**СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“ ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА**


# КУРСОВ ПРОЕКТ
# ПО СИСТЕМИ, ОСНОВАНИ НА ЗНАНИЯ

**Тема:**

Система за билкова аптека,

която по данни за определени билкови съставки предлага лекарства, които да могат да се приготвят от тези съставки

**Изготвили:**

Кристина Красимирова Савова, 71912

Евелина Момчилова Попова, 71973

София, януари 2022г.

## 1. Задание – кратко описание на решаваната задача

Реализирайте система за билкова аптека, която по данни за определени билкови съставки да предлага лекарства, които да могат да се приготвят от тези съставки. С други думи, потребителят ще въведе списък от билки и тяхното количество в грамове, а програмата ще представи възможни рецепти.

За тази цел ще разполагаме с два csv файла:

Първият трябва съдържа информация за съвкупност от билки, като всяка една ще има свои **име, цена и срок на годност (година)**.

Вторият ще съдържа **рецепти за лекарства**, за всяка от която ще се представят **всички билкови съставки**, необходими за даденото лекарство; заедно с това ще се изчислява и цената на самото лекарство (цената му ще се образува на базата на цените на самите билки от рецептата му, умножени по тяхното **необходимо за създаването на лекарството** количество), както и колко порции могат да се съставят от количеството билки, които той въвежда. Идеята на програмата е да проверява дали бихме могли да съставим конкретно лекарство според билките, с които разполагаме. И ако е така, ще бъдем наясно колко порции от дадено лекарство могат да се направят от въведените от нас съставки, както и каква е цената на една порция от него.

**Важно е да се отбележи**, че в нашата програма цената, годината на годност и количеството не могат да бъдат по-малки или равни на 0. А името на съставка или рецепта не може да има стойност null. Правим валидации както в методите в класовете, така и при изграждането на потребителския интерфейс.

## 2. Описание на предложения алгоритъм за решаване на задачата (чрез псевдокод). Описание на структурата и произхода на използваните данни

### 2.1. Използвани алгоритми

Можем да считаме, че нашата задача се базира на небезизвестния генетичен алгоритъм. Генетичният алгоритъм представлява вариант на стохастично търсене в лъч, при който новите състояния се генерират чрез комбиниране на двойки родителски състояния вместо чрез модифициране на текущото състояние. Той се базира на естествените принципи от биологичната еволюция. Основната му идея е да се наподоби в опростен вид процеса на естествен отбор

(селекция), с цел да се открие възможно най-добър алгоритъм за решаване на дадена задача. Алгоритъмът започва работа с множество (популация) от n случайно генерирани състояния.

В нашата задача можем да забележим много фактори, наподобяващи логиката на генетичния алгоритъм.

**Рецепта А** съдържа конкретна съвкупност от съставки и техните предварително зададени количества.

**Рецепта В** съдържа друга, различна съвкупност от съставки и техните предварително зададени количества.

Нека рецептите А и В играят ролята на родителски състояния, а гените ще бъдат името и количеството на дадените съставки, включени в рецептите.

**Рецепта С** съдържа част от **Рецепта А** и част от **Рецепта В**. С други думи, нека изберем **една точка на кръстосване**. Нашият низ от началото си до точката на кръстосване е копие на началната част на единия родител (**Рецепта А**), останалата му част е копие на съответната част на втория родител (**Рецепта В**).

*Пример*: Вземайки предвид само съставките, ако направим кръстосване в единична точка при рецептите за световъртеж (Dizziness) и за нарушаване на съня (Sleep Disorder), можем да получим рецептата за успокоително (Tranquiliser).

| | | | | |
|---|---|---|---|---|
| **Dizziness** | balm | valerian | hawthorn | |
| **Sleep disorder** | linden | hop cones | mint | hawthorn |
| **Tranquiliser** | balm | valerian | mint | hawthorn |

**Рецепта D** съдържа почти същите съставки като тези от **Рецепта С**, но количествата на две от съставките на двете рецепти са различни. Тук можем да говорим и за **мутация**.

*Пример*: Рецептите за успокоително (Tranquiliser) и световъртеж (Dizziness) и разменените количества на съставките **валериан (valerian) и маточина (balm)**.

| | |
|---|---|
| Tranquilizer | mint (0.03) hawthorn (0.12) valerian (0.1) balm (0.04) |
| Dizziness | balm (0.1) hawthorn (0.05) valerian (0.04) |

### 2.2. Описание на метода hasEnoughQuantity

Целта на метода е да приема като аргумент списък със съставки, въведени от потребителя, и връща булева стойност. Така може да се провери дали има достатъчно количество от необходимите съставки

за създаването на рецептата. Най-напред се създава списък от съставките, които са въведени от потребителя, като се вземат само тези от тях, съдържащи се в конкретната рецепта. След това проверяваме дали броя на съставките в този списък съответства на броя съставки в рецептата. Ако броят се различава, това означава, че нямаме всички съставки, следователно не можем да направим тази рецепта и методът връща false. Ако броят е равен, тогава сравняваме имената на въведените съставки с имената на съставките в рецептата. Ако количеството на съставките, които въвежда потребителят, е по-голямо или равно на вече зададеното, то в такъв случай имаме достатъчно съставки за изготвянето на рецептата и методът връща стойност true.

**Псевдокод:**

function hasEnoughQuantity(Argument userComponents) returns boolean
declare array list of components and add all components which are from userComponents and are contained in prescription's components
if the size of the array list is different from the size of the list of prescription's components
then return false
loop for each component of the array list
loop for each component of the prescription
if their names are equal then
if the unit of user component is less than the unit of prescription component
then return false
end loop
end loop
return true
end function

### 2.3. Описание на метода calculateMaxPortions

Методът приема като аргумент списък със съставки, въведени от потребителя, и изчислява максималния брой порции, които могат да се получат от количеството на тези съставки (задължително цяло число).

Първо създаваме списък от цели числа. Всяко число е резултат от делението на въведеното от потребителя количество и зададеното количество за дадена съставка в рецептата. Т.е. тези числа съответстват на това колко пъти можем да използваме съставките за тази рецепта. Накрая взимаме най-малкия елемент от списъка и той показва максималния брой порции, които могат да бъдат съставени.

**Псевдокод:**

```
function calculateMaxPortions(Argument input)
    declare empty array list of integers
    loop for each Component element of input
    loop for each Component element of prescription's components
    if their names are equal
    declare userUnit and set it to the unit of the input component
    declare prescriptionUnit and set it to the unit of the prescription component
    add the integer division of userUnit divided by prescriptionComponent to
    the array list
    end loop
    end loop
    set the serving to the minimum element of the array list
end function
```

### 2.4. Описание на метода getPrescriptions

Този метод връща списък с всички рецепти, които могат да се получат от списъка с подадени от потребителя медицински съставки.

**Псевдокод:**

```
function getPrescriptions(Argument input) returns array list of prescriptions
    declare empty array list of prescriptions
    loop for each component from the database
    loop for each component of input
    if their names are equal then
    set the years of the user component to the years of the component from the
    database
    set the price of the user component to the price of the component from the
    database
    end loop
    end loop
    loop for each component from the database
    if call function hasEnoughQuantity(input) for the prescription is true then
    call function calculateMaxPortion(input) to set the servings
    add the prescription to the array list
    end loop
    return the array list of prescriptions
end function
```

### 2.5. Описание на метода parseComponents

Това е помощен метод за четене на рецептите, който прочита съставките от файла с рецепти, премахвайки скобите и всеки интервал, така че да се прочете само името на съставката и неговото количество.

**Псевдокод:**

```
function parseComponents(String input) returns an array list of components
        declare an array list result which will save all components from the file
        declare a component <- Component which will save the current component from the file

        declare and initialize an array with strings -> parsed which removes the ‘)’ and the
        space between the unit and the name of the component in the file
        declare a String tmp which takes the last symbol from each line and removes it -> [‘)’]
        for each String s of parsed
        declare and initialize a String array only with the name and the unit of a component
        declare a flag found which shows if a component is found
        for each component p of components
        if the name of a component is in the list of all components
        set flag = true
        initialize the component <- new Component(name of the component, get its
        years ears from our list of components, get its price from the list of components,
        get its quantity from the file with prescriptions
        break
        if the component is not found
        throw an exception to show that exactly this component was not found
        add the component to the result
        return result
end function
```

## 2.6. Описание на метода readPrescriptions

Методът приема име на файл и прочита рецептите, които са въведени в него.

**Псевдокод:**

```
function readPrescriptions(String filename)
declare variable openFile <- BufferedReader
try
        initialize openFile with the name of file <- FileReader
        read the first line
        declare and initialize a variable line which reads the text of each line
        while the text in line is not null
                declare and initialize a String array lineData with the text without ‘,’
                if the lineData has length > 0
```

```
                add the prescription in an array list which is declared as a private data
                member to the class {read the name of a prescription, call the
                parseComponents() function to read the components}
catch each exception if there is any
end function
```

### 2.7. Описание на метода readComponents

Този метод прочита информацията за всички въведени във файла съставки.

**Псевдокод:**

```
function readComponents(String filename)
    declare variable openFile <- BufferedReader
    try
        initialize openFile with the name of file <- FileReader
        read the first line
        declare and initialize a variable line which reads the text of each line
        while the text in line is not null
            declare and initialize a String array lineData with the text without ‘,’
            if the lineData has length > 0
                add the component in an array list which is declared as a private data
                member to the class {read the name of a component,
                read its calories, read its price, read its unit}
    catch each exception if there is any
end function
```

### 2.8. Описание на метода btnAddOnAction

Той е част от реализацията на потребителския интерфейс. Целта му е да добавя всяка избрана от потребителя съставка от падащото меню в текстова област, предназначена за въведените съставки.

**Псевдокод:**

```
function btnAddOnAction(event)
    if the user does not enter a quantity
        return an alert of type Alert.AlertType.ERROR which shows a message to
        make the user to enter a quantity
    if the user enters an invalid name of a component
        return an alert of type Alert.AlertType.ERROR which shows a message to
        make the user to enter a valid name or to choose a valid component from our list
        in the combo box
    else if the user enters a valid name of a component
        for each component from our array list of components which is a private data member
        of the class Controller
            if the list contains this component
                add it to the text area
```

```
        set the text of the text field for the quantity to null so the user can enter another value
end function
```

### 2.9. Описание на метода btnGenerateOnAction

Той е част от реализацията на потребителския интерфейс. Идеята е в текстова област да се генерират всички рецепти, които могат да се направят от въведените от потребителя съставки.

**Псевдокод:**

```
function btnGenerateOnAction(event)
        if the user does not enter any component
                return an alert of type Alert.AlertType.ERROR which shows a message to
                make the user to enter a component
        else if there are some components
                declare a variable input of type array list which will save only the name of the
                component and the entered by the user quantity
                declare and initialize a variable components of type array of String which get the
                text from the text area with components, dividing the components with new line
                for each String p of components
                        declare an array of Strings subcomponents which contains the information
                        without the comma and the space after it
                        add all components to the input
        set text to the text area for prescriptions using the method getPrescriptions <- KnowledgeBase
        if there are not any valid prescriptions
                return an alert of type Alert.AlertType.ERROR which shows a message to make the user to
        enter new components
end function
```

### 2.10. Описание на метода btnRemoveLastOnAction

Той е част от реализацията на потребителския интерфейс. Идеята е да се премахва последно въведената съставка от потребителя, в случай че е сгрешил при избора на съставка.

**Псевдокод:**

```
function btnRemoveLastOnAction(event)
        if the user does not enter any component in the text area for components
                return an alert of type Alert.AlertType.ERROR which shows a message that
                this operation can not be done
        declare variable result <- StringBuilder which takes the last entered component
        declare and initialize a String array tmp which get the text from the text area for
        components, divided by a new line
        for each value i to the component before the last one
                append all components except the last one
        set text to the text area of components but without the last component
end function
```

### 2.11. Въвеждане на съставки в падащото меню (JavaFX)

```
// read the files consisting of components and prescriptions
knowledgeBase = new KnowledgeBase( componentsFile: "components.csv",  prescriptionsFile: "prescriptions.csv");
componentArrayList = knowledgeBase.getComponents(); // get all components from the knowledge base

// add all names of components in the combo box, so the user can pick from them
for (Component component : componentArrayList) {
    cboChooseComponent.getItems().add(component.getName());
}

// additional formatting of the text areas and the combo box
txaComponents.setWrapText(Boolean.TRUE);
txaPrescriptions.setWrapText(Boolean.TRUE);
cboChooseComponent.setEditable(Boolean.TRUE);
```

### 2.12. Описание на данни

**Component.java**

private String name - В този низ записваме името на съставката

private Integer years - Тук записваме годината на годност на съставката

private Double price - Тук записваме цената на съставката

private Double unit - Тук записваме грамовете на съставката

**Prescription.java**

private String name - Тук в този низ записваме името на рецептата

private Integer serving - Тук записваме колко най-много порции могат да бъдат направени от съответните съставки

private Double price - Тук записваме цената на едно лекарство за една порция

private Integer years - Тук записваме годината на годност на лекарството

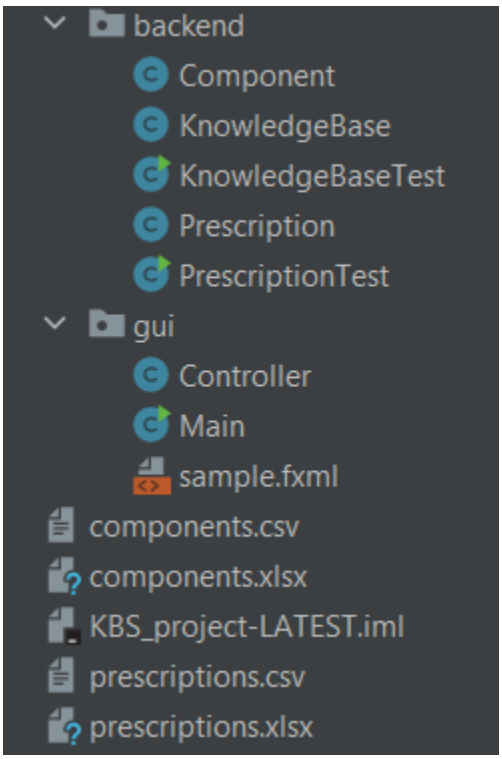
private ArrayList<Component> components - Тук всяка рецепта се състои от списък със съставки

**KnowledgeBase.java**

private ArrayList<Component> components - Това е списък от всички съставки

private ArrayList<Prescription>prescriptions - Това е списък от всички рецепти

## 3. Програмна реализация. Примерни резултати от работата на създаденото приложение




Програмата ни е структурирана в две папки (два пакета). В първата папка (**backend)** се съдържа функционалността на системата, като също така там са описани всички създадени класове, необходими за реализирането на системата, основана на правила. А втората папка (**gui)** съответно представлява логиката зад самия потребителски интерфейс.

Заедно с тези два пакета програмата ни включва и двата гореспоменати **.xlsx** файла, в които въвеждаме рецептите и необходимите за тях съставки, след което ги преобразуваме в **.csv** файлове, за да можем да четем данните от тях.

Започваме със създаването на клас **Component**, който описва една съставка със следните характеристики – име, цена, срок на годност (година) и количество (в грамове - “Unit”). В този клас имаме конструктор, който създава съставка, както по зададени от потребителя стойности, така и със стойности по подразбиране, които ние предварително сме задали. Освен това имаме методи за въвеждане и валидация на стойностите, както и методи за достъпване на член-данните на класа. Накрая в метода toString() извеждаме пълната информация за съставката.

Следва създаването на още един ключов клас - **Prescription**, който описва една медицинска рецепта, съдържаща данни за името на рецептата, порциите, крайния срок на годност (във формата на година) и списък от съставките, необходими за създаването ѝ. Аналогично на класа **Component**, имаме конструктор, който създава медицинска рецепта по име и списък от билкови съставки. Методите за валидация задават годината на изтичане на годността и цената за всяка рецепта, като за второто се събират цените на съставките, умножени по необходимия грамаж на всяка една. Счетохме,

че за формиране на една разумна за такива лекарства цена е добре да зададем и базова цена от 2.50лв, включена към горната формула.

### 3.1. PrescriptionTest.java

За да проверим функционалността на класа Prescription, създаваме тестов клас PrescriptionTest. В него въвеждаме съставки, с които знаем, че можем да създадем определен брой порции от дадена рецепта. В случая въвеждаме лавандула и лайка, които са за една порция от рецептата за лекарство против акне:

```
ArrayList<Component> components =  new ArrayList<>(){
    {
        add(new Component( name: "lavender",  years: null,  price: null,  unit: 3.0));
        add(new Component( name: "camomile",  years: null,  price: null,  unit: 30.0));
    }
};
Prescription prescription = new Prescription( name: "Acne", new ArrayList<>(){
    {
        add(new Component( name: "lavender",  years: 2024,  price: null,  unit: 1.0));
        add(new Component( name: "camomile",  years: 2022,  price: null,  unit: 10.0));
    }
});
```

```
Prescription name: Acne, Maximum portions: 1, Price per portion: 2.70 lv., Shelf-Life in Years: 2022
Components:
 Component: lavender, Shelf-Life in Years: 2024 Component: camomile, Shelf-Life in Years: 2022

Is there enough quantity of herbs to make the medicine? Yes
Max Medicine Portions: 3
Process finished with exit code 0
```

### 3.2. KnowledgeBase.java и KnowledgeBaseTest.java

След това създаваме класа **KnowledgeBase**, съдържащ списък с билкови съставки и списък с рецепти. В конструктора на класа се осъществява четенето на информацията от .csv файловете и така всъщност ги добавяме в списъците. Освен методите за четене имаме и метод getPrescriptions, който

приема списък с билки и връща като резултат списък с рецептите, които биха могли да се получат от тях (в случай че има такива). Ако не се срещат такива, връща като резултат празния списък.

За да тестваме нашата логика, накрая създаваме и класа **KnowledgeBaseTest**.

Нека първо проверим метода getPresciptions с примерни билкови съставки, чиито срокове на годност и цена избираме да приемат стойност null, защото се генерират на базата на въведените от нас година и цена в таблицата с всички съставки:

```
ArrayList<Component> components = new ArrayList<>() {
    {
        add(new Component( name: "lavender", years: null, price: null, unit: 600.0));
        add(new Component( name: "camomile", years: null, price: null, unit: 80.0));
        add(new Component( name: "mint", years: null, price: null, unit: 40.0));
        add(new Component( name: "hawthorn", years: null, price: null, unit: 90.0));
        add(new Component( name: "valerian", years: null, price: null, unit: 30.0));
        add(new Component( name: "balm", years: null, price: null, unit: 450.0));
        add(new Component( name: "tutsan", years: null, price: null, unit: 100.0));
        add(new Component( name: "thyme", years: null, price: null, unit: 400.0));
        add(new Component( name: "basil", years: null, price: null, unit: 65.0));
        add(new Component( name: "agrimony", years: null, price: null, unit: 200.0));
        add(new Component( name: "yarrow", years: null, price: null, unit: 85.0));
        add(new Component( name: "milkvetch", years: null, price: null, unit: 100.0));
        add(new Component( name: "linden", years: null, price: null, unit: 850.0));
        add(new Component( name: "hop cones", years: null, price: null, unit: 260.0));
        add(new Component( name: "white comfrey", years: null, price: null, unit: 170.0));
        add(new Component( name: "coltsfoot", years: null, price: null, unit: 55.0));
        add(new Component( name: "juniper", years: null, price: null, unit: 190.0));
        add(new Component( name: "mustard seed", years: null, price: null, unit: 5.0));
        add(new Component( name: "raspberry", years: null, price: null, unit: 15.0));
        add(new Component( name: "marigold", years: null, price: null, unit: 15.0));
        add(new Component( name: "oregano", years: null, price: null, unit: 520.0));
    }
};
```

Така установяваме, че с тези дадени билки можем да съставим общо 9 рецепти за лекарства:

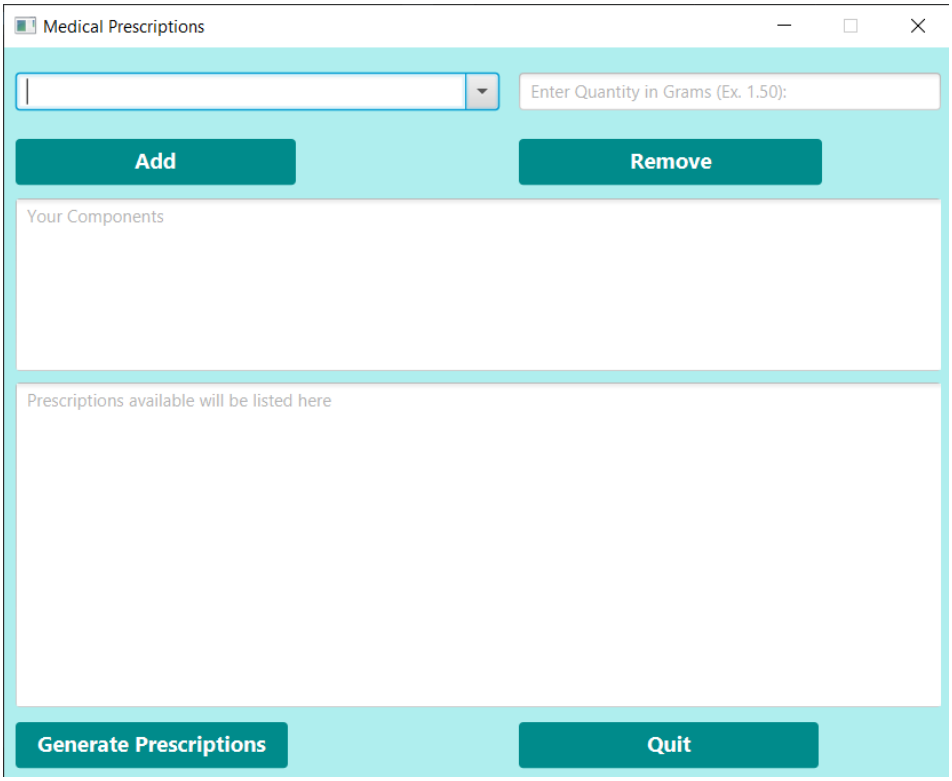
```
C:\Users\krisi\.jdks\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2020.3.2\lib\idea_rt.jar=63628:C:\Program Files\JetBrains\IntelliJ IDEA 2020
[Prescription name: Acne, Maximum portions: 1600, Price per portion: 2.63 lv., Shelf-Life in Years: 2022
Components:
 Component: lavender, Shelf-Life in Years: 2027 Component: camomile, Shelf-Life in Years: 2024
, Prescription name: Tranquilizer, Maximum portions: 300, Price per portion: 2.72 lv., Shelf-Life in Years: 2022
Components:
 Component: mint, Shelf-Life in Years: 2023 Component: hawthorn, Shelf-Life in Years: 2023 Component: valerian, Shelf-Life in Years: 2028 Component: balm, Shelf-Life in Years: 20
, Prescription name: Varicose veins, Maximum portions: 650, Price per portion: 2.81 lv., Shelf-Life in Years: 2022
Components:
 Component: tutsan, Shelf-Life in Years: 2023 Component: thyme, Shelf-Life in Years: 2025 Component: balm, Shelf-Life in Years: 2025 Component: basil, Shelf-Life in Years: 2027
, Prescription name: Ovaritis, Maximum portions: 425, Price per portion: 2.85 lv., Shelf-Life in Years: 2022
Components:
 Component: agrimony, Shelf-Life in Years: 2024 Component: yarrow, Shelf-Life in Years: 2027 Component: milkvetch, Shelf-Life in Years: 2026
, Prescription name: Sleep disorder, Maximum portions: 300, Price per portion: 3.47 lv., Shelf-Life in Years: 2022
Components:
 Component: hawthorn, Shelf-Life in Years: 2023 Component: mint, Shelf-Life in Years: 2023 Component: linden, Shelf-Life in Years: 2025 Component: hop cones, Shelf-Life in Years:
, Prescription name: Dizziness, Maximum portions: 300, Price per portion: 2.65 lv., Shelf-Life in Years: 2022
Components:
 Component: balm, Shelf-Life in Years: 2025 Component: hawthorn, Shelf-Life in Years: 2023 Component: valerian, Shelf-Life in Years: 2028
, Prescription name: Brain stimulation, Maximum portions: 50, Price per portion: 2.77 lv., Shelf-Life in Years: 2022
Components:
 Component: white comfrey, Shelf-Life in Years: 2024 Component: coltsfoot, Shelf-Life in Years: 2026 Component: juniper, Shelf-Life in Years: 2023 Component: mustard seed, Shelf-
, Prescription name: Tonsillitis, Maximum portions: 150, Price per portion: 2.87 lv., Shelf-Life in Years: 2022
Components:
 Component: raspberry, Shelf-Life in Years: 2022 Component: marigold, Shelf-Life in Years: 2024 Component: oregano, Shelf-Life in Years: 2025 Component: camomile, Shelf-Life in Y
, Prescription name: Sunburn, Maximum portions: 1000, Price per portion: 2.60 lv., Shelf-Life in Years: 2022
Components:
 Component: camomile, Shelf-Life in Years: 2024 Component: marigold, Shelf-Life in Years: 2024 Component: yarrow, Shelf-Life in Years: 2027 Component: tutsan, Shelf-Life in Years
]
```

Файлът съдържа и проверка на правилното отваряне и прочитане на components.csv и prescriptions.csv. Тъй като списъкът е дълъг и детайлен, ето и част от резултата в следния скрийншот:

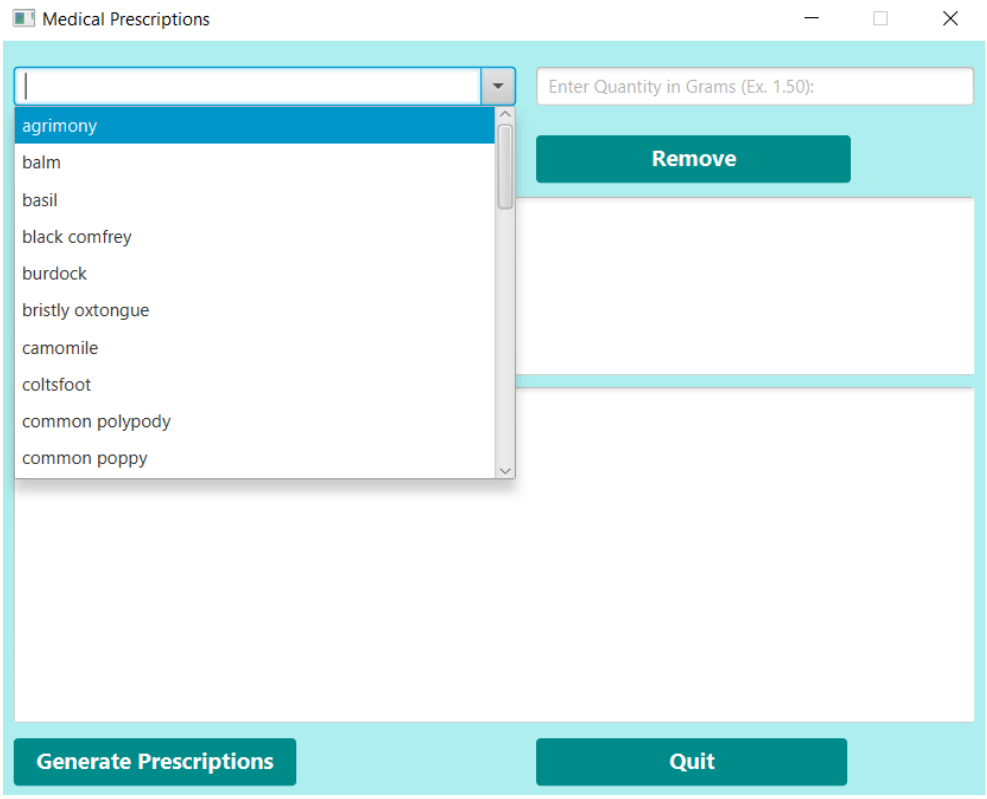
```
Components: [ Component: agrimony, Shelf-Life in Years: 2024,  Component: balm, Shelf-Life in Years: 2025,  Component: basil, Shelf-Life in Years: 2027,  Component: black comfre
Prescriptions: [Prescription name: Acne, Maximum portions: 1, Price per portion: 2.63 lv., Shelf-Life in Years: 2022
Components:
 Component: lavender, Shelf-Life in Years: 2027 Component: camomile, Shelf-Life in Years: 2024
, Prescription name: Tranquilizer, Maximum portions: 1, Price per portion: 2.72 lv., Shelf-Life in Years: 2022
Components:
 Component: mint, Shelf-Life in Years: 2023 Component: hawthorn, Shelf-Life in Years: 2023 Component: valerian, Shelf-Life in Years: 2028 Component: balm, Shelf-Life in Years: 2
, Prescription name: Varicose veins, Maximum portions: 1, Price per portion: 2.81 lv., Shelf-Life in Years: 2022
Components:
 Component: tutsan, Shelf-Life in Years: 2023 Component: thyme, Shelf-Life in Years: 2025 Component: balm, Shelf-Life in Years: 2025 Component: basil, Shelf-Life in Years: 2027
, Prescription name: Ovaritis, Maximum portions: 1, Price per portion: 2.85 lv., Shelf-Life in Years: 2022
Components:
 Component: agrimony, Shelf-Life in Years: 2024 Component: yarrow, Shelf-Life in Years: 2027 Component: milkvetch, Shelf-Life in Years: 2026
, Prescription name: Sleep disorder, Maximum portions: 1, Price per portion: 3.47 lv., Shelf-Life in Years: 2022
Components:
 Component: hawthorn, Shelf-Life in Years: 2023 Component: mint, Shelf-Life in Years: 2023 Component: linden, Shelf-Life in Years: 2025 Component: hop cones, Shelf-Life in Years
, Prescription name: Dizziness, Maximum portions: 1, Price per portion: 2.65 lv., Shelf-Life in Years: 2022
Components:
 Component: balm, Shelf-Life in Years: 2025 Component: hawthorn, Shelf-Life in Years: 2023 Component: valerian, Shelf-Life in Years: 2028
, Prescription name: Brain stimulation, Maximum portions: 1, Price per portion: 2.77 lv., Shelf-Life in Years: 2022
Components:
 Component: white comfrey, Shelf-Life in Years: 2024 Component: coltsfoot, Shelf-Life in Years: 2026 Component: juniper, Shelf-Life in Years: 2023 Component: mustard seed, Shelf
, Prescription name: Tonsillitis, Maximum portions: 1, Price per portion: 2.87 lv., Shelf-Life in Years: 2022
Components:
 Component: raspberry, Shelf-Life in Years: 2022 Component: marigold, Shelf-Life in Years: 2024 Component: oregano, Shelf-Life in Years: 2025 Component: camomile, Shelf-Life in
, Prescription name: Sunburn, Maximum portions: 1, Price per portion: 2.60 lv., Shelf-Life in Years: 2022
Components:
 Component: camomile, Shelf-Life in Years: 2024 Component: marigold, Shelf-Life in Years: 2024 Component: yarrow, Shelf-Life in Years: 2027 Component: tutsan, Shelf-Life in Year
, Prescription name: Freezing, Maximum portions: 1, Price per portion: 3.15 lv., Shelf-Life in Years: 2022
```

### 3.3. Потребителски интерфейс (GUI)

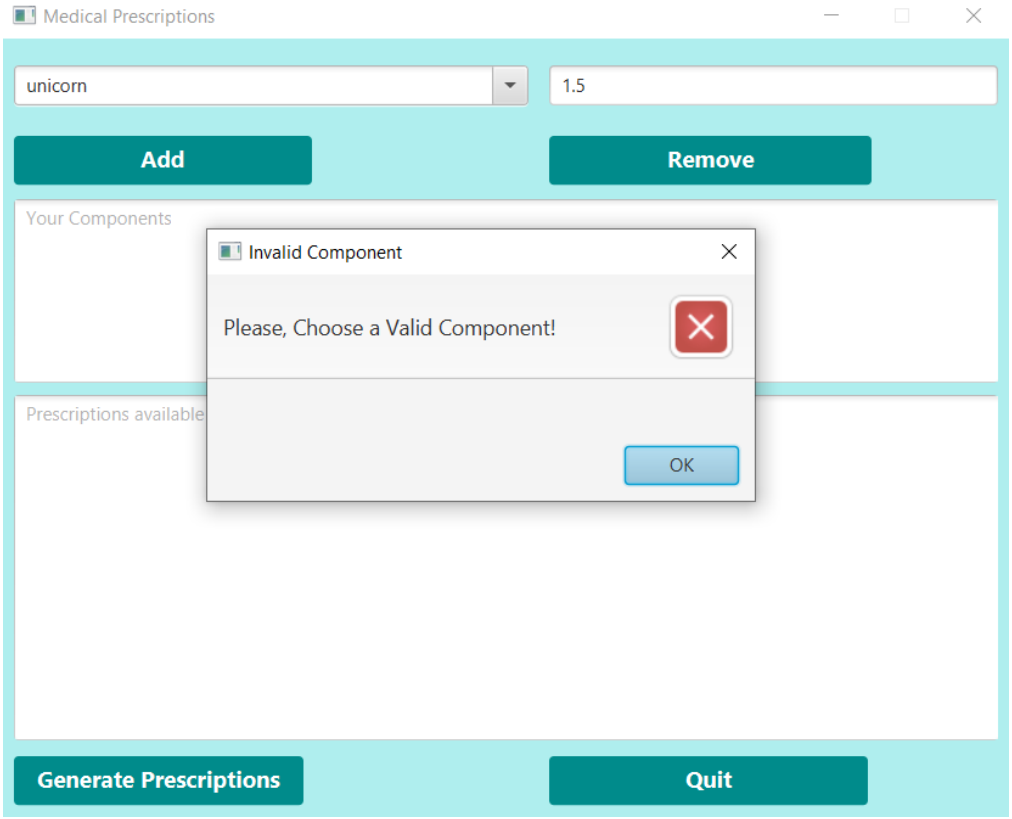
За да представим функционалността на нашата система, основана на правила, решихме да направим потребителски интерфейс с помощта на JavaFX, който изглежда по следния начин:



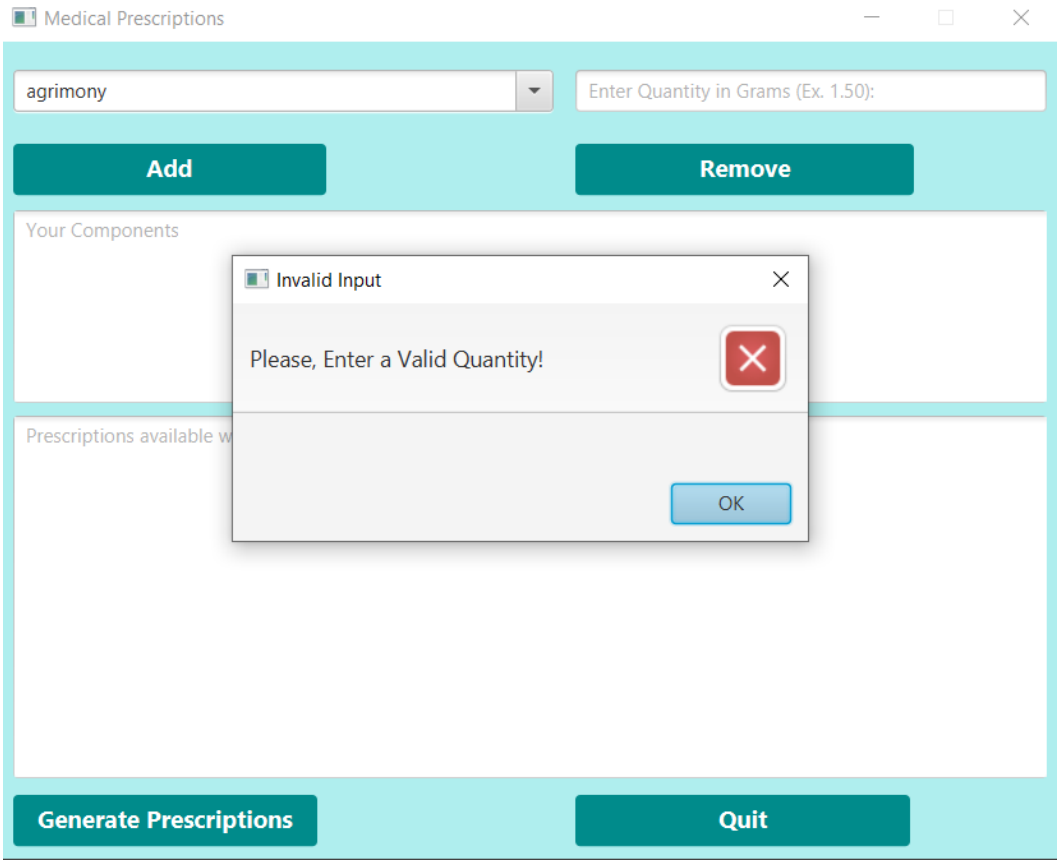

Оттук потребителят има възможност да избира съставки от падащото меню, в което сме ги добавили. За негова леснота сме ги подредили по азбучен ред:



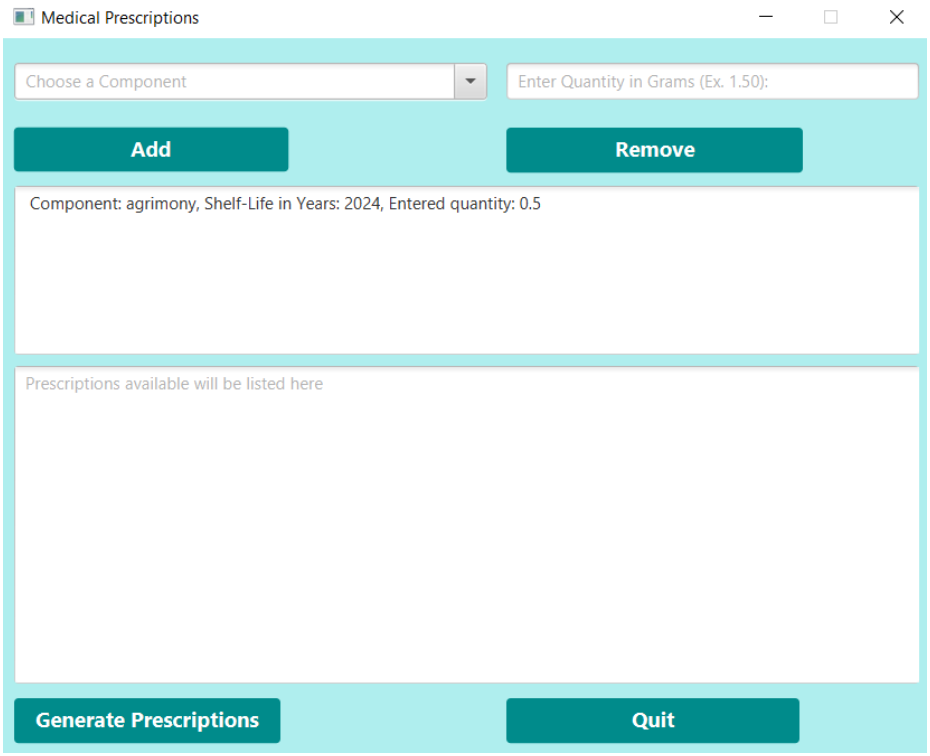

Освен да избира от менюто, той има право и да въвежда сам имена на съставки в самата търсачка. В случай че въведе невалидно име (такова, което е написано грешно, такова, което не се среща в нашия списък), получава следното съобщение:



След като е избрал валидна съставка, е задължително да въведе конкретно количество. В противен случай при натискане на бутона “Generate Prescriptions” се извежда съобщение за грешка, което информира потребителя какво трябва да направи:

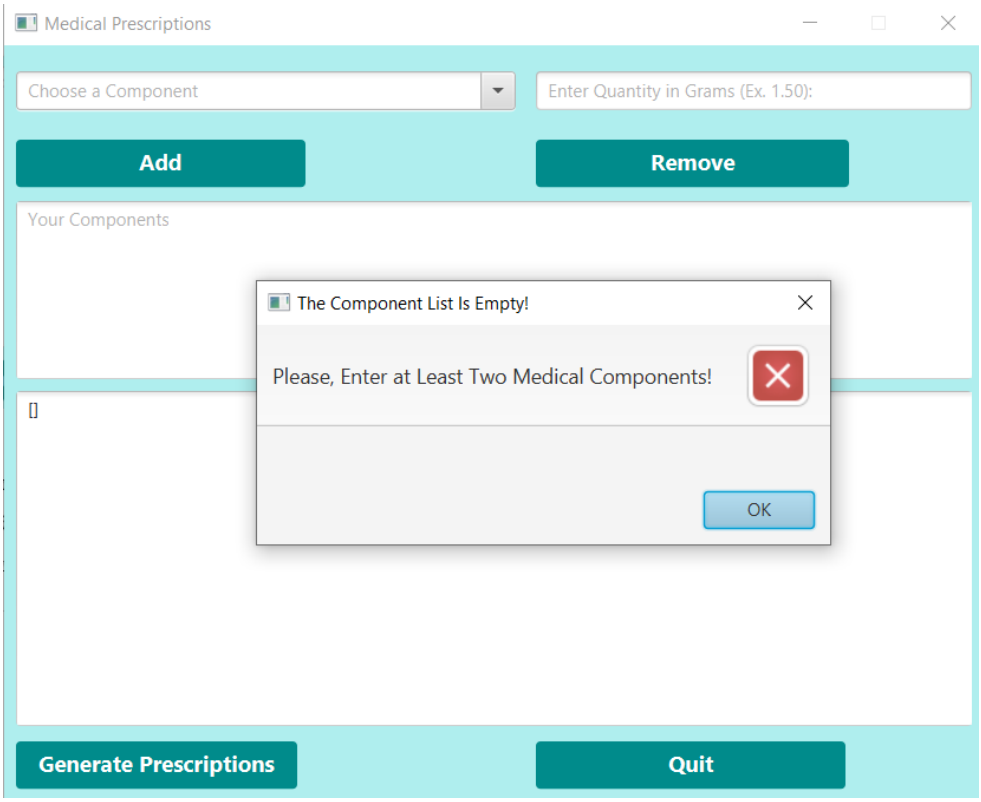



След като потребителят вече е въвел валидно име на съставка и дадено количество, след като се натисне бутонът “Add”, съставката се добавя в текстовото поле под бутона с пълната информация за нея (както се вижда, част от информацията идва и от components таблицата):

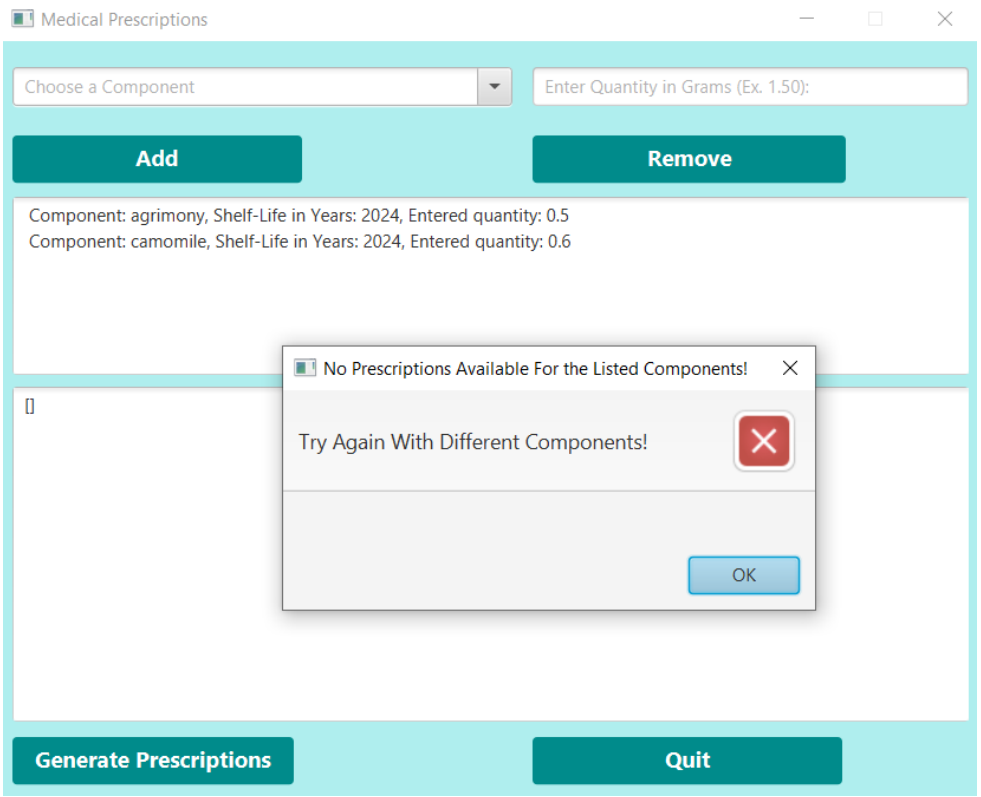



По този начин, колкото и съставки да избере потребителят, всички те ще се добавят в текстовата област. С помощта на бутона “Remove” потребителят може да премахва последната добавена съставка и тогава той се изтрива от списъка с въведени съставки.

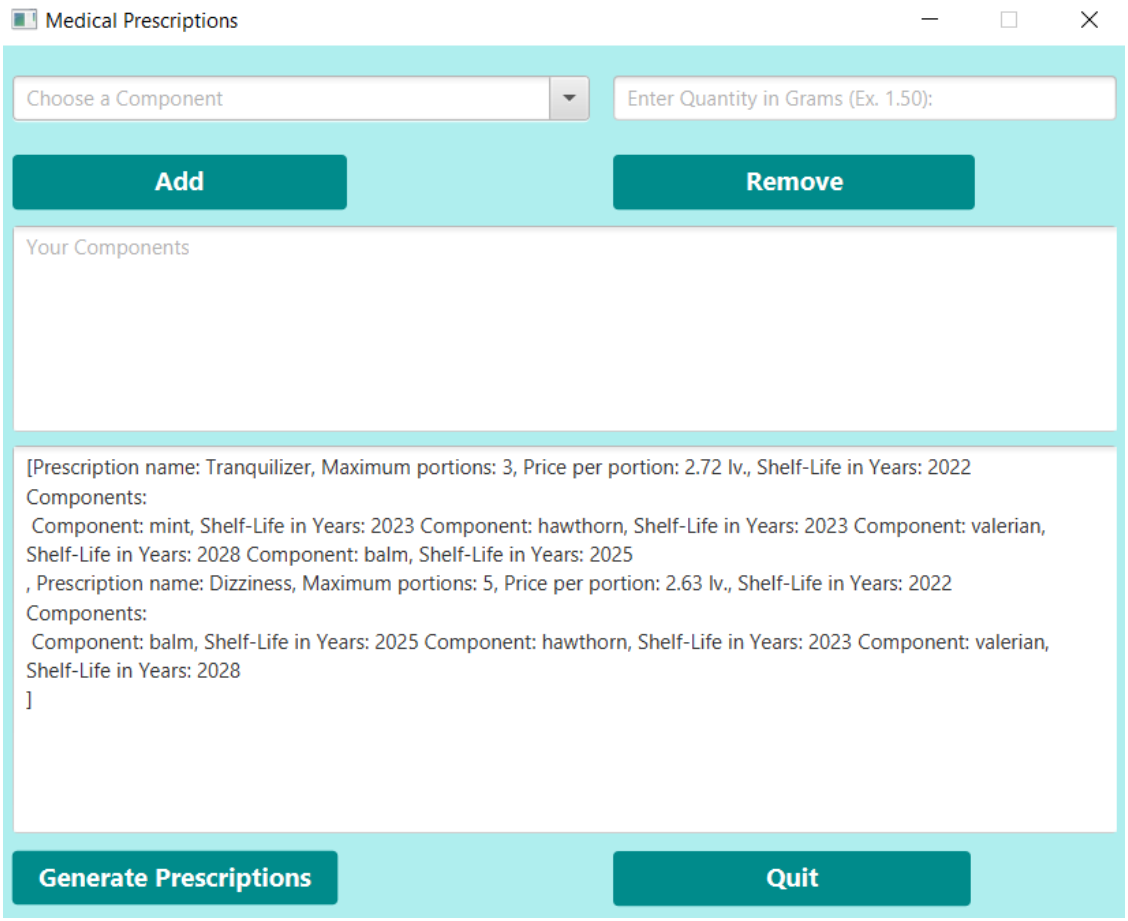
Бутонът “Generate Prescriptions” генерира рецептите, които могат да се получат на базата на избраните от потребителя съставки. В случай че полето с съставки е празно, се извежда съобщение за грешка:




Ако пък няма подходяща рецепта за тези съставки, връщаме празния списък и отново съобщение за грешка:



Иначе, ако всички съставки могат да се включат в дадена рецепта, данните за нея се извеждат в текстовата област под съставките, като списъкът от съставки се нулира, за да се позволи въвеждането на нови съставки:




## 4. Възможни подобрения на съставка

В бъдеще е възможно проектът да се развие като се добавят следните функционалности:

- След като потребителят въведе желаните от него съставки и програмата покаже съответната рецепта, е нужно да затворим потребителския Java интерфейс и да стартираме програмата наново, за да продължим с нова рецепта. Това считаме, че би могло да бъде неудобно за потребителя, и в бъдеще би било добре да се добави бутон “Clear”, който да изчиства историята на показаните лекарства и да подпомогне за по-лесното използване на интерфейса.

- В проекта всяка съставка има срок на годност. Считаме, че една много добра функционалност е да се изчислява срокът на годност на лекарство, направено от различни съставки. Т.е. да се

проверят всички години на годност и да се направи алгоритъм, който смята каква ще бъде година на годност на крайния продукт.

- Друга интересна функционалност е филтриране на съставките чрез добавяне на буква в търсачката. Т.е. когато натиснеш определена буква от клавиатурата да се покажат всички съставки, започващи със съответната буква, за да може потребителят да се ориентира по-лесно и по-бързо.

## 5. Източници


- https://herbilka.life/билкови-рецепти-за-лечение-на-различн/
- Проф. Мария Нишева-Павлова, Презентации от лекции по СОЗ
- Гл. ас. д-р Кристина Арнаудова, доц. д-р Йоаннис Патиас, Презентации от упражнения по СОЗ