.						
Утверд.		Гребеннік І.В						

ГЮИК. 506190.002 С10



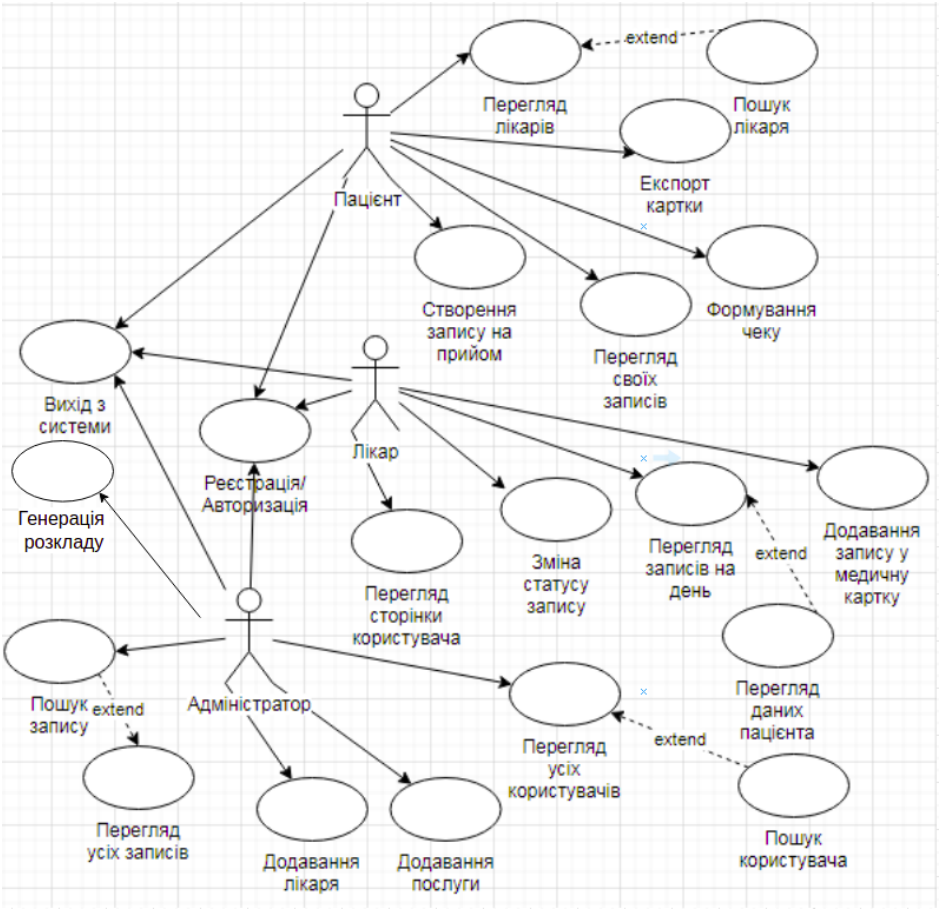
					ГЮИК. 506190.002 С10			
					Розробка та дослідження компоненту автоматизованого стоврення розкладу інформаційної системи медичного закладу	Лит.	Маса	Масштаб
Ізм.	Лист	№ докум.	Підпис	Дата				
Розроб.		Гончаренко О.І.				Лист 1	Листів 1	
Перевірив		Решетнік В.М.						
Т. Контр.								
Реценз.					Контекстна діаграма функціональної моделі компоненту ІСМЗ	ХНУРЕ Кафедра СТ		
Н.Контр.		Решетнік В.М.						
Утверд.		Гребеннік І.В						

ГЮИК. 506190.002 С10



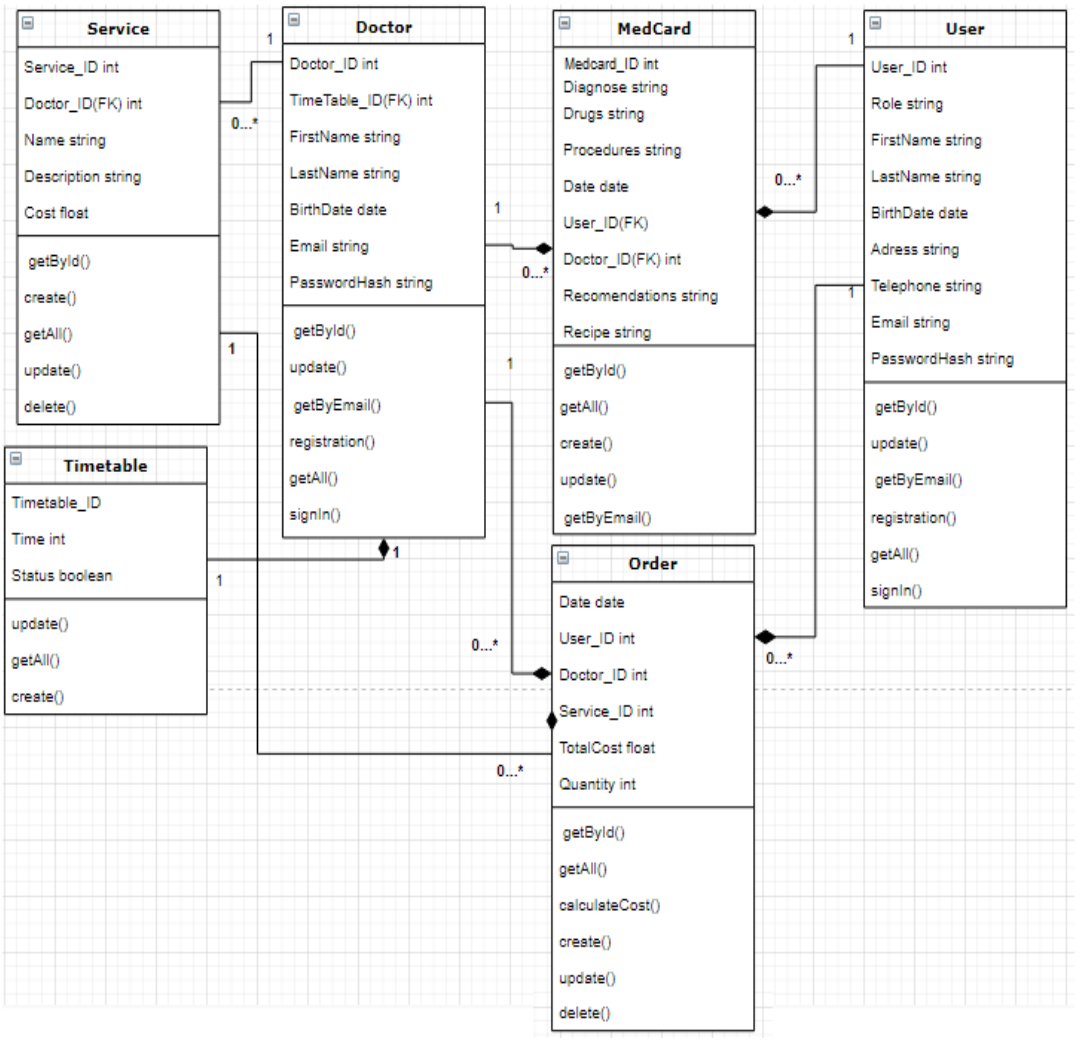
					ГЮИК. 506190.002 С10			
					Розробка та дослідження компоненту автоматизованого стоврення розкладу інформаційної системи медичного закладу	Лит.	Маса	Масштаб
Ізм.	Лист	№ докум.	Підпис	Дата				
Розроб.		Гончаренко О.І.				Лист 1	Листів 1	
Перевірив		Решетнік В.М.						
Т. Контр.								
Реценз.					Діаграма декомпозиції першого рівня	ХНУРЕ Кафедра СТ		
Н.Контр.		Решетнік В.М.						
Утверд.		Гребеннік І.В						

ГЮИК. 506190.002 С10



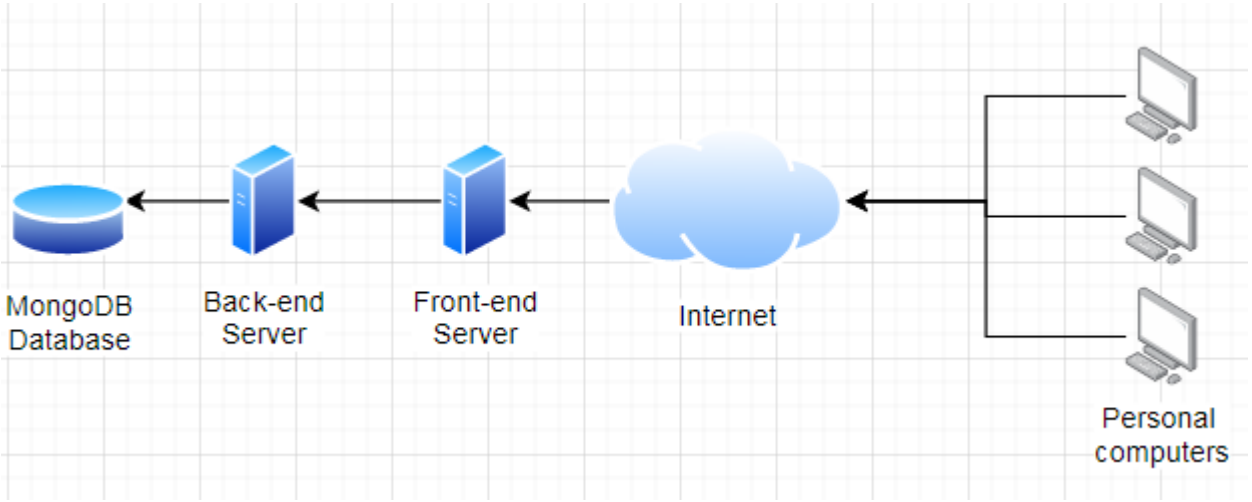
					ГЮИК. 506190.002 С10			
Ізм.	Лист	№ докум.	Підпис	Дата	Розробка та дослідження компоненту автоматизованого стоврення розкладу інформаційної истеми медичного закладу	Лит.	Маса	Масштаб
Розроб.		Гончаренко О.І.						
Перевірив		Решетнік В.М.				Лист 1	Листів 1	
Т. Контр.								
Реценз.					Діаграма варіантів використання компоненту медичної ІС	ХНУРЕ Кафедра СТ		
Н.Контр.		Решетнік В.М.						
Утверд.		Гребеннік І.В						

ГЮИК. 506190.002 С11



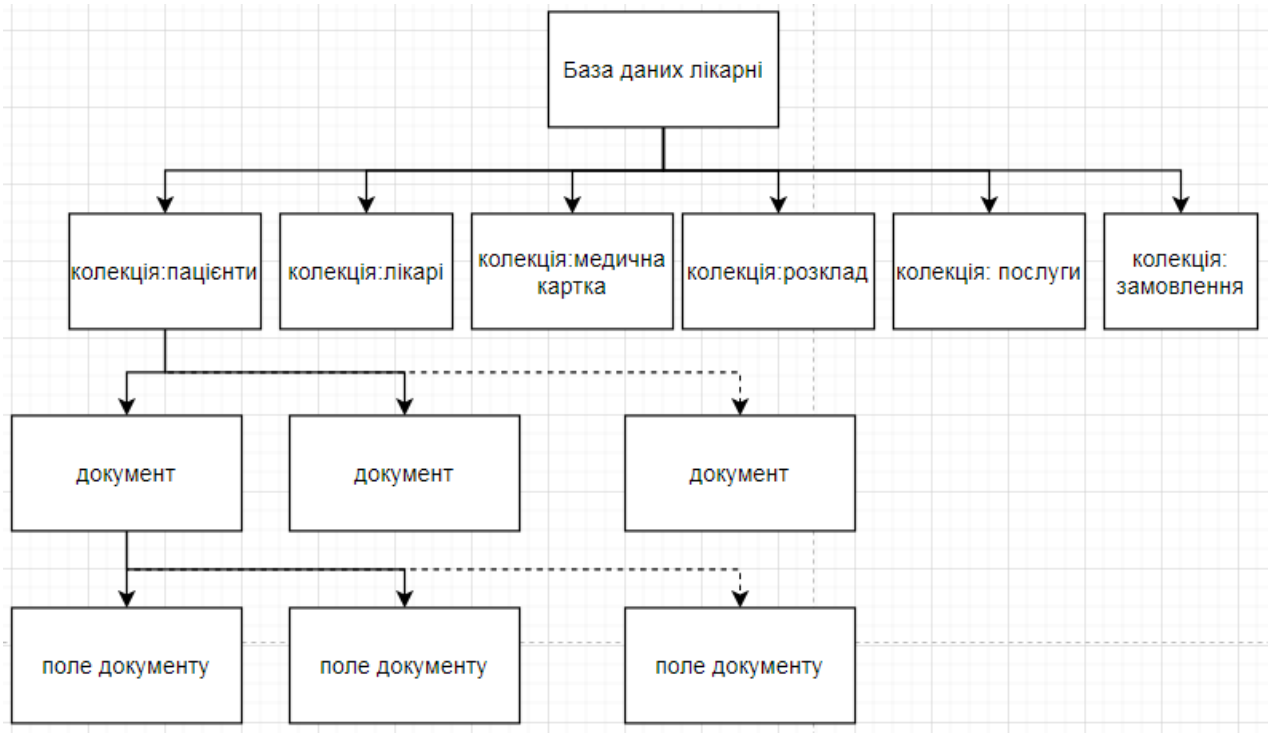
					ГЮИК. 506190.002 С11		
Ізм.	Лист	№ докум.	Підпис	Дата	Розробка та дослідження компоненту автоматизованого стоврення розкладу інформаційної истеми медичного закладу	Лит.	Маса / Масштаб
Розроб.		Гончаренко О.І.					
Перевірив		Решетнік В.М.				Лист 1	Листів 1
Т. Контр.							
Реценз.					Діаграма класів ІСМЗ	ХНУРЕ Кафедра СТ	
Н.Контр.		Решетнік В.М.					
Утверд.		Гребеннік І.В					

СТРУКТУРНА СХЕМА ІСМЗ



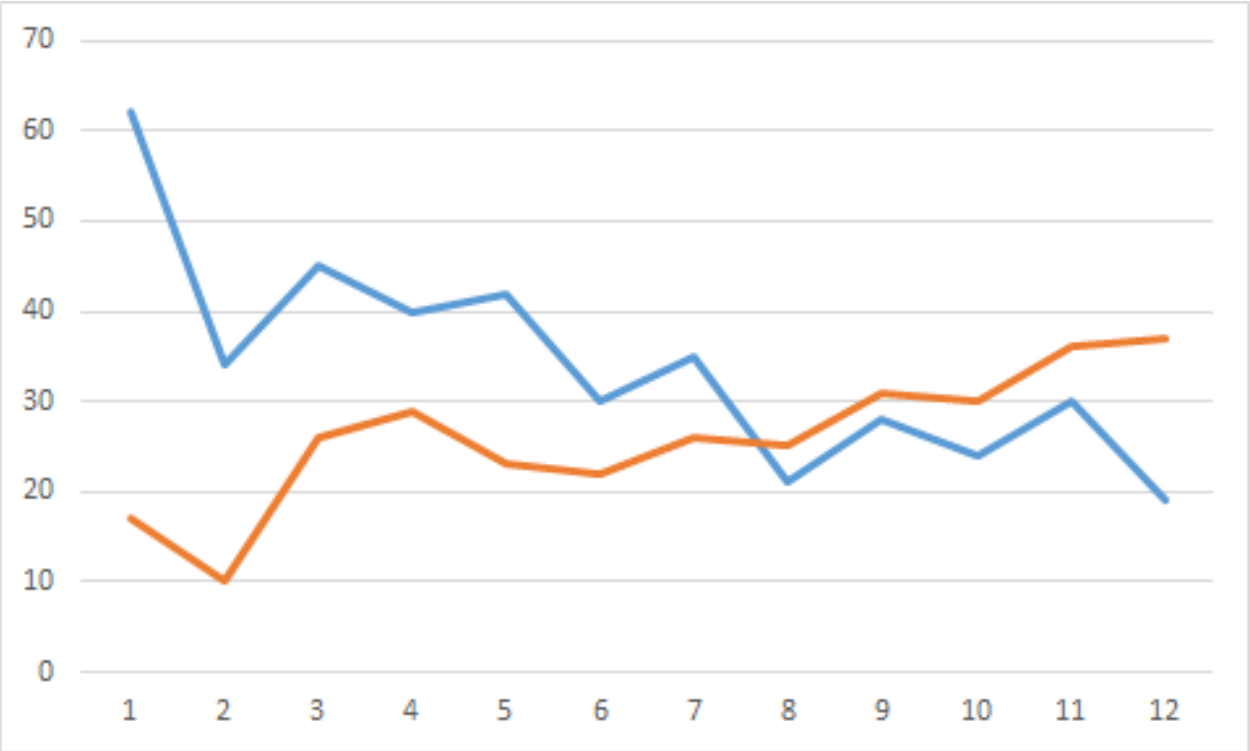
Розроб.	Гончаренко О.І.			Розробка та дослідження компоненту автоматизованого створення розкладу інформаційної системи медичного закладу	
Перевірив	Решетнік В.М.				
Н.Контроль	Решетнік В.М.				
				ІТПм-20-1	Лист 1
Утвердив	Гребеннік І.В.			СТ	Листів 1

СХЕМА СТРУКТУРИ БАЗИ ДАНИХ ІСМЗ



Розроб.	Гончаренко О.І.			Розробка та дослідження компоненту автоматизованого створення розкладу інформаційної системи медичного закладу	
Перевірив	Решетнік В.М.				
Н.Контроль	Решетнік В.М.				
				ІТПм-20-1	Лист 1
Утвердив	Гребеннік І.В.			СТ	Листів 1

ГРАФІК ЗАЛЕЖНОСТІ КІЛЬКОСТІ ЗАПИСІВ ВІД ЧАСУ ПІСЛЯ ІНТЕГРАЦІЇ У СИСТЕМУ МОДУЛЯ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ РОЗКЛАДУ



Розроб.	Гончаренко О.І.			Розробка та дослідження компоненту автоматизованого створення розкладу інформаційної системи медичного закладу	
Перевірив	Решетнік В.М.				
Н.Контроль	Решетнік В.М.				
				ІТПм-20-1	Лист 1
Утвердив	Гребеннік І.В.			СТ	Листів 1

ГРАФІК ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО КОМПОНЕНТУ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ РОЗКЛАДУ

Розроб.	Гончаренко О.І.			Розробка та дослідження компоненту автоматизованого стоврення розкладу інформаційної системи медичного закладу	
Перевірив	Решетнік В.М.				
Н.Контроль	Решетнік В.М.				
				ІТПм-20-1	Лист 1
Утвердив	Гребеннік І.В.			СТ	Листів 1

ДОДАТОК Б

Текст програми

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

«ЗАТВЕРДЖУЮ»

керівник кваліфікаційної роботи

доц. Решетнік В.М.

Розробка та дослідження компоненту автоматизованого стоврення розкладу інформаційної системи медичного закладу

Текст програми

ЛИСТ ЗАТВЕРДЖЕННЯ

ГЮИК.506190.002 12 01 -ЛЗ

УЗГОДЖЕНО:

РОЗРОБИВ:

ст. гр. ІТПм-20-1

Гончаренко О.І.

2021 р

ЗАТВЕРДЖЕНО

ГЮИК.506190.002 12 01 -ЛЗ

Розробка та дослідження компоненту автоматизованого стоврення розкладу інформаційної системи медичного закладу

ПАКЕТ ПРОГРАМ ВИЗНАЧЕННЯ СТРУКТУРИ ОБ’ЄКТА ПРОЕКТУВАННЯ

Текст програми

ГЮИК.506190.002 12 01

Аркушів 7

2021

```
import { CrossoverFunction } from "./CrossoverMethods";
import { MutationFunction } from "./MutationMethod";

export type Offspring<T> = [baby1: T[], baby2: T[]];

export class Chromosome<T> {
    private length: number;
    private genes: T[] = [];
    private fitness = 0;
    private randGene: () => T;
    private randNum: () => number;

    constructor(genes: T[], randomGene: () => T, randomNumber: ()
=> number) {
        this.randGene = randomGene;
        this.randNum = randomNumber;
        this.genes = genes;
        this.length = genes.length;
    }

    public static generate<T>(count: number, randGene: () => T,
randomNumber: () => number) {
        const genes: T[] = [];
        for (let i = 0; i < count; i++) {
            genes.push(randGene());
        }

        return new Chromosome<T>(genes, randGene, randomNumber);
    }

    public getFitness(): number {
        return this.fitness;
    }

    public setFitness(f: number): void {
        this.fitness = f;
    }

    // returns the number of bits which are different = Hamming
distance
    public compare(bob: Chromosome<T>): number {
        let count = 0;

        for (let i = 0; i < Math.min(this.length, bob.length);
i++) {
            count += this.genes[i] !== bob.genes[i] ? 1 : 0;
        }
```

```
  return count;
    }

    public mutateWith(mutationRate = 1 / this.length, mutateFn:
MutationFunction<T>): void {
        mutateFn(this, mutationRate);
    }

    public crossoverWith(bob: Chromosome<T>, crossoverFn:
CrossoverFunction<T>): Offspring<T> {
        return crossoverFn(this, bob);
    }

    public setGenes(genes: T[]): void {
        this.genes = [...genes];
    }

    public getGenes(): T[] {
        return this.genes;
    }

    public copy(bob: Chromosome<T>): void {
        this.genes = bob.genes.slice();
        this.length = bob.length;
        this.randGene = bob.randGene;
        this.fitness = bob.fitness;
    }

    public clone(): Chromosome<T> {
        const clone = Chromosome.generate(
            this.length,
            this.randGene,
            this.randNum
        );
        clone.copy(this);
        return clone;
    }
    public randomGene(): T {
        return this.randGene();
    }

    public randomNumber(): number {
        return this.randNum();
    }
}
import { Chromosome } from "./Chromosome";

export type MutationFunction<T> = (chromo: Chromosome<T>,
```

```
mutationRate: number) => void;

type MutationMethodName = 'flip' | 'swap';

export const mutationMethod: { [K in MutationMethodName]:
MutationFunction<any> } = {
    flip: <T>(chromosome: Chromosome<T>, mutationRate: number):
void => {
        const genes = chromosome.getGenes();
        const length = genes.length;

        for (let i = 0; i < length; i++) {
            if (chromosome.randomNumber() < mutationRate) {
                genes[i] = chromosome.randomGene();
            }
        }
    },

    swap: <T>(chromosome: Chromosome<T>, mutationRate: number):
void => {
        const genes = chromosome.getGenes();
        const length = genes.length;

        for (let i = 0; i < length; i++) {
            if (chromosome.randomNumber() < mutationRate) {
                const j = Math.floor(Math.random() * length);
                [genes[i], genes[j]] = [genes[j], genes[i]];
            }
        }
    }
};
import { Chromosome, Offspring } from "./Chromosome";

export type CrossoverFunction<T> = (alice: Chromosome<T>, bob:
Chromosome<T>) => Offspring<T>;

type CrossoverMethodName = 'singlePoint' | 'twoPoint' | 'uniform'
| 'halfUniform' | 'ordered';

export const crossoverMethod: { [K in CrossoverMethodName]:
CrossoverFunction<any> } = {
    singlePoint: <T>(alice: Chromosome<T>, bob: Chromosome<T>):
Offspring<T> => {
        const p = Math.floor(alice.randomNumber() *
alice.getGenes().length);
        const b1 = [...alice.getGenes().slice(0, p),
```

```
   ...bob.getGenes().slice(p)];
        const b2 = [...bob.getGenes().slice(0, p),
...alice.getGenes().slice(p)];

        return [b1, b2];
    },

    twoPoint: <T>(alice: Chromosome<T>, bob: Chromosome<T>):
Offspring<T> => {
        const b1: T[] = [];
        const b2: T[] = [];

        const alicesGenes = alice.getGenes();
        const bobsGenes = bob.getGenes();
        const aliceChromoLength = alice.getGenes().length;

        let p1 = Math.floor(alice.randomNumber() *
aliceChromoLength);
        let p2 = Math.floor(alice.randomNumber() *
aliceChromoLength);

        if (p1 > p2) {
            [p1, p2] = [p2, p1];
        }

        for (let i = 0; i < aliceChromoLength; i++) {
            b1.push(i < p1 ? alicesGenes[i] : (i < p2 ?
bobsGenes[i] : alicesGenes[i]));
            b2.push(i < p1 ? bobsGenes[i] : (i < p2 ?
alicesGenes[i] : bobsGenes[i]));
        }

        return [b1, b2];
},

    uniform: <T>(alice: Chromosome<T>, bob: Chromosome<T>):
Offspring<T> => {
        const b1: T[] = [];
        const b2: T[] = [];

        const alicesGenes = alice.getGenes();
        const bobsGenes = bob.getGenes();
        const aliceChromoLength = alice.getGenes().length;

        for (let i = 0; i < aliceChromoLength; i++) {
            const swap = alice.randomNumber() < 0.5;
            b1.push(swap ? bobsGenes[i] : alicesGenes[i]);
            b2.push(swap ? alicesGenes[i] : bobsGenes[i]);
```

```
                }

        return [b1, b2];
    },

    halfUniform: <T>(alice: Chromosome<T>, bob: Chromosome<T>):
Offspring<T> => {
        let b1: T[] = [];
        let b2: T[] = [];

        const alicesGenes = alice.getGenes();
        const bobsGenes = bob.getGenes();
        const aliceChromoLength = alice.getGenes().length;

        const diffBits: number[] = [];

        for (let i = 0; i < aliceChromoLength; i++) {
            if (alicesGenes[i] !== bobsGenes[i]) {
                diffBits.push(i);
            }
        }

        const N = diffBits.length;

        b1 = alicesGenes.slice();
        b2 = bobsGenes.slice();

        for (let i = 0; i < N / 2; i++) {
            let idx = Math.floor(alice.randomNumber() *
diffBits.length);
            b1[diffBits[idx]] = bobsGenes[diffBits[idx]];
            b2[diffBits[idx]] = alicesGenes[diffBits[idx]];
            diffBits.splice(idx, 1); }

        return [b1, b2];
    },

    ordered: <T>(alice: Chromosome<T>, bob: Chromosome<T>):
Offspring<T> => {
        const b1: T[] = [];
        const b2: T[] = [];

        const alicesGenes = alice.getGenes();
        const bobsGenes = bob.getGenes();
        const aliceChromoLength = alice.getGenes().length;

        let inf = Math.floor(alice.randomNumber() *
aliceChromoLength);
```

```
        let sup = Math.floor(alice.randomNumber() *
aliceChromoLength);
        let tmp = inf;

        inf = Math.min(inf, sup);
        sup = Math.max(tmp, sup);

        for (let i = inf; i < sup; i++) {
            b1[i] = bobsGenes[i];
            b2[i] = alicesGenes[i];
        }

        for (let i = 0; i < aliceChromoLength; i++) {
            if (b1.indexOf(alicesGenes[i]) === -1) {
                b1[i] = alicesGenes[i];
            } else {
                for (let j = 0; j < aliceChromoLength; j++) {
                    if (b1.indexOf(alicesGenes[j]) === -1) {
                        b1[i] = alicesGenes[j];
                    }
                }
            }

            if (b2.indexOf(bobsGenes[i]) === -1) {
                b2[i] = bobsGenes[i];
            } else {
for (let j = 0; j < bobsGenes.length; j++) {
                    if (b2.indexOf(bobsGenes[j]) === -1) {
                        b2[i] = bobsGenes[j];
                    }
                }
            }

        }

        return [b1, b2];
    }
};
```