MINISTRY OF EDUCATION AND SCIENCE OF THE KYRGYZ REPUBLIC

KYRGYZ-GERMAN INSTITUTE OF APPLIED INFORMATICS

**Coursework**

**Programming languages 2**

**On the subject: «Clinic»**

Supervisor: A. Kibekbaev(Senior Lecturer)**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

English Language Adviser: A. Zheenalieva (Senior Lecturer)**\_\_\_\_\_\_\_**

А. Asanova (Senior Lecturer)**\_\_\_\_\_\_\_\_**

Completed by WIN-1-21: Imashev D.T.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Akylbek kyzy G.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Kartanbaev C.Z.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Bishkek 2022**

**Content**

[**Introduction** 3](#_Toc103685362)

[**Charter laws and rights** 4](#_Toc103685363)

[**Roles in the team** 5](#_Toc103685364)

[**Progress stage and planning minutes** 6](#_Toc103685365)

[**Software Requirements Specification** 7](#_Toc103685366)

[**Used tools** 7](#_Toc103685367)

[**Architectural representation** 8](#_Toc103685368)

[**Log in to the system** 9](#_Toc103685369)

[**Account** 11](#_Toc103685370)

[**Patient part** 11](#_Toc103685371)

[**Medassistant part** 18](#_Toc103685372)

[**Doctor part** 24](#_Toc103685373)

[**Main doctor part** 34](#_Toc103685374)

[**Conclusion** 49](#_Toc103685375)

[**References** 50](#_Toc103685376)

# **Introduction**

**Abstract:** One of the most important conditions for ensuring the effective functioning of any organization is the availability of a developed information system. An information system is a system that implements automated data collection, processing and manipulation and includes technical means of data processing, software and maintenance personnel.

**Subjects:** A clinic is a health facility that is primarily focused on the care of outpatients. Clinics can be privately operated or publicly managed and founded. They typically cover the primary care needs of populations in local communities, in contrast to larger hospitals which offer more specialized treatments and admit inpatients for overnight stays.

**Goals:** The purpose of our course work is to create a possible application for the clinic, with the help of which all users of this application: patient, nurse, doctor, chief physician would have the opportunity to receive the information they need and patients could receive treatment.

**Objectives:** The automated information system "AIS Clinic" is designed to summarize information about a medical institution, the number of doctors, the number of nurses, the number of patients for each doctor, the ability to view the medical history, the ability to search for a patient, etc. The users of the program are doctors, nurses, patients, the chief physician of the medical center. Everyone will have their own access to the system by login and password. Depending on the account category of a particular user, the corresponding menu opens.

"AIS Clinic" also provides the following functions:

* Log in to your account;
* The ability to choose from the menu based on the type of person's account;
* The ability to search by the name of the patient, if it is the attending physician or the head physician;
* The ability to search by the name of the attending physician or nurse, if it is the head doctor;
* The ability to add or remove certain medical histories to a particular patient;
* The ability to add or remove a certain doctor or nurse - through the account of the chief physician.

**Relevance:** Due to the large number of diseases caused by the recent coronavirus pandemic, there was a need for applications thanks to which it was possible to get your diagnosis without leaving home, as well as receive appropriate treatment.

**Target group:** Our target group is the employees of the clinic as well as those who are treated in this clinic.

# **Charter laws and rights**

* To receive, on request, limited medical leave for health reasons or for another valid reason.
* The right to privacy and the non-disclosure of private information, provided that it is not prejudicial to the work.
* Voluntarily physically attend our meetings in University.
* Political, religious agitation within our team is prohibited.
* It is prohibited to leave the team without raising the issue for general discussion.

# **Roles in the team**

**Imashev Diyas:**

* Authorization;
* Main doctor's part of the code;
* Medassistant's part of the code.

**Akylbek kyzy Gulzina:**

* Patient's part of the code;
* Documentation.

**Kartanbaev Chingiz:**

* Doctor’s part of the code;
* Documentation.

# **Progress stage and planning minutes**

Table 1. Phase

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Task mode** | **Task name** | **Duration** | **Beginning** | **End** |
| 1 | Completed | Getting a task | 1 day | 3 February | 3 February |
| 2 | Completed | Understanding  the requirements | 2 day | 3 February | 5 February |
| 3 | Completed | Separation of tasks | 1 day | 5 February | 5 February |
| 4 | Completed | Creating a login menu | 3day | 15 March | 18 March |
| 5 | Completed | Account creation | 32 day | 20 March | 22 April |
| 6 | Completed | Excel connection | 7 day | 15 April | 22 April |
| 7 | Completed | Documentation | 7 day | 8 May |  |
| 8 | Completed | Presentation | 4 day |  |  |

# **Software Requirements Specification**

This program was created using IntelliJ IDEA Community Edition and Java. The database is in Excel and Windows 10. It should have no compatibility issues with older versions of Microsoft Visual Studio, IntelliJ IDEA Community Edition and Java.

# **Used tools**

***Program language:*** *Java*

***IDE:*** *Intellij idea*

***Database:*** *Excel file, Txt file*

***GUI:*** *Terminal*

# **Architectural representation**

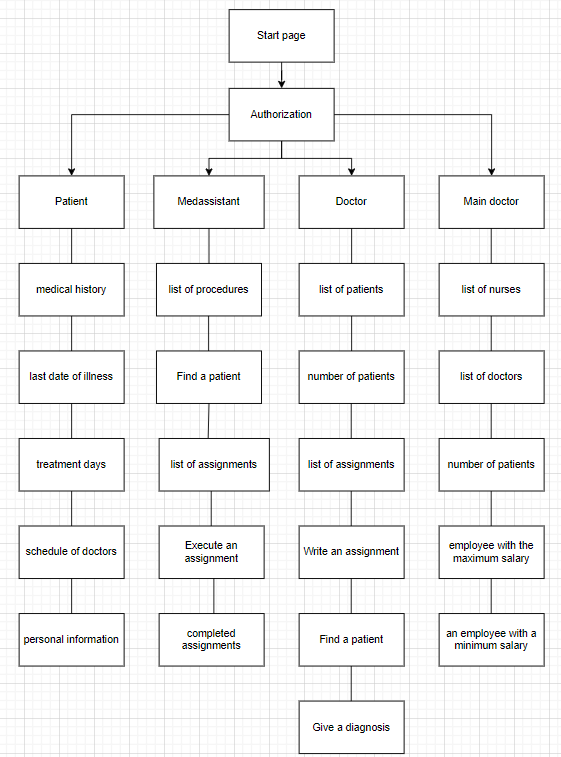
The program starts with authorization, where the user enters his login and password. And depending on the data entered, the user logs into the account.

The patient's account can view their medical history, date of illness, days of treatment, personal information, doctors' schedules.

The med assistant’s account can view a list of procedures for patients, a list of assignments, can also mark the tasks that the med assistant has completed, and view a list of completed assignments.

The doctor's account can view the list of patients, the number of patients, the list of assignments for med assistant, completed assignments. Also, the doctor can write an assignment for med assistant and give a diagnosis to the patient.

The Main doctor account can view the list of patients, the list of attending doctors, the number of patients, the employee with the minimum and maximum salary.



Pic. 1. Architectural representation

# **Log in to the system**

The beginning of the program begins with the selection of an account.

public static String authorization() throws IOException {  
 System.*out*.print("""  
   
 Enter the number of your account type  
   
 1 - assistant  
 2 - doctor  
 3 - main doctor  
 4 - patient  
   
 5 - shut down the program: \s""");  
  
 Scanner input\_type\_of\_account = new Scanner(System.*in*);  
 String type\_of\_account = input\_type\_of\_account.nextLine();  
 switch (type\_of\_account) {  
 case "1" -> System.*out*.println(*assistant\_checking*());  
 case "2" -> System.*out*.println(*doctor\_checking*());  
 case "3" -> System.*out*.println(main\_doctor\_checking());  
 case "4" -> System.*out*.println(patient\_checking());  
 case "5" -> System.*out*.println("Goodbye! See you later.");  
 default -> {  
 System.*out*.println("""  
   
 Sorry, but we didn't find this type of account, please repeat.  
 """);  
 System.*out*.println(*authorization*());  
 }  
 }  
 return "";  
 }  
}

Pic. 2. Account Selection

Next, you register directly in the accounts themselves. Below is an example of registering for a doctor's account.

public static String doctor\_checking() throws IOException {  
  
 String[] words;  
 FileReader fr = new FileReader("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\doctor's\_data.txt");  
 BufferedReader br = new BufferedReader(fr);  
 String s;  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("""  
 Login: \s""");  
 String login = input.nextLine();  
 System.*out*.print("Password: ");  
 String password = input.nextLine();  
 String data = login + password;  
 int count = 0;  
 while ((s = br.readLine()) != null) {  
 words = s.split(" ");  
 for (String word : words) {  
 if (word.equals(data)) {  
 count++;}}}  
 if (count == 1) {  
 System.*out*.print("""  
  
 Authorization was successful  
 """);  
 System.*out*.println(*doctor*());}

Pic. 3. Logging to the doctor's account

If the user enters the wrong username and password, an error message will appear “You have incorrectly entered your username and/or password, please repeat.”, and will prompt you to select the next step.

else {  
 System.*out*.println("""  
  
 You have incorrectly entered your username and/or password, please repeat.  
 """);  
  
 System.*out*.print("""  
 Choose the next move  
  
 1 - try again  
  
 2 - get back to the main page  
 3 - shut down the program: \s""");  
 Scanner input\_next\_move\_after\_fail = new Scanner(System.*in*);  
 String next\_move\_after\_fail = input\_next\_move\_after\_fail.nextLine();  
 switch (next\_move\_after\_fail) {  
 case "1" -> System.*out*.println(*doctor\_checking*());  
 case "2" -> System.*out*.println(*authorization*());  
 case "3" -> System.*out*.println("Goodbye! See you later.");  
 default -> System.*out*.println("" + *doctor\_checking*());  
 }  
 }  
 return "";  
}

Pic. 4. Incorrect login and password entry

# **Account**

# **Patient part**

After logging in to the patient account, the user will receive the message "Authorization was successful".

String[] words;  
FileReader fr = new FileReader("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patient's\_data.txt");  
BufferedReader br = new BufferedReader(fr);  
String s;  
Scanner input = new Scanner(System.*in*);  
System.*out*.print("""  
   
 Login: \s""");  
String login = input.nextLine();  
System.*out*.print("Password: ");  
String password = input.nextLine();  
String data = login + password;  
int count = 0;  
while ((s = br.readLine()) != null) {  
 words = s.split(" ");  
 for (String word : words) {  
 if (word.equals(data)) {  
 count++;}}}  
  
if (count == 1) {  
 System.*out*.println("""  
   
 Authorization was successful  
 """);  
 System.*out*.println(*patient*());}  
else {  
 System.*out*.println("""  
   
 You have incorrectly entered your username and/or password  
 """);  
  
 System.*out*.print("""  
 Choose the next move  
   
 1 - try again  
   
 2 - get back to the main page  
 3 - shut down the program: \s""");  
 Scanner input\_next\_move\_after\_fail = new Scanner(System.*in*);  
 String next\_move\_after\_fail = input\_next\_move\_after\_fail.nextLine();  
 switch (next\_move\_after\_fail) {  
 case "1" -> System.*out*.println(*patient\_checking*());  
 case "2" -> System.*out*.println(*authorization*());  
 case "3" -> System.*out*.println("Goodbye! See you later.");  
 default -> System.*out*.println("" + *patient\_checking*());  
 }  
}  
return "";

Pic. 5. Authorization in the patient’s account

Next, the user is asked to select one of the menus.

System.*out*.print("""  
 Enter the number of menu  
   
 1 - doctors' schedule  
 2 - last date of illness  
 3 - medical history  
 4 - personal information  
 5 - treatment period  
   
 6 - get back to main page  
 7 - shut down the program: \s""");  
Scanner input\_number\_of\_menu = new Scanner(System.*in*);  
String number\_of\_menu = input\_number\_of\_menu.nextLine();

Pic. 6. Patient menu

1. Doctors’ schedule

Entering this menu, the user must enter the first and last name of the doctor, after which the program will display the schedule of the entered doctor.

switch (number\_of\_menu) {  
  
 case "1" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\doctors.xlsx";  
  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.print("Doctor's first name and last name: ");  
 String name = input.nextLine();  
  
 InputStream is = new FileInputStream(path);  
  
 XSSFWorkbook xwb = new XSSFWorkbook(is);  
 XSSFSheet sheet = xwb.getSheet("Doctors");  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 if (cell.getCellType() == CellType.*STRING*) {  
  
 if (cell.getRichStringCellValue().getString().trim().equals(name)) {  
  
 String message = cell + "'s schedule";  
  
 if (message.length() > 13) {  
  
 System.*out*.println(message);  
  
 String path2 = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\d\_schedule.xlsx";  
  
 InputStream is2 = new FileInputStream(path2);  
  
 XSSFWorkbook xwb2 = new XSSFWorkbook(is2);  
 XSSFSheet personal\_sheet = xwb2.getSheet(name);  
  
 FormulaEvaluator formulaEvaluator = xwb2.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row2 : personal\_sheet) {  
  
 for (Cell cell2 : row2) {  
  
 switch (formulaEvaluator.evaluateInCell(cell2).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell2.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell2.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();}}  
  
 else {System.*out*.println(*next\_step*());}  
 }  
 }  
 }  
 }  
 System.*out*.println(*next\_step*());}

Pic. 7. Doctors’ schedule

1. Last date of illness

By entering this menu, the user will receive his last date of illness.

case "2" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patients.xlsx";  
  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.print("Your first name and last name: ");  
 String name = input.nextLine();  
  
 InputStream is = new FileInputStream(path);  
  
 XSSFWorkbook xwb = new XSSFWorkbook(is);  
 XSSFSheet sheet = xwb.getSheet("Patients");  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 if (cell.getCellType() == CellType.*STRING*) {  
  
 if (cell.getRichStringCellValue().getString().trim().equals(name)) {  
  
 String message = String.*valueOf*(cell);  
  
 if (message.length() > 2) {  
  
 String path2 = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\medical\_histories.xlsx";  
  
 InputStream is2 = new FileInputStream(path2);  
  
 XSSFWorkbook xwb2 = new XSSFWorkbook(is2);  
 XSSFSheet personal\_sheet = xwb2.getSheet(name);  
  
 Row row2 = personal\_sheet.getRow(personal\_sheet.getLastRowNum());  
 Cell cell2 = row2.getCell(0);  
 String value = cell2.getStringCellValue();  
 System.*out*.println("The last date of your illness: " + value);}  
  
 else {  
 System.*out*.println("Mr(s) " + name + " your medical history is not in database");  
 System.*out*.println(*next\_step*());  
 }  
 }  
 }  
 }  
 }  
 System.*out*.println(*next\_step*());  
}

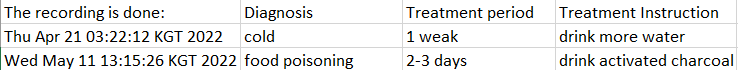
Pic. 8. Last date of illness

1. Medical history

By entering this menu, the user will receive his own medical history.

case "3" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patients.xlsx";  
  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.print("Your first name and last name: ");  
 String name = input.nextLine();  
  
 InputStream is = new FileInputStream(path);  
  
 XSSFWorkbook xwb = new XSSFWorkbook(is);  
 XSSFSheet sheet = xwb.getSheet("Patients");  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 if (cell.getCellType() == CellType.*STRING*) {  
  
 if (cell.getRichStringCellValue().getString().trim().equals(name)) {  
  
 String message = String.*valueOf*(cell);  
  
 if (message.length() > 2) {  
  
 String path2 = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\medical\_histories.xlsx";  
  
 InputStream is2 = new FileInputStream(path2);  
  
 XSSFWorkbook xwb2 = new XSSFWorkbook(is2);  
 XSSFSheet personal\_sheet = xwb2.getSheet(name);  
  
 FormulaEvaluator formulaEvaluator = xwb2.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row2 : personal\_sheet) {  
  
 for (Cell cell2 : row2) {  
  
 switch (formulaEvaluator.evaluateInCell(cell2).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell2.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell2.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();}}  
  
 else {  
 System.*out*.println("Mr(s) " + name + " your medical history is not in database");  
 System.*out*.println(*next\_step*());}  
 }  
 }  
 }  
}  
System.*out*.println(*next\_step*());}

Pic. 9. Medical history



Pic. 10. Medical history

1. Personal information

Entering this menu, the user must enter his first and last name, if he entered correctly, he will receive his personal data.

case "4" -> {  
  
String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patients.xlsx";  
  
Scanner input = new Scanner(System.*in*);  
  
System.*out*.print("Your first name and last name: ");  
String name = input.nextLine();  
  
InputStream is = new FileInputStream(path);  
  
XSSFWorkbook xwb = new XSSFWorkbook(is);  
XSSFSheet sheet = xwb.getSheet("Patients");  
  
for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 if (cell.getCellType() == CellType.*STRING*) {  
  
 if (cell.getRichStringCellValue().getString().trim().equals(name)) {  
  
 String message = String.*valueOf*(cell);  
  
 if (message.length() > 2) {  
  
 XSSFSheet personal\_sheet = xwb.getSheet(name);  
  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row2 : personal\_sheet) {  
  
 for (Cell cell2 : row2) {  
  
 switch (formulaEvaluator.evaluateInCell(cell2).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell2.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell2.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();}}

Pic. 11. Personal information

If he entered incorrectly, he will receive the following message "Your personal information is not in the database".

else {  
 System.*out*.println("Mr(s) " + name + " your personal information is not in database");  
 System.*out*.println(*next\_step*());}  
}

Pic. 12. Incorrect first and last name input

1. Treatment period

After entering this menu, the user must enter the first and last name, after which

the program will display the patient's treatment period.

case "5" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patients.xlsx";  
  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.print("Your first name and last name: ");  
 String name = input.nextLine();  
  
 InputStream is = new FileInputStream(path);  
  
 XSSFWorkbook xwb = new XSSFWorkbook(is);  
 XSSFSheet sheet = xwb.getSheet("Patients");  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 if (cell.getCellType() == CellType.*STRING*) {  
  
 if (cell.getRichStringCellValue().getString().trim().equals(name)) {  
  
 String message = String.*valueOf*(cell);  
  
 if (message.length() > 2) {  
  
 XSSFSheet personal\_sheet = xwb.getSheet(name);  
  
 Row row2 = personal\_sheet.getRow(personal\_sheet.getLastRowNum());  
 Cell cell2 = row2.getCell(2);  
 String value = cell2.getStringCellValue();  
 System.*out*.println("Your treatment period: " + value);}  
  
 else {  
 System.*out*.println("Mr(s) " + name + " information about your treatment period is not in database");  
 System.*out*.println(*next\_step*());  
 }  
 }  
 }  
 }  
 }  
 System.*out*.println(*next\_step*());  
}

Pic.13. Treatment period

1. Get back to the main page

After entering this menu, the user returns to the account selection.

case "6" -> System.*out*.println(*authorization*());

Pic. 14. Get back to the main page

1. Shut down the program

The program terminates and the following message “Goodbye! See you later” appears in front of the user.

case "7" -> System.*out*.println("Goodbye! See you later.");

Pic. 15. Shut down the program

# **Medassistant part**

After logging in to the Medassistant account, the user will receive the message "Authorization was successful".

public static String assistant\_checking() throws IOException {  
  
 String[] words;  
 FileReader fr = new FileReader("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assistant's\_data.txt");  
 BufferedReader br = new BufferedReader(fr);  
 String s;  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("""  
   
 Login: \s""");  
 String login = input.nextLine();  
 System.*out*.print("Password: ");  
 String password = input.nextLine();  
 String data = login + password;  
 int count = 0;  
 while ((s = br.readLine()) != null) {  
 words = s.split(" ");  
 for (String word : words) {  
 if (word.equals(data)) {  
 count++;}}}  
  
 if (count == 1) {  
 System.*out*.println("""  
   
 Authorization was successful  
 """);  
 System.*out*.println(*assistant*());}  
  
 else {  
 System.*out*.println("""  
   
 You have incorrectly entered your username and/or password  
 """);  
  
 System.*out*.print("""  
 Choose the next move  
   
 1 - try again  
   
 2 - get back to the main page  
 3 - shut down the program: \s""");  
 Scanner input\_next\_move\_after\_fail = new Scanner(System.*in*);  
 String next\_move\_after\_fail = input\_next\_move\_after\_fail.nextLine();  
 switch (next\_move\_after\_fail) {  
 case "1" -> System.*out*.println(*assistant\_checking*());  
 case "2" -> System.*out*.println(*authorization*());  
 case "3" -> System.*out*.println("Goodbye! See you later.");  
 default -> System.*out*.println("" + *assistant\_checking*());  
 }  
 }  
 return "";}

Pic. 16. Authorization in the medassistant’s account

Next, the user is asked to select one of the menus.

System.*out*.print("""  
   
 Enter the number of menu  
   
 1 - assignments  
 2 - complete an assignment  
 3 - completed assignments  
 4 - procedures  
 5 - search  
   
 6 - get back to the main page  
 7 - shut down the program: \s""");

Scanner input\_number\_of\_menu = new Scanner(System.*in*);  
String number\_of\_menu = input\_number\_of\_menu.nextLine();

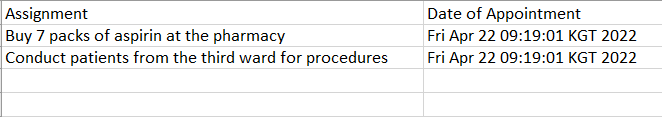
Pic. 17. Medassistantmenu

1. Assignments

By entering this menu, the user will receive a list of all orders written by the doctor.

switch (number\_of\_menu) {  
  
 case "1" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assignments.xlsx";  
  
 FileInputStream fis = new FileInputStream(path);  
 XSSFWorkbook xwb = new XSSFWorkbook(fis);  
 XSSFSheet sheet = xwb.getSheet("Assignments");  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 switch (formulaEvaluator.evaluateInCell(cell).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell.getStringCellValue() + "\t\t");  
 }  
 }  
  
 System.*out*.println();  
 }  
  
 System.*out*.println(*next\_step*());}

Pic. 18. List of assignments



Pic. 19. List of assignments

1. Complete an assignment

After entering this menu, the user must enter the completed task, then he enters his first and last name. This data is then saved in an Excel file.

case "2" -> {  
  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.print("Assignment: ");  
 String assignment = input.nextLine();  
  
 Date date = new Date();  
  
 System.*out*.print("Your first name and last name: ");  
 String name = input.nextLine();  
  
 FileInputStream file = new FileInputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assignments.xlsx");  
 XSSFWorkbook xwb = new XSSFWorkbook(file);  
 XSSFSheet sheet = xwb.getSheet("Completed Assignments");  
 int lastRow = sheet.getLastRowNum();  
  
 Row row = sheet.createRow(++lastRow);  
 row.createCell(0).setCellValue(assignment);  
 row.createCell(1).setCellValue(String.*valueOf*(date));  
 row.createCell(2).setCellValue(name);  
 file.close();  
  
 FileOutputStream output\_file =new FileOutputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assignments.xlsx");  
 xwb.write(output\_file);  
 output\_file.close();  
 System.*out*.println("""  
  
 The data has been successfully saved""");  
 System.*out*.println(*next\_step*());}

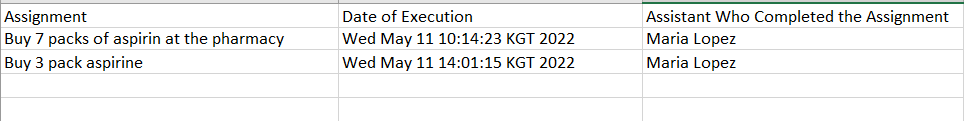
Pic. 20. Complete an assignment

1. Сompleted assignments

By entering this menu, the user will receive a list of all completed tasks written by the doctor.

case "3" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assignments.xlsx";  
  
 FileInputStream fis = new FileInputStream(path);  
 XSSFWorkbook xwb = new XSSFWorkbook(fis);  
 XSSFSheet sheet = xwb.getSheet("Completed Assignments");  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 switch (formulaEvaluator.evaluateInCell(cell).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();  
 }  
  
 System.*out*.println(*next\_step*());}

Pic. 21. Сompleted assignments



Pic. 22. Сompleted assignments

1. Procedures

After entering this menu, the user must enter his first and last name. Next, the program outputs all the procedures for the user

case "4" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\procedures.xlsx";  
  
 FileInputStream fis = new FileInputStream(path);  
 XSSFWorkbook xwb = new XSSFWorkbook(fis);  
  
 System.*out*.print("Patient's first name and last name: ");  
 Scanner input = new Scanner(System.*in*);  
 String name = input.nextLine();  
  
 XSSFSheet sheet = xwb.getSheet(name);  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 switch (formulaEvaluator.evaluateInCell(cell).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();  
 }  
  
 System.*out*.println(*next\_step*());}

Pic. 23. Procedures

1. Search

Entering this menu, the user must enter the patient's first and last name

case "5" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patients.xlsx";  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("Patient's first name and last name: ");  
 String name = input.nextLine();  
  
 InputStream is = new FileInputStream(path);  
  
 XSSFWorkbook xwb = new XSSFWorkbook(is);  
 XSSFSheet sheet = xwb.getSheet("Patients");  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 if (cell.getCellType() == CellType.*STRING*) {  
  
 if (cell.getRichStringCellValue().getString().trim().equals(name)) {  
  
 String message = cell + "'s information is recorded in the database";  
  
 if (message.length() > 43) {  
  
 System.*out*.println(message);

Pic. 24. Search

Then the user chooses whether he wants to view the patient's personal data.

System.*out*.print("Do you want to see patient's personal information? (0 - no / 1 - yes): ");  
 String choose\_step = input.nextLine();  
  
 if (Objects.*equals*(choose\_step, "1")) {  
  
 XSSFSheet personal\_sheet = xwb.getSheet(name);  
  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row2 : personal\_sheet) {  
  
 for (Cell cell2 : row2) {  
  
 switch (formulaEvaluator.evaluateInCell(cell2).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell2.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell2.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();  
 }  
 }  
 else {System.*out*.println(*next\_step*());}  
 }  
 }  
 }  
 }  
}  
 System.*out*.println(*next\_step*());  
}

Pic. 25. Patient's personal data

1. Get back to the main page

After entering this menu, the user returns to the account selection.

case "6" -> System.*out*.println(*authorization*());

Pic. 26. Get back to the main page

1. Shut down the program

The program terminates and the following message “Goodbye! See you later” appears in front of the user.

case "7" -> System.*out*.println("Goodbye! See you later.");

Pic. 27. Shut down the program

# **Doctor part**

After logging in to the doctor's account, the user will receive the message "Authorization was successful".

public static String doctor\_checking() throws IOException {  
  
 String[] words;  
 FileReader fr = new FileReader("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\doctor's\_data.txt");  
 BufferedReader br = new BufferedReader(fr);  
 String s;  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("""  
  
 Login: \s""");  
 String login = input.nextLine();  
 System.*out*.print("Password: ");  
 String password = input.nextLine();  
 String data = login + password;  
 int count = 0;  
 while ((s = br.readLine()) != null) {  
 words = s.split(" ");  
 for (String word : words) {  
 if (word.equals(data)) {  
 count++;}}}  
  
 if (count == 1) {  
 System.*out*.print("""  
  
 Authorization was successful  
 """);  
 System.*out*.println(*doctor*());}  
  
 else {  
 System.*out*.println("""  
  
 You have incorrectly entered your username and/or password, please repeat.  
 """);  
  
 System.*out*.print("""  
 Choose the next move  
  
 1 - try again  
  
 2 - get back to the main page  
 3 - shut down the program: \s""");  
 Scanner input\_next\_move\_after\_fail = new Scanner(System.*in*);  
 String next\_move\_after\_fail = input\_next\_move\_after\_fail.nextLine();  
 switch (next\_move\_after\_fail) {  
 case "1" -> System.*out*.println(*doctor\_checking*());  
 case "2" -> System.*out*.println(*authorization*());  
 case "3" -> System.*out*.println("Goodbye! See you later.");  
 default -> System.*out*.println("" + *doctor\_checking*());  
 }  
 }  
 return "";  
}

Pic. 28. Authorization in the doctor's account

Next, the user is asked to select one of the menus.

System.*out*.print("""  
  
 Enter the number of menu  
  
 1 - add  
 2 - assignments  
 3 - completed assignments  
 4 - delete  
 5 - give an assignment  
 6 - patients  
 7 - search  
  
 8 - get back to the main page  
 9 - shut down the program: \s""");

Pic. 29. Doctormenu

1. Add

After entering this menu, the user must enter the patient's personal data.

Scanner input\_the\_number\_of\_menu = new Scanner(System.*in*);  
String number\_of\_menu = input\_the\_number\_of\_menu.nextLine();  
  
switch (number\_of\_menu) {  
  
 case "1" -> {  
  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.print("Patient's first name: ");  
 String first\_name = input.nextLine();  
  
 System.*out*.print("Patient's last name: ");  
 String last\_name = input.nextLine();  
  
 System.*out*.print("Patient's date of birth (day.month.year): ");  
 String date\_of\_birth = input.nextLine();  
  
 System.*out*.print("Patient's weight: ");  
 String weight = input.nextLine();  
  
 System.*out*.print("Patient's height: ");  
 String height = input.nextLine();  
  
 System.*out*.print("Patient's blood type: ");  
 String blood\_type = input.nextLine();  
  
 System.*out*.print("Diagnosis: ");  
 String diagnosis = input.nextLine();  
  
 System.*out*.print("Treatment period: ");  
 String treatment = input.nextLine();  
  
 System.*out*.print("Treatment Instruction: ");  
 String instruction = input.nextLine();  
  
 Scanner input2 = new Scanner(System.*in*);  
 System.*out*.print("Procedure: ");  
 String procedure = input2.nextLine();  
  
 System.*out*.print("Timeline of procedure: ");  
 String timeline = input2.nextLine();  
  
 String txt\_path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patient's\_data.txt";  
 BufferedWriter bw = new BufferedWriter(new FileWriter(txt\_path, true));  
 bw.write("\n" + first\_name + last\_name);  
 bw.close();

FileInputStream file = new FileInputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patients.xlsx");  
  
 XSSFWorkbook xwb = new XSSFWorkbook(file);  
  
// first  
 XSSFSheet sheet = xwb.getSheet("Patients");  
 int lastRow1 = sheet.getLastRowNum();  
  
 Row row = sheet.createRow(++lastRow1);  
 row.createCell(0).setCellValue(first\_name + " " + last\_name);  
  
// second  
 xwb.createSheet(first\_name + " " + last\_name);  
 XSSFSheet sheet2 = xwb.getSheet(first\_name + " " + last\_name);  
  
 Row row2\_1 = sheet2.createRow(0);  
 row2\_1.createCell(0).setCellValue("Name");  
 row2\_1.createCell(1).setCellValue("Date of Birth");  
 row2\_1.createCell(2).setCellValue("Weight");  
 row2\_1.createCell(3).setCellValue("Height");  
 row2\_1.createCell(4).setCellValue("Blood Type");  
 row2\_1.createCell(5).setCellValue("Date of admission");  
  
 Date date = new Date();  
 Row row2\_2 = sheet2.createRow(1);  
 row2\_2.createCell(0).setCellValue(first\_name + " " + last\_name);  
 row2\_2.createCell(1).setCellValue(date\_of\_birth);  
 row2\_2.createCell(2).setCellValue(weight);  
 row2\_2.createCell(3).setCellValue(height);  
 row2\_2.createCell(4).setCellValue(blood\_type);  
 row2\_2.createCell(5).setCellValue(String.*valueOf*(date));  
 file.close();

Pic. 30. Entering personal data

The user then enters the patient's diagnosis, treatment period, treatment instructions, procedures, and procedure time. This data is then saved in an Excel file.

// third  
 FileInputStream file3 = new FileInputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\medical\_histories.xlsx");  
  
 XSSFWorkbook xwb3 = new XSSFWorkbook(file3);  
  
 xwb3.createSheet(first\_name + " " + last\_name);  
 XSSFSheet sheet3 = xwb3.getSheet(first\_name + " " + last\_name);  
  
 Row row3\_1 = sheet3.createRow(0);  
 row3\_1.createCell(0).setCellValue("The recording is done:");  
 row3\_1.createCell(1).setCellValue("Diagnosis");  
 row3\_1.createCell(2).setCellValue("Treatment Period");  
 row3\_1.createCell(3).setCellValue("Treatment Instruction");  
  
  
 Date date3 = new Date();  
 Row row3\_2 = sheet3.createRow(1);  
 row3\_2.createCell(0).setCellValue(String.*valueOf*(date3));  
 row3\_2.createCell(1).setCellValue(diagnosis);  
 row3\_2.createCell(2).setCellValue(treatment);  
 row3\_2.createCell(3).setCellValue(instruction);  
 file3.close();  
  
// fourth  
 FileInputStream file4 = new FileInputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\procedures.xlsx");  
  
 XSSFWorkbook xwb4 = new XSSFWorkbook(file4);  
  
 xwb4.createSheet(first\_name + " " + last\_name);  
 XSSFSheet sheet4 = xwb4.getSheet(first\_name + " " + last\_name);  
  
 Row row4\_1 = sheet4.createRow(0);  
 row4\_1.createCell(0).setCellValue("Procedure");  
 row4\_1.createCell(1).setCellValue("Timeline");  
  
 Row row4\_2 = sheet4.createRow(1);  
 row4\_2.createCell(0).setCellValue(procedure);  
 row4\_2.createCell(1).setCellValue(timeline);  
 file4.close();  
 FileOutputStream output\_file = new FileOutputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patients.xlsx");  
 xwb.write(output\_file);  
 output\_file.close();  
  
 FileOutputStream output\_file3 = new FileOutputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\medical\_histories.xlsx");  
 xwb3.write(output\_file3);  
 output\_file3.close();  
  
 FileOutputStream output\_file4 = new FileOutputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\procedures.xlsx");  
 xwb4.write(output\_file4);  
 output\_file4.close();  
  
 System.*out*.println("""  
  
 The data has been successfully saved""");  
 System.*out*.println(*next\_step*());}

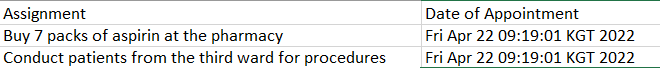
Pic. 31. Diagnostic appointments

1. Assignments

After entering this menu, the user is shown a list of assignments for the nurse.

case "2" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assignments.xlsx";  
  
 FileInputStream fis = new FileInputStream(path);  
 XSSFWorkbook xwb = new XSSFWorkbook(fis);  
 XSSFSheet sheet = xwb.getSheet("Assignments");  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 switch (formulaEvaluator.evaluateInCell(cell).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell.getStringCellValue() + "\t\t");  
 }  
 }  
  
 System.*out*.println();  
 }  
  
 System.*out*.println(*next\_step*());}

Pic. 32. List of assignments



Pic. 33. List of assignment

1. Completed assignments

After entering this menu, the user is shown a list of completed assignments.

case "3" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assignments.xlsx";  
  
 FileInputStream fis = new FileInputStream(path);  
 XSSFWorkbook xwb = new XSSFWorkbook(fis);  
 XSSFSheet sheet = xwb.getSheet("Completed Assignments");  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 switch (formulaEvaluator.evaluateInCell(cell).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell.getStringCellValue() + "\t\t");  
 }  
 }  
  
 System.*out*.println();  
 }  
  
 System.*out*.println(*next\_step*());  
}

Pic. 34. Completed assignments

1. Delete

After entering this menu, in order for the user to delete the patient, it is necessary to enter his first and last name, then the selected patient will be deleted.

case "4" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patients.xlsx";  
  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.print("Patient's first name: ");  
 String name = input.nextLine();  
  
 System.*out*.print("Patient's last name: ");  
 String last\_name = input.nextLine();  
  
 File file = new File("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patient's\_data.txt");  
  
 File temp = File.*createTempFile*("patient's\_data", ".txt", file.getParentFile());  
  
 String charset = "UTF-8";  
  
 BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream(file), charset));  
  
 PrintWriter writer = new PrintWriter(new OutputStreamWriter(new FileOutputStream(temp), charset));  
  
 for (String line; (line = reader.readLine()) != null;) {  
  
 line = line.replace(name + last\_name, "");  
 writer.println(line);  
 }  
 reader.close();  
 writer.close();  
  
 file.delete();  
  
 temp.renameTo(file);  
  
 FileInputStream fis = new FileInputStream(path);  
 XSSFWorkbook xwb = new XSSFWorkbook(fis);  
 XSSFSheet sheet = xwb.getSheet("Patients");  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 if (cell.getCellType() == CellType.*STRING*) {  
  
 if (cell.getRichStringCellValue().getString().trim().equals(name + " " + last\_name)) {  
  
 sheet.removeRow(cell.getRow());  
 }  
 }  
 }  
 }  
 xwb.removeSheetAt(xwb.getSheetIndex(name + " " + last\_name));  
  
 FileOutputStream fos = new FileOutputStream(path);  
 xwb.write(fos);  
 fos.close();  
  
 String path2 = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\medical\_histories.xlsx";  
  
 FileInputStream fis2 = new FileInputStream(path2);  
 XSSFWorkbook xwb2 = new XSSFWorkbook(fis2);  
  
 xwb2.removeSheetAt(xwb2.getSheetIndex(name + " " + last\_name));  
  
 FileOutputStream fos2 = new FileOutputStream(path2);  
 xwb2.write(fos2);  
 fos2.close();  
  
 String path3 = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\procedures.xlsx";  
  
 FileInputStream fis3 = new FileInputStream(path3);  
 XSSFWorkbook xwb3 = new XSSFWorkbook(fis3);  
  
 xwb3.removeSheetAt(xwb3.getSheetIndex(name + " " + last\_name));  
  
 FileOutputStream fos3 = new FileOutputStream(path3);  
 xwb3.write(fos3);  
 fos3.close();  
  
 System.*out*.println(*next\_step*());}

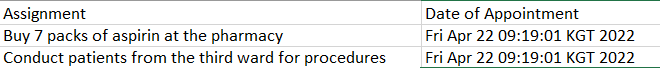
Pic. 35. Delete

1. Give an assignment

After entering this menu, the user must enter a task for the nurse. This data is then saved in an Excel file.

case "5" -> {  
  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("Assignment: ");  
 String assignment = input.nextLine();  
  
 Date date = new Date();  
  
 FileInputStream file = new FileInputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assignments.xlsx");  
 XSSFWorkbook xwb = new XSSFWorkbook(file);  
 XSSFSheet sheet = xwb.getSheet("Assignments");  
 int lastRow = sheet.getLastRowNum();  
  
 Row row = sheet.createRow(++lastRow);  
 row.createCell(0).setCellValue(assignment);  
 row.createCell(1).setCellValue(String.*valueOf*(date));  
 file.close();  
  
 FileOutputStream output\_file =new FileOutputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assignments.xlsx");  
 xwb.write(output\_file);  
 output\_file.close();  
 System.*out*.println("""  
  
 The data has been successfully saved""");  
 System.*out*.println(*next\_step*());}

Pic. 36. Give an assignment



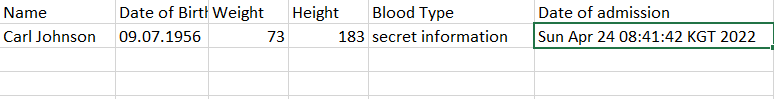
Pic. 37. List of assignment

1. Patients

After entering this menu, the user is shown a list of all patients and their number.

case "6" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patients.xlsx";  
  
 FileInputStream fis = new FileInputStream(path);  
 XSSFWorkbook xwb = new XSSFWorkbook(fis);  
 XSSFSheet sheet = xwb.getSheet("Patients");  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 switch (formulaEvaluator.evaluateInCell(cell).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();  
 }  
  
 int totalRows = sheet.getLastRowNum();  
 System.*out*.println("Total number of patients: " + totalRows);  
 System.*out*.println(*next\_step*());  
}

Pic. 38. List and number of patients



Pic. 39. List of patients

1. Search

If the user has selected the search menu, the user must enter the first and last name of the person he wants to find.

case "7" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patients.xlsx";  
  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.print("Patient's first name and last name: ");  
 String name = input.nextLine();  
  
 InputStream is = new FileInputStream(path);  
  
 XSSFWorkbook xwb = new XSSFWorkbook(is);  
 XSSFSheet sheet = xwb.getSheet("Patients");  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 if (cell.getCellType() == CellType.*STRING*) {  
  
 if (cell.getRichStringCellValue().getString().trim().equals(name)) {  
  
 String message = cell + "'s information is recorded in the database";  
  
 if (message.length() > 43) {  
  
 System.*out*.println(message);

Pic. 40. Search

Next, the user is asked if he wants to view the patient's personal data.

System.*out*.print("Do you want to see patient's medical history or personal information? (0 - no / 1 - yes): ");  
String choose\_step = input.nextLine();

Pic. 41. Confirmation

If the user agrees, then he must choose what he wants to see: medical history or personal data.

if (Objects.*equals*(choose\_step, "1")) {  
  
 System.*out*.println("""  
   
 Enter the number of menu  
   
 1 - medical history  
 2 - personal information: \s""");  
 String choose\_step2 = input.nextLine();  
  
 switch (choose\_step2){  
  
 case "1" -> {  
  
 String path2 = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\medical\_histories.xlsx";  
  
 InputStream is2 = new FileInputStream(path2);  
  
 XSSFWorkbook xwb2 = new XSSFWorkbook(is2);  
 XSSFSheet personal\_sheet = xwb2.getSheet(name);  
  
 FormulaEvaluator formulaEvaluator = xwb2.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row2 : personal\_sheet) {  
  
 for (Cell cell2 : row2) {  
  
 switch (formulaEvaluator.evaluateInCell(cell2).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell2.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell2.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();}}  
  
 case "2" -> {  
 XSSFSheet personal\_sheet2 = xwb.getSheet(name);  
  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row2 : personal\_sheet2) {  
  
 for (Cell cell2 : row2) {  
  
 switch (formulaEvaluator.evaluateInCell(cell2).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell2.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell2.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();}}  
  
 default -> System.*out*.println(*next\_step*());  
 }  
 }  
 else {System.*out*.println(*next\_step*());}  
 }  
 }  
 }  
 }  
 }  
 System.*out*.println(*next\_step*());  
}

Pic. 42. Patient's personal data

1. Get back to the main page

After entering this menu, you return back to the account selection.

case "8" -> System.*out*.println(*authorization*());

Pic. 43. Get back to the main page

1. Shut down the program

The program terminates and the following message “Goodbye! See you later” appears in

front of the user.

case "9" -> System.*out*.println("Goodbye! See you later.");

Pic. 44. Shut down the program

# **Main doctor part**

After logging in to the main doctor's account, the user will receive the message "Authorization was successful".

public static String main\_doctor\_checking() throws IOException {  
  
 String[] words;  
 FileReader fr = new FileReader("C:/Users/Wharton/Documents/Course Works/Course Work (Java)/Course Work/files/main\_doctor's\_data.txt");  
 BufferedReader br = new BufferedReader(fr);  
 String s;  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("""  
  
 Login: \s""");  
 String login = input.nextLine();  
 System.*out*.print("Password: ");  
 String password = input.nextLine();  
 String data = login + password;  
 int count = 0;  
 while ((s = br.readLine()) != null) {  
 words = s.split(" ");  
 for (String word : words) {  
 if (word.equals(data)) {  
 count++;}}}  
  
 if (count == 1) {  
 System.*out*.print("""  
  
 Authorization was successful  
 """);  
 System.*out*.println(*main\_doctor*());}  
  
 else {  
 System.*out*.println("""  
  
 You have incorrectly entered your username and/or password, please repeat.  
 """);  
  
 System.*out*.print("""  
 Choose the next move  
  
 1 - try again  
  
 2 - get back to the main page  
 3 - shut down the program: \s""");  
 Scanner input\_next\_move\_after\_fail = new Scanner(System.*in*);  
 String next\_move\_after\_fail = input\_next\_move\_after\_fail.nextLine();  
 switch (next\_move\_after\_fail) {  
 case "1" -> System.*out*.println(*main\_doctor\_checking*());  
 case "2" -> System.*out*.println(*authorization*());  
 case "3" -> System.*out*.println("Goodbye! See you later.");  
 default -> System.*out*.println("" + *main\_doctor\_checking*());  
 }  
 }  
 return "";  
}

Pic. 45. Authorization in the main doctor's account

Next, the user is asked to select one of the menus.

System.*out*.print("""  
  
 Enter the number of menu  
  
 1 - add  
 2 - assistants  
 3 - delete  
 4 - doctors  
 5 - patients  
 6 - search  
  
 9 - get back to the main page  
 10 - shut down the program: \s""");  
  
Scanner input\_number\_of\_menu = new Scanner(System.*in*);  
String number\_of\_menu = input\_number\_of\_menu.nextLine();

Pic. 46. Main doctormenu

1. Add

After entering this menu, the user must choose who he wants to add a doctor or a nurse.

case "1" -> {  
  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("""  
  
 Enter the type of account  
  
 1 - assistant  
 2 - doctor: \s""");  
 String type\_of\_account = input.nextLine();  
  
 switch (type\_of\_account) {

Pic. 47. Account Selection

If the user has chosen a nurse, then the user must enter the personal data of the nurse, then enter her salary. After the entered data is saved in a file.

case "1" -> {  
  
 Scanner input1 = new Scanner(System.*in*);  
  
 System.*out*.print("Assistant's first name: ");  
 String first\_name = input1.nextLine();  
  
 System.*out*.print("Assistant's last name: ");  
 String last\_name = input1.nextLine();  
  
 Scanner input\_int = new Scanner(System.*in*);  
 System.*out*.print("Assistant's salary: ");  
 int salary = input\_int.nextInt();  
  
 System.*out*.print("Assistant's age: ");  
 String age = input1.nextLine();  
  
 String txt\_path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assistant's\_data.txt";  
 BufferedWriter bw = new BufferedWriter(new FileWriter(txt\_path, true));  
 bw.write("\n" + first\_name + last\_name);  
 bw.close();  
  
 FileInputStream file = new FileInputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assistants.xlsx");  
  
 XSSFWorkbook xwb = new XSSFWorkbook(file);  
  
 XSSFSheet sheet1 = xwb.getSheet("Assistants");  
 int lastRow1 = sheet1.getLastRowNum();  
  
 Row row1 = sheet1.createRow(++lastRow1);  
 row1.createCell(0).setCellValue(first\_name + " " + last\_name);  
 row1.createCell(1).setCellValue(salary);  
  
 xwb.createSheet(first\_name + " " + last\_name);  
 XSSFSheet sheet2 = xwb.getSheet(first\_name + " " + last\_name);  
  
 Row row2\_1 = sheet2.createRow(0);  
 row2\_1.createCell(0).setCellValue("Name");  
 row2\_1.createCell(1).setCellValue("Age");  
 row2\_1.createCell(2).setCellValue("Salary");  
 row2\_1.createCell(3).setCellValue("Date Employment");  
  
 Date date = new Date();  
 Row row2\_2 = sheet2.createRow(1);  
 row2\_2.createCell(0).setCellValue(first\_name + " " + last\_name);  
 row2\_2.createCell(1).setCellValue(age);  
 row2\_2.createCell(2).setCellValue(salary);  
 row2\_2.createCell(3).setCellValue(String.*valueOf*(date));  
 file.close();  
  
 FileOutputStream output\_file = new FileOutputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patients.xlsx");  
 xwb.write(output\_file);  
 output\_file.close();  
  
 System.*out*.println("""  
  
 The data has been successfully saved""");  
 System.*out*.println(*next\_step*());}

Pic. 48. Entering personal data

If the user has chosen a doctor, the user must enter the doctor's personal data, then enter her salary, further the user enters the doctor's schedule. After the entered data is saved in the file.

case "2" -> {  
  
 Scanner input1 = new Scanner(System.*in*);  
  
 System.*out*.print("Doctor's first name: ");  
 String first\_name = input1.nextLine();  
  
 System.*out*.print("Doctor's last name: ");  
 String last\_name = input1.nextLine();  
  
 Scanner input\_int = new Scanner(System.*in*);  
 System.*out*.print("Doctor's salary: ");  
 int salary = input\_int.nextInt();  
  
 System.*out*.print("Doctor's age: ");  
 String age = input1.nextLine();  
  
 Scanner input\_schedule = new Scanner(System.*in*);  
  
 System.*out*.println("Doctor's schedule (enter the time interval in which the doctor works on this day. For example, 13:00-15:00)");  
  
 System.*out*.print("Monday: ");  
 String monday = input\_schedule.nextLine();  
  
 System.*out*.print("Tuesday: ");  
 String tuesday = input\_schedule.nextLine();  
  
 System.*out*.print("Wednesday: ");  
 String wednesday = input\_schedule.nextLine();  
  
 System.*out*.print("Thursday: ");  
 String thursday = input\_schedule.nextLine();  
  
 System.*out*.print("Friday: ");  
 String friday = input\_schedule.nextLine();  
  
 String txt\_path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\doctor's\_data.txt";  
 BufferedWriter bw = new BufferedWriter(new FileWriter(txt\_path, true));  
 bw.write("\n" + first\_name + last\_name);  
 bw.close();  
  
 FileInputStream file = new FileInputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\doctors.xlsx");  
  
 XSSFWorkbook xwb = new XSSFWorkbook(file);  
  
 XSSFSheet sheet1 = xwb.getSheet("Doctors");  
 int lastRow1 = sheet1.getLastRowNum();  
  
 Row row1 = sheet1.createRow(++lastRow1);  
 row1.createCell(0).setCellValue(first\_name + " " + last\_name);  
 row1.createCell(1).setCellValue(salary);  
  
 xwb.createSheet(first\_name + " " + last\_name);  
 XSSFSheet sheet2 = xwb.getSheet(first\_name + " " + last\_name);  
  
 Row row2\_1 = sheet2.createRow(0);  
 row2\_1.createCell(0).setCellValue("Name");  
 row2\_1.createCell(1).setCellValue("Age");  
 row2\_1.createCell(2).setCellValue("Salary");  
 row2\_1.createCell(3).setCellValue("Date Employment");  
  
 Date date = new Date();  
 Row row2\_2 = sheet2.createRow(1);  
 row2\_2.createCell(0).setCellValue(first\_name + " " + last\_name);  
 row2\_2.createCell(1).setCellValue(age);  
 row2\_2.createCell(2).setCellValue(salary);  
 row2\_2.createCell(3).setCellValue(String.*valueOf*(date));  
 file.close();  
  
 FileInputStream file3 = new FileInputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\d\_schedule.xlsx");  
  
 XSSFWorkbook xwb3 = new XSSFWorkbook(file3);  
  
 xwb3.createSheet(first\_name + " " + last\_name);  
 XSSFSheet sheet3 = xwb3.getSheet(first\_name + " " + last\_name);  
  
 Row row3\_1 = sheet3.createRow(0);  
 row3\_1.createCell(0).setCellValue("Monday");  
 row3\_1.createCell(1).setCellValue("Tuesday");  
 row3\_1.createCell(2).setCellValue("Wednesday");  
 row3\_1.createCell(3).setCellValue("Thursday");  
 row3\_1.createCell(4).setCellValue("Friday");  
  
  
 Row row3\_2 = sheet3.createRow(1);  
 row3\_2.createCell(0).setCellValue(monday);  
 row3\_2.createCell(1).setCellValue(tuesday);  
 row3\_2.createCell(2).setCellValue(wednesday);  
 row3\_2.createCell(3).setCellValue(thursday);  
 row3\_2.createCell(4).setCellValue(friday);  
 file3.close();  
  
 FileOutputStream output\_file = new FileOutputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\doctors.xlsx");  
 xwb.write(output\_file);  
 output\_file.close();  
  
 FileOutputStream output\_file3 = new FileOutputStream("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\d\_schedule.xlsx");  
 xwb3.write(output\_file3);  
 output\_file3.close();  
  
 System.*out*.println("""  
  
 The data has been successfully saved""");  
 System.*out*.println(*next\_step*());}  
  
 default -> System.*out*.print(*next\_step*());  
 }  
}

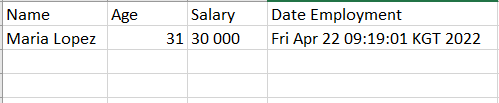
Pic. 49. Entering a schedule and saving data

1. Assistants

If the user has selected the "assistant" menu, the user is shown a list of assistant and their salary.

case "2" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assistants.xlsx";  
  
 FileInputStream fis = new FileInputStream(path);  
 XSSFWorkbook xwb = new XSSFWorkbook(fis);  
 XSSFSheet sheet = xwb.getSheet("Assistants");  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 switch (formulaEvaluator.evaluateInCell(cell).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell.getStringCellValue() + "\t\t");  
 }  
 }  
  
 System.*out*.println();  
 }  
  
 int totalRows = sheet.getLastRowNum();  
 System.*out*.println("Total number of assistants: " + totalRows);  
 System.*out*.println(*next\_step*());}

Pic. 50. Assistants



Pic. 51. List of assistants

1. Delete

After entering this menu, the user must choose who he wants to delete - a doctor or a nurse.

case "3" -> {  
  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("""  
  
 Enter the type of account  
  
 1 - assistant  
 2 - doctor: \s""");  
  
 String type\_of\_account = input.nextLine();

Pic. 52. Account Selection

If the user has decided to delete a nurse, then the user must enter the nurse's first and last name, and this nurse will be deleted from the file.

switch (type\_of\_account) {  
  
 case "1" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assistants.xlsx";  
  
 Scanner input2 = new Scanner(System.*in*);  
  
 System.*out*.print("Assistant's first name and last name: ");  
 String name = input2.nextLine();  
  
 File file = new File("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assistant's\_data.txt");  
  
 File temp = File.*createTempFile*("assistant's\_data", ".txt", file.getParentFile());  
  
 String charset = "UTF-8";  
  
 BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream(file), charset));  
  
 PrintWriter writer = new PrintWriter(new OutputStreamWriter(new FileOutputStream(temp), charset));  
  
 for (String line; (line = reader.readLine()) != null;) {  
  
 line = line.replace(name, "");  
 writer.println(line);  
 }  
 reader.close();  
 writer.close();  
  
 file.delete();  
  
 temp.renameTo(file);  
  
 FileInputStream fis = new FileInputStream(path);  
 XSSFWorkbook xwb = new XSSFWorkbook(fis);  
 XSSFSheet sheet = xwb.getSheet("Assistants");  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 if (cell.getCellType() == CellType.*STRING*) {  
  
 if (cell.getRichStringCellValue().getString().trim().equals(name)) {  
  
 sheet.removeRow(cell.getRow());  
 }  
 }  
 }  
 }  
 xwb.removeSheetAt(xwb.getSheetIndex(name));  
  
 FileOutputStream fos = new FileOutputStream(path);  
 xwb.write(fos);  
 fos.close();  
  
 System.*out*.println(*next\_step*());}

Pic. 53. Deleting an assistant

If the user has decided to delete the doctor, the user must enter the first and last name of the doctor, and the doctor will be deleted from the file.

case "2" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\doctors.xlsx";  
  
 Scanner input2 = new Scanner(System.*in*);  
  
 System.*out*.print("Doctor's first name and last name: ");  
 String name = input2.nextLine();  
  
 File file = new File("C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\doctor's\_data.txt");  
  
 File temp = File.*createTempFile*("doctor's\_data", ".txt", file.getParentFile());  
  
 String charset = "UTF-8";  
  
 BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream(file), charset));  
  
 PrintWriter writer = new PrintWriter(new OutputStreamWriter(new FileOutputStream(temp), charset));  
  
 for (String line; (line = reader.readLine()) != null;) {  
  
 line = line.replace(name, "");  
 writer.println(line);  
 }  
 reader.close();  
 writer.close();  
  
 file.delete();  
  
 temp.renameTo(file);  
  
 FileInputStream fis = new FileInputStream(path);  
 XSSFWorkbook xwb = new XSSFWorkbook(fis);  
 XSSFSheet sheet = xwb.getSheet("Doctors");  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 if (cell.getCellType() == CellType.*STRING*) {  
  
 if (cell.getRichStringCellValue().getString().trim().equals(name)) {  
  
 sheet.removeRow(cell.getRow());  
 }  
 }  
 }  
 }  
 xwb.removeSheetAt(xwb.getSheetIndex(name));  
  
 FileOutputStream fos = new FileOutputStream(path);  
 xwb.write(fos);  
 fos.close();  
  
 String path2 = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\d\_schedule.xlsx";  
  
 FileInputStream fis2 = new FileInputStream(path2);  
 XSSFWorkbook xwb2 = new XSSFWorkbook(fis2);  
  
 xwb2.removeSheetAt(xwb2.getSheetIndex(name));  
  
 FileOutputStream fos2 = new FileOutputStream(path2);  
 xwb2.write(fos2);  
 fos2.close();  
  
 System.*out*.println(*next\_step*());}  
  
 default -> System.*out*.print(*next\_step*());  
 }  
}

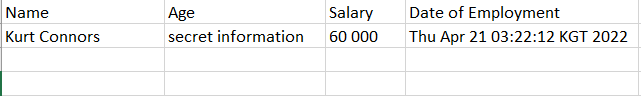
Pic. 54. Removing the doctor

1. Doctor

If the user has selected the "doctor" menu, the user is shown a list of doctors and their salary.

case "4" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\doctors.xlsx";  
  
 FileInputStream fis = new FileInputStream(path);  
 XSSFWorkbook xwb = new XSSFWorkbook(fis);  
 XSSFSheet sheet = xwb.getSheet("Doctors");  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 switch (formulaEvaluator.evaluateInCell(cell).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell.getStringCellValue() + "\t\t");  
 }  
 }  
  
 System.*out*.println();  
 }  
  
 int totalRows = sheet.getLastRowNum();  
 System.*out*.println("Total number of doctors: " + totalRows);  
 System.*out*.println(*next\_step*());}

Pic. 55. Doctor



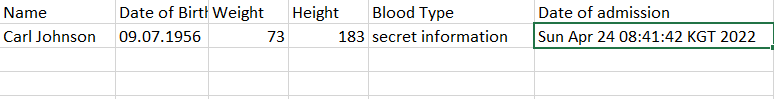
Pic. 56. List of doctors

1. Patients

By entering the "patients" menu, the user will receive a list of all patients and their number.

case "5" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patients.xlsx";  
  
 FileInputStream fis = new FileInputStream(path);  
 XSSFWorkbook xwb = new XSSFWorkbook(fis);  
 XSSFSheet sheet = xwb.getSheet("Patients");  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 switch (formulaEvaluator.evaluateInCell(cell).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell.getStringCellValue() + "\t\t");  
 }  
 }  
  
 System.*out*.println();  
 }  
 int totalRows = sheet.getLastRowNum();  
 System.*out*.println("Total number of patients: " + totalRows);  
 System.*out*.println(*next\_step*());  
}

Pic. 57. Patients



Pic. 58. List of patients

1. Search

Search menu, entering this menu, the user chooses who he wants to find: doctor; nurse; patient.

case "6" -> {  
 System.*out*.println("""  
  
 Enter the type of account  
  
 1 - assistant  
 2 - doctor  
 3 - patient: \s""");  
 Scanner input\_type\_of\_account = new Scanner(System.*in*);  
 String type\_of\_account = input\_type\_of\_account.nextLine();

Pic. 59. Account Selection

If the user has chosen to search for a nurse, then the user must enter the first and last name of whom he wants to find.

case "1" -> {  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\assistants.xlsx";  
  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.print("Assistant's first name and last name: ");  
 String name = input.nextLine();  
  
 InputStream is = new FileInputStream(path);  
  
 XSSFWorkbook xwb = new XSSFWorkbook(is);  
 XSSFSheet sheet = xwb.getSheet("Assistants");  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 if (cell.getCellType() == CellType.*STRING*) {  
  
 if (cell.getRichStringCellValue().getString().trim().equals(name)) {  
  
 String message = cell + "'s information is recorded in the database";  
  
 if (message.length() > 43) {  
  
 System.*out*.println(message);  
 System.*out*.print("Do you want to see assistant's personal information? (0 - no / 1 - yes): ");  
 String choose\_step = input.nextLine();  
  
 if (Objects.*equals*(choose\_step, "1")) {  
  
 XSSFSheet personal\_sheet2 = xwb.getSheet(name);  
  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row2 : personal\_sheet2) {  
  
 for (Cell cell2 : row2) {  
  
 switch (formulaEvaluator.evaluateInCell(cell2).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell2.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell2.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();}}  
  
 }  
 else {System.*out*.println(*next\_step*());}  
 }  
 }  
 }  
 }  
 System.*out*.println(*next\_step*());  
 }

Pic. 60. Search for an assistant

If the user has chosen to search for a doctor, the user must also enter the first and last name of the person he wants to find. Next, the user is asked if he wants to view the doctor's personal data. If the user agrees, then he must then choose what he wants to see: the doctor's schedule or personal data.

case "2" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\doctors.xlsx";  
  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.print("Doctor's first name and last name: ");  
 String name = input.nextLine();  
  
 InputStream is = new FileInputStream(path);  
  
 XSSFWorkbook xwb = new XSSFWorkbook(is);  
 XSSFSheet sheet = xwb.getSheet("Doctors");  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 if (cell.getCellType() == CellType.*STRING*) {  
  
 if (cell.getRichStringCellValue().getString().trim().equals(name)) {  
  
 String message = cell + "'s information is recorded in the database";  
  
 if (message.length() > 43) {  
  
 System.*out*.println(message);  
 System.*out*.print("Do you want to see doctor's schedule or personal information? (0 - no / 1 - yes): ");  
 String choose\_step = input.nextLine();  
  
 if (Objects.*equals*(choose\_step, "1")) {  
  
 System.*out*.println("""  
   
 Enter the number of menu  
   
 1 - schedule  
 2 - personal information: \s""");  
 String choose\_step2 = input.nextLine();  
  
 switch (choose\_step2){  
  
 case "1" -> {  
  
 String path2 = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\d\_schedule.xlsx";  
  
 InputStream is2 = new FileInputStream(path2);  
  
 XSSFWorkbook xwb2 = new XSSFWorkbook(is2);  
 XSSFSheet personal\_sheet = xwb2.getSheet(name);  
  
 FormulaEvaluator formulaEvaluator = xwb2.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row2 : personal\_sheet) {  
  
 for (Cell cell2 : row2) {  
  
 switch (formulaEvaluator.evaluateInCell(cell2).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell2.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell2.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();}}  
  
 case "2" -> {  
 XSSFSheet personal\_sheet2 = xwb.getSheet(name);  
  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row2 : personal\_sheet2) {  
  
 for (Cell cell2 : row2) {  
  
 switch (formulaEvaluator.evaluateInCell(cell2).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell2.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell2.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();}}  
  
 default -> System.*out*.println(*next\_step*());  
 }  
 }  
 else {System.*out*.println(*next\_step*());}  
 }  
 }  
 }  
 }  
 }  
 System.*out*.println(*next\_step*());  
}

Pic. 61. Search for a doctor

If the user has chosen to search for a patient, the user must also enter the first and last name of the person he wants to find. Next, the user is asked if he wants to view the patient's personal data. If the user agrees, then he must choose what he wants to see: medical history or personal data.

case "3" -> {  
  
 String path = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\patients.xlsx";  
  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.print("Patient's first name and last name: ");  
 String name = input.nextLine();  
  
 InputStream is = new FileInputStream(path);  
  
 XSSFWorkbook xwb = new XSSFWorkbook(is);  
 XSSFSheet sheet = xwb.getSheet("Patients");  
  
 for (Row row : sheet) {  
  
 for (Cell cell : row) {  
  
 if (cell.getCellType() == CellType.*STRING*) {  
  
 if (cell.getRichStringCellValue().getString().trim().equals(name)) {  
  
 String message = cell + "'s information is recorded in the database";  
  
 if (message.length() > 43) {  
  
 System.*out*.println(message);  
 System.*out*.print("Do you want to see patient's medical history or personal information? (0 - no / 1 - yes): ");  
 String choose\_step = input.nextLine();  
  
 if (Objects.*equals*(choose\_step, "1")) {  
  
 System.*out*.println("""  
   
 Enter the number of menu  
   
 1 - medical history  
 2 - personal information: \s""");  
 String choose\_step2 = input.nextLine();  
  
 switch (choose\_step2){  
  
 case "1" -> {  
  
 String path2 = "C:\\Users\\User\\IdeaProjects\\Course Work (Java)\\Course Work\\files\\medical\_histories.xlsx";  
  
 InputStream is2 = new FileInputStream(path2);  
  
 XSSFWorkbook xwb2 = new XSSFWorkbook(is2);  
 XSSFSheet personal\_sheet = xwb2.getSheet(name);  
 FormulaEvaluator formulaEvaluator = xwb2.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row2 : personal\_sheet) {  
  
 for (Cell cell2 : row2) {  
  
 switch (formulaEvaluator.evaluateInCell(cell2).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell2.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell2.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();}}  
  
 case "2" -> {  
 XSSFSheet personal\_sheet2 = xwb.getSheet(name);  
  
 FormulaEvaluator formulaEvaluator = xwb.getCreationHelper().createFormulaEvaluator();  
  
 for (Row row2 : personal\_sheet2) {  
  
 for (Cell cell2 : row2) {  
  
 switch (formulaEvaluator.evaluateInCell(cell2).getCellType()) {  
  
 case *NUMERIC* -> System.*out*.print(cell2.getNumericCellValue() + "\t\t");  
 case *STRING* -> System.*out*.print(cell2.getStringCellValue() + "\t\t");  
 }  
 }  
 System.*out*.println();}}  
  
 default -> System.*out*.println(*next\_step*());  
 }  
 }  
 else {System.*out*.println(*next\_step*());}  
 }  
 }  
 }  
 }  
 }  
 System.*out*.println(*next\_step*());  
}

Pic. 62. Patient Search

1. Get back to the main page

After entering this menu, the user returns to the account selection.

case "7" -> System.*out*.println(*authorization*());

Pic. 63. Get back to the main page

1. Shut down the program

The program terminates and the following message “Goodbye! See you later” appears in front of the user.

case "8" -> System.*out*.println("Goodbye! See you later.");

Pic. 64. Shut down the program

# **Conclusion**

The result of our work was the creation of a program thanks to which all patients of the clinic have access to all necessary information, such as personal data, medical history, treatment days, and can also view the doctors' schedules. The nurse has access to view a list of procedures for patients and a list of assignments, and the nurse can also mark the tasks that she has completed. The doctor has access to viewing lists: patients; assignments for nurses. Also, the doctor can write an assignment for the nurse, and give the diagnosis to the patient. The main doctor, however, has the greatest abilities, he can either add a doctor and a nurse, or remove them. It can also search for all doctors, nurses and patients. The main doctor has access to the lists: nurses; doctors; patients. Also, the main doctor can look at the employee and his salary.

Summing up, we can say that it was much easier to create a program for users of this field than in the first semester. Also, on our way there were various problems, from the easy at that time, to the hard. For example, there were some difficulties with the syntax of the Java programming language, because it differs from Python.

Throughout the entire process of creating the coursework, our team improved in writing code and studied a lot of useful information. Each participant contributed. We really worked as a team.

Our work was hard, but quite interesting. We learned how to work better with various files using software libraries, and also improved our programming and documentation skills.

# **References**

Documentation of Java: <https://docs.oracle.com/en/java/>

Documentation of excel: <https://betacode.net/11259/read-write-excel-file-in-java-using-apache-poi>

Documentation of csv: <https://techcave.ru/posts/99-rabotaem-s-failami-csv-v-java-s-ispolzovaniem-biblioteki-opencsv.html>

Documentation of txt: <https://metanit.com/java/tutorial/6.11.php>