Одеський національний політехнічний університет

Інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Пояснювальна записка

до розрахунково-графічної роботи

з дисципліни «Проектний практикум»

на тему «Система підтримки рішень для автоматизованого медичного діагностування»

Виконав: ст. гр. АС-123

Тимченко Б.І.

Перевірив:

доц. Блажко О. А.

Одеса, 2016

**ЗМІСТ**

[1 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ 3](#_Toc449562924)

[1.1 Визначення бізнес-вимог 3](#_Toc449562925)

[1.2 Визначення власних вимог 4](#_Toc449562926)

[1.3 Функціональні вимоги 8](#_Toc449562927)

[1.4 Нефункціональні вимоги 13](#_Toc449562928)

[1.5 Планування розробки 14](#_Toc449562929)

[Етап 1. Оцінка акторів. 21](#_Toc449562930)

[Етап 2. Оцінка варіантів використання. 22](#_Toc449562931)

[Етап 3. Оцінка технічних факторів 22](#_Toc449562932)

[Етап 4. Оцінка зовнішніх факторів 24](#_Toc449562933)

[Етап 5. Результуючі оцінки. 25](#_Toc449562934)

[2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ 26](#_Toc449562935)

[2.1 Концептуальне проектування 26](#_Toc449562936)

[2.2 Логічне проектування 29](#_Toc449562937)

[2.2.1 Алгоритми 29](#_Toc449562938)

[2.2.2 Інтерфейс користувача 31](#_Toc449562939)

[3 КОНСТРУЮВАННЯ ПРОГРАМНОГО ПРОДУКТУ 35](#_Toc449562940)

[3.1 Опис програмних технологій 35](#_Toc449562941)

[3.2 Опис програмних бібліотек 35](#_Toc449562942)

[3.3 Особливості створення програмних модулів з урахуванням мови програмування 35](#_Toc449562943)

[3.4 Особливості створення структур даних 35](#_Toc449562944)

[3.5 Модульне тестування 36](#_Toc449562945)

[3.5.1 Тестування моделі 37](#_Toc449562946)

[Модель: Використання декількох нейронних мереж 38](#_Toc449562947)

[3.5.2 Тестування моделі 39](#_Toc449562948)

[4 ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ 40](#_Toc449562949)

[**4.1** Функціональне тестування 40](#_Toc449562950)

[5 РОЗГОРТАННЯ ПРОГРАМНОГО ПРОДУКТУ 52](#_Toc449562951)

[5.1 Інструкція з встановлення 52](#_Toc449562952)

[5.2 Інструкція з використання 52](#_Toc449562953)

# 1 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

## 1.1 Визначення бізнес-вимог

**Предметна область**: медицина, діагностика захворювань дихальних шляхів.

Терміни:

**КВВП-дані** (Конденсат вологи повітря, що видихається) - на основі цих даних ставиться діагноз.

**Медкартки** - медична карта пацієнта, містить таку інформацію: ПІБ, стать, рік народження, історія хвороб, алергія, контактні дані та інше.

**Спектрометр** - прилад, за допомогою якого проводиться збір КВВП-даних.

**Класифікатор** - програма, яка групує наявні КВВП-дані і визначає, до яких хвороб що відноситься. Спочатку проходить навчання, потім зіставлення з новими даними і видача результату.

**Об'єкт автоматизації**: процес діагностування хвороби у пацієнта.

**Зацікавлені особи**: лікар, пацієнт.

**Проблема**: відсутність у лікаря програмного забезпечення, яке дозволяє проводити діагностику. Діагностика проводиться вручну, тобто довго (кілька хвилин-годин, наприклад), до того ж може спричинити за собою помилки (людський фактор). Автоматизація дозволяє проводити діагностику не більше 2-3 секунд.

На табл. 1.1 зображена порівняльна таблиця аналогів.

Таблиця 1.1 – Порівняльна таблиця аналогів системи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Оцінка молекул | Ідент. бактерій | Ідент. хвороб | Ідент. Пульмонологічних хвороб |
| Dendral | + |  |  |  |
| MYCIN |  | + |  |  |
| CADUCEUS |  |  | + | + |
| DiaSpectrEx | + |  | + | + |

Першим проектом в медицині, що використовував машинне навчання, був Dendral – перша експертна система. Вона була розроблена у ранніх 1960-х та за допомогою «знання» хімії могла оцінювати невідомі молекули за їхніми мас-спектрограмами.

Другою експертною системою, що мала значно ширші можливості, була MYCIN, що був похідною системою від Dendral. Вона використовувала принципи штучного інтелекту для того, щоб ідентифікувати бактерії та визначити курс лікування. Оператор відповідав на довгі серії простих питань, на базі чого система виводила список бактерій, відсортований за вірогідністю їхнього прояву.

Найпотужнішою системою вважається CADUCEUS, що була здатна відрізнити більше тисячі захворювань. Вона була розроблена в 1980-х роках.

Також існувало багато немедичних експертних систем, але зараз вони класифікуються не як штучний інтелект, але як структурні пошукові машини.

**Мета**: написання якісного ПО для діагностування. Підвищення зручності, за рахунок створення friendly user interface, підвищення точності діагностування (діагностування за допомогою класифікатора дає точність 95% на даний момент).

## 1.2 Визначення власних вимог

Для формалізації вимог до розроблюваної системі було проведено збір та аналіз інформації на основі документів «Бачення», сформованих майбутніми користувачами системи. Користувачів системи доцільно розбити на наступні категорії:

* «**Пацієнт**» (Patient) - обстежуваний, у якого в процесі діагностування ідентифікується стан дихальної системи. Комп'ютерна система повинна вміти вводити, обробляти і зберігати дані про безліч пацієнтів;
* «**Медичний працівник**» (Medicals) - особа, яка виконує заходи щодо проведення обстеження. Дану категорію можна розбити на дві підкатегорії: «Лаборант» (Laborant) і «Лікар» (Doctor). На лаборанта покладаються формалізовані дії - введення даних про пацієнта і запуск обчислювальних процедур, що входять в комп'ютерну систему. Лікар відповідає за постановку діагнозу в термінах предметної області;
* «**Адміністратор**» (Admin) - особа, основною функцією якого є пересилання результатів діагностування пацієнтові.

Крім того, одні функції системи мають строго певного виконавця, інші ж можуть виконуватися різними категоріями користувачів (наприклад, реєстрація, авторизація і т.д.).

Система повинна забезпечувати проведення реєстрації (первинне використання системи) та авторизації (використання системи при наявності облікового запису) користувачів. Система повинна передбачати можливість внесення основних персональних, медичних і додаткових (допоміжних) даних про пацієнта. Після дослідження спектрального складу конденсату вологості видихається пацієнтом повітря система повинна реєструвати КВВП-дані обстежуваного.

На рис. 1.1 зображена діаграма варіантів використання.

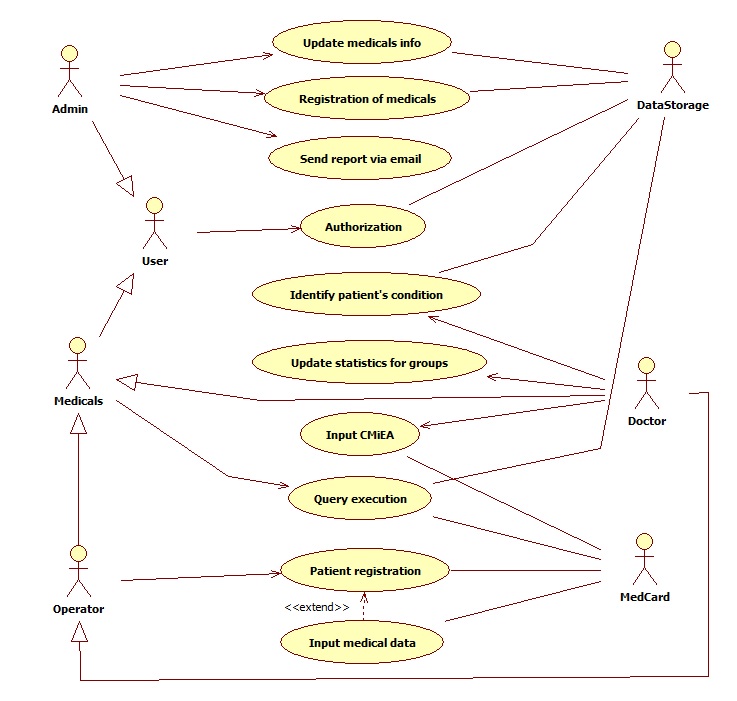


Рисунок 1.1 – Діаграма варіантів використання проектованої системи

Програма повинна передбачати формування медичної карти пацієнта, для якої можливі наступні режими: створення нової карти з унікальним номером, перегляд, доповнення вже створеної карти даними, редагування, роздруківка, зберігання в базі даних, пошук карти в базі даних за різними критеріями - простим і складовим. Кожен пацієнт при необхідності може бути обстежений необмежену кількість разів з внесенням всіх результатів обстеження в свою медичну карту. Також необхідна можливість угрупування пацієнтів з метою перегляду вибірки даних або завдання певної дії відразу для групи пацієнтів (наприклад, для відправки результатів діагностики відразу декільком пацієнтам). При наявності контактних даних пацієнта система повинна відправляти діагностичні дані за вказаною електронною адресою. При відсутності потрібної інформації під час пошуку по базі даних або угрупованню має виводитися відповідне повідомлення. Також необхідно видавати повідомлення про помилку при відсутності доступу до БД. Система не повинна поширювати персональну інформацію про пацієнта.

Програма повинна виконувати обробку та аналіз цілісності КВВП-даних пацієнта, здійснювати подальший спектральний аналіз КВВП-даних, перевіряти спектри на відповідність нормальному закону розподілу, обчислювати значення діагностичних ознак відповідно до закладених методик і алгоритмів. На підставі розрахованих значень діагностичних ознак ідентифікується стан дихальної системи пацієнта. При неможливості проведення автоматичної ідентифікації задіюється лікар, який співвідносить пацієнта з однієї з існуючих діагностичних груп або створює для пацієнта нову групу. Кожна група повинна мати свій опис, що зберігається у відповідних таблицях БД.

У кожного користувача є свої обмеження у використанні системи: лаборант може вносити дані про пацієнта, формувати медичну карту, виконувати обчислення діагностичних ознак, а також розрахунок спектрів і перевірку нормальності КВВП-даних пацієнта. Всі медпрацівники можуть змінювати дані про пацієнта, але оновлювати медичну карту може лише лікар. Також лікар може співвідносити дані пацієнта з діагностичною групою і оновлювати діагностику (перераховувати статистики для групи). Адміністратор відповідає за пересилання діагностичних дані пацієнтові, якщо він залишив свої контактні дані (номер мобільного телефону, e-mail).

Так як розроблювальний програмний продукт призначений для використання медпрацівниками, які володіють обчислювальною технікою на рівні користувача, то він повинен бути легкий і простий у використанні, тобто володіти зручним дружнім інтерфейсом. Завдання дій має здійснюватися з використанням команд меню, кнопок піктограм, основним діям повинні відповідати підказки. Крім того, необхідно розробити доступну довідкову систему, що дозволяє швидко і легко опанувати принципами роботи з програмою.

Програмний продукт має власний логотип та назву. Логотип продукту зображено на рис. 1.2.

Назва продукту: DiaSpectrEx



Рисунок 1.2. – Логотип програмної системи

## 1.3 Функціональні вимоги

Специфікація прецедентів (варіантів використання – ВВ).

1. *ВВ «Registration of medicals».*

Передумова: Програма запущена, є права для реєстрації у відповідному профілі.

1. Запуск режиму реєстрації.
2. Адміністратор вводить дані.
3. Перевірка правильності даних.

3а. Дані невірні, перехід до п. 2.

1. Збереження даних в DataStorage.
2. *ВВ «Authorization».*

Передумова: Запущений режим «Авторизація».

1. Користувач вводить логін та пароль.
2. Перевірка правильності введених даних в DataStorage.

2а. Невірний логін або пароль. Перехід до п.1.

1. Вхід до головного меню програми.
2. *ВВ «Input CMiEA».*

Передумова: Запущений режим «Сеанс».

1. Лікар вводить значення КВВП-даних.
2. Система перевіряє коректність вводу.

2а. Дані некоректні, виведення повідомлення про помилку на екран. Перехід до п.1.

1. Збереження введених даних у MedCard.
2. *ВВ «Patient registration».*

Передумова: Оператор авторизувався.

1. Оператор обирає режим «Зареєструвати пацієнта».
2. Оператор вводить дані про пацієнта (ПІБ, телефон, паспорт, історія хвороб, алергія, дата народження, дата вступу на облік, організація, що направила).
3. Перевірка коректності вводу.

3а. Дані введені некоректно, перехід до п. 2.

1. Збереження даних в MedCard.
2. *ВВ «Send report via email».*

Передумова: Адміністратор авторизувався. Сформований звіт про діагностування пацієнта.

1. Адміністратор обирає необхідного пацієнта зі списку всіх пацієнтів.

1а. Пацієнт не знайдений, перехід до п.1.

1. Адміністратор обирає звіт зі списку звітів.
2. Адміністратор відправляє обраний звіт на email пацієнта.
3. *ВВ «Update medicals info».*

Передумова: Адміністратор авторизувався і обрав режим оновлення даних о медпрацівниках.

1. Адміністратор обирає необхідного медпрацівника зі списку всіх медпрацівників.
2. Адміністратор обирає режим «Змінити дані».
3. Адміністратор змінює необхідні дані.
4. Збереження даних у DataStorage.
5. *ВВ «Identify patient’s condition».*

Передумова: Лікар обрав режим «Сеанс» та ввів КВВП-дані.

1. Система виконує класифікацію введених даних.

1а. Дані не належать жодному з відомих класів. Повернення помилки.

1. Система виводить передбачувану хворобу пацієнта.
2. *ВВ «Update statistics for groups».*

Передумова: Лікар авторизувався та обрав режим «Оновлення статистики».

1. Перевірка кількості даних у MedCard.

1а. Недостатня кількість для оновлення, вивід повідомлення на екран.

1. Вивід оновленої статистики на екран.
2. *ВВ «Input medical data».*

Передумова: Створена мед карта пацієнта, у неї внесені обов’язкові дані про пацієнта.

1. Оператор вводить медичні дані про пацієнта (передбачувана діагностична група, організація, що направила, алергія, історія хвороб).
2. Перевірка коректності введених даних.

2а. Дані некоректні, перехід до п.1.

1. Збереження даних у MedCard.
2. *ВВ «Query execution».*

Передумова: Медпрацівник авторизувався.

1. Медпрацівник обирає необхідний запит зі списку запитів.
2. Вводить додаткові дані до запиту.
3. Система виконує запит і виводить результат на екран.

Розглянемо на рис. 1.3 виконання деяких варіантів використання у вигляді алгоритмів або представлення покрокового виконання бізнес-процесу за допомогою діаграми діяльності.

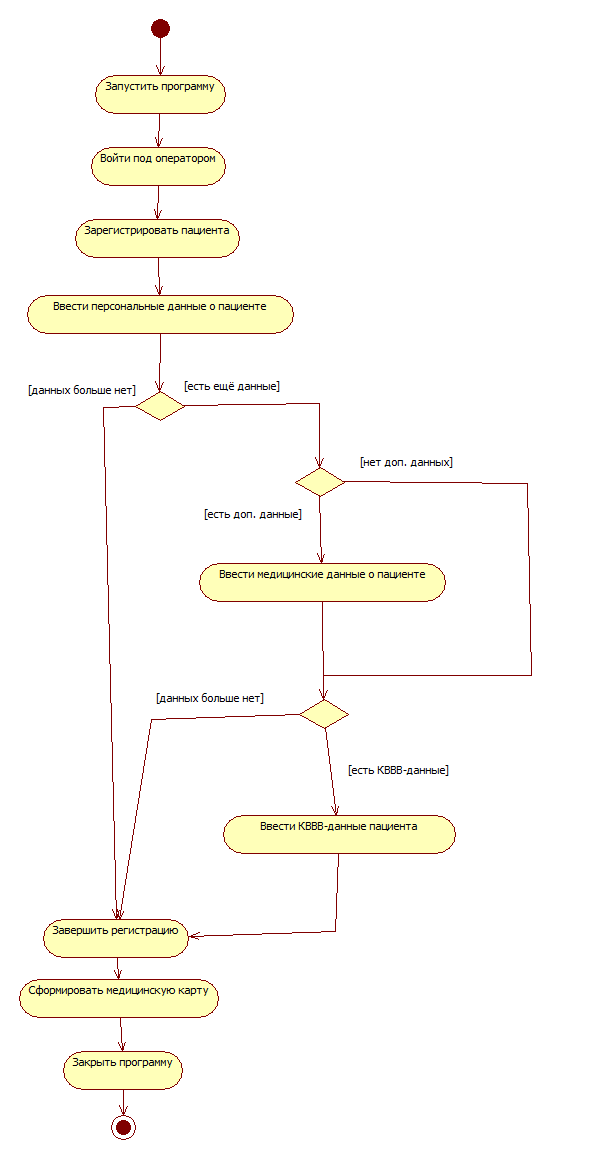


Рисунок 1.3 - Діаграма діяльності ВВ «Patient registration»

На рис. 1.4 зображена діаграма діяльності для усієї системи.

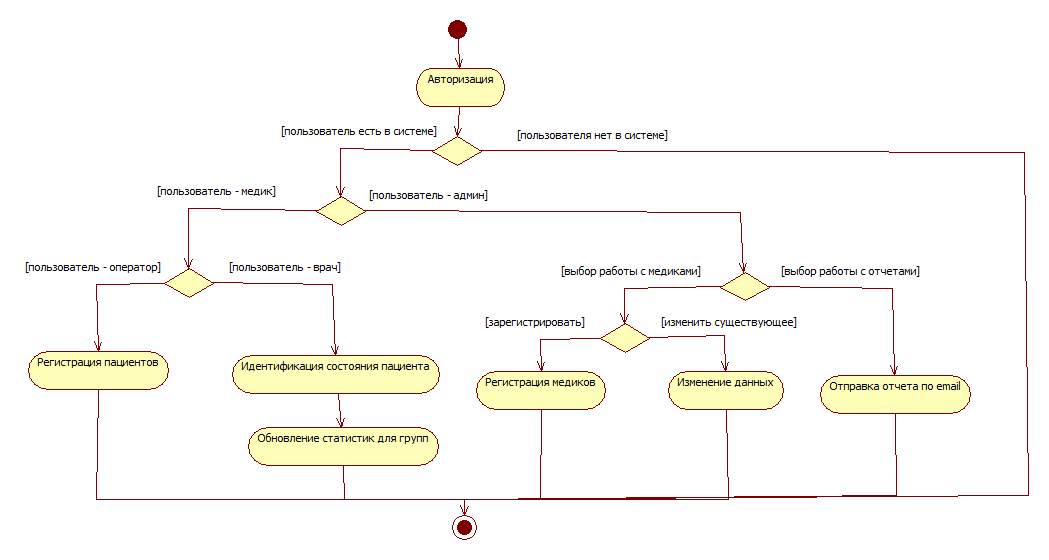


Рисунок 1.4 - Діаграма діяльності усієї системи вцілому

## 1.4 Нефункціональні вимоги

**Функціональність**

Правильність

Точність - А = 186, Б = 200 => 0.93, де А - кількість елементів даних, реалізованих із заданними рівнями точності, Б - кількість елементів даних, для яких в специфікації задані рівні точності.

Придатність

Повнота функціональної реалізації - система відповідає заявленим специфікаціям. 1 - А/Б = 1 - 2/20 = 0,9, де А - число нереалізованих функцій, Б - число функцій, описаних в специфікації.

**Мобільність**

Налагодження - система має бути встановлена в зазначеному середовищі.

Об’єм робот по установці - А/Б = 7/8 =  0.88, де А - кількість автоматичних кроків інсталяції, Б - необхідна кількість кроків.

Взаємозамінність - система може бути використана замість існуючого програмнго забезпечення.

Спадкоємність даних - А/Б = 15/20 = 0.75, де А - кількість елементів даних ПС, що продовжують використовуватися після заміщення, Б - кількість елементів старих даних, які повинні використовуватися зі старого ПС.

**Надійність**

Завершеність

Кількість тестових комбінацій, виконаних при перевірці = 6, кількість тестових комбінацій за вимогами = 7 => 6/7 = 0.85

Відновлення

Кількість реалізованих вимог до відновлення = 5, загальна кількість вимог до відновлення = 12 => 0.42

Стійкість до помилки

Кількість безпечних функцій =  45, кількість небезпечних дій = 67 => 45 / 67 = 0.67

**Функціональність**

Придатність - 0,9 (відсутні 2 функції, описані 20 => 1 - 2/20 = 0.9)

Правильність - (Кількість результатів з незадовільним рівнем точності - 3, час роботи - 25 годин => 3/25 = 0.12)

**Мобільність**

Адаптованість (кількість працездатних даних при інсталяції - 32, очікуване число працездатних даних - 40 => 32/40 = 0.8)

Налагодження - число успішних випадків інсталяції = 7, загальне число = 11 => 7/11 = 0.63

Сумісність - кількість обмежень = 5, тривалість роботи = 10 годин => 5/10 = 0.5

**Надійність**

Стійкість до помилки - кількість запобіглих відмов = 10, тестових випадків 13 => 10/13 = 0.77

Відновлення - успішне відновлення = 8, згідно вимог 10 випадків потрібно розглянути => 8/10 =  0.8

**Продуктивність**

Коефіцієнт продуктивності

Продуктивний час = 20, тривалість задачі =15, тривалість допомоги = 7, тривалість обробки помилок = 4, тривалість пошуку = 2 => 20/(15-(7+4+2))  = 10

Система має опрацьовувати дані швидше за інші існуючі способи.

**Результативність**

Завершені задачі = 127, всього задач = 129 => 127/129 = 0.98

**Задоволеність**

Кількість випадків використання програми згідно специфікації = 120, кількість випадків запланованого використання = 129 => 0.93

## 1.5 Планування розробки

На рис. 1.5 зображена діаграма класів ПС.

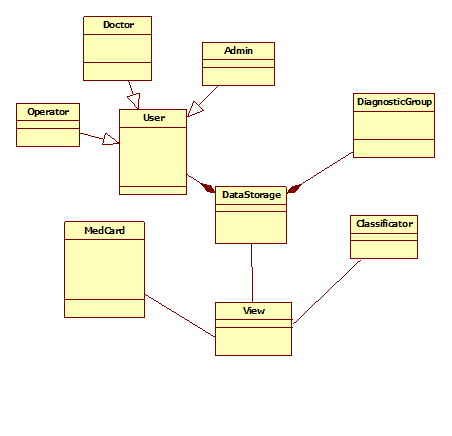


Рисунок 1.5 - Діаграма класів

Інтерфейс програми відображається у класі View, база даних усієї системи представляється у вигляді двох класів MedCard – клас, який предоставляє інформацію про пацієнтів, DataStorage – дані про медичний персонал та діагностичні групи. Усі обчислення щодо класифікації відбуваються у класі Classificator.

Розглянемо на рисунках 1.6 – 1.8 виконання деяких варіантів використання з плином часу, що дозволяє продемонструвати діаграма взаємодії (послідовностей).

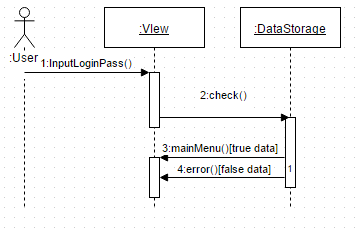


Рисунок 1.6 - Діаграма послідовностей ВВ «Authorization»

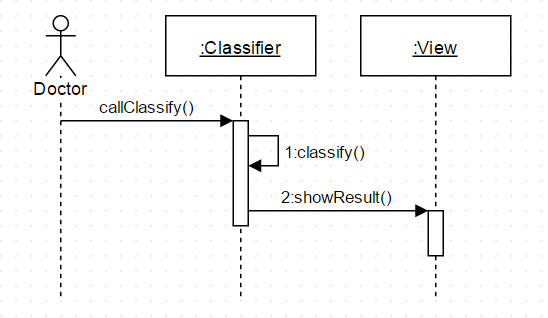


Рисунок 1.7 - Діаграма послідовностей ВВ «Identify patient’s condition»

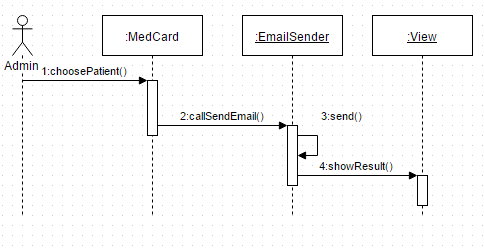


Рисунок 1.8 - Діаграма послідовностей ВВ «Send report via email»

**Конфигурация дерева решений**

Конфігураця дерева рішенья склаждається з формування дерева ров’язків, що будується на основі XML документа. Нижче наведена структура XML документа.

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<tree>

<classifier idx = **"2"**>

<disease>**Pneumonia**</disease>

<code>**4**</code>

<path>**cp-a.net**</path>

</classifier>

<classifier idx = **"1"**>

<disease>**Bronchitis**</disease>

<code>**2**</code>

<path>**cb-a.net**</path>

</classifier>

<classifier idx = **"3"**>

<disease>**Normal**</disease>

<code>**1**</code>

<path>**cn-a.net**</path>

</classifier>

</tree>

**Описание тегов конфигурации**

<?xml version=**"1.0"** encoding=**"utf-8"**?> - версія та кодування файла

<tree></tree> - корневий елемент, дерево класифікації

<classifier></classifier> - елемент, що описує підкласифікатор дерева, атрибут idx описує його порядок в класифікаторі

<path></path>- путь до файлу з нейронною мережею, що відповідає за певний діагноз (відносно файла classifier.xml)

<code></code> - код захворювання

<disease></disease> - название захворювання

**Присутність всіх тегів обов’язкова**

Кроссплатформове. СУБД MySQL. Структури даних - РБД.

IDE NetBeans (Java). Система контролю версій - Git.

**План розробки:**

Перш за все, потрібно визначити терміни дипломного проектування. Роботи було розпочато в момент отримання відомостей про необхідність проектування у 7-8 семестрах навчання. Отже, початком робіт буде **1 вересня 2015** року. Великий захист дипломної роботи буде проведено **1 липня 2016 року**, отже, до цієї дати проектування має бути повністю виконане.

Припустимо, що, оскільки це дипломний проект, студент працює кожен день. Таким чином, в наявності є **304** дні.

Також в деякі етапи проектування залучений дипломний керівник, який працює лише по будніх днях. Будемо вважати, що керівник реагує моментально і дає відповіді без затримки, а отже, не впливає на критичний шлях. У першому етапі будо визначено, що для реалізації проекту буде потрібно 1714 годин, або 75 днів, якщо працювати 24 години на добу. Можна вивести оцінку кількості годин на добу, враховуючи оцінку кількості днів на реалізацію.

Nгод = 1714 / 304 = 5.63

Малоймовірно, що студент буде працювати 6 годин на добу кожен день, тому для зниження навантаження деякі етапи можна виконувати паралельно.

Можна побудувати структурну декомпозицію робіт (WBS). WBS у середовищі MS Project показана на рис. 2. Основні етапи робіт пов’язані також з датами виконання курсових проектів, адже вони перетинаються з дипломом.

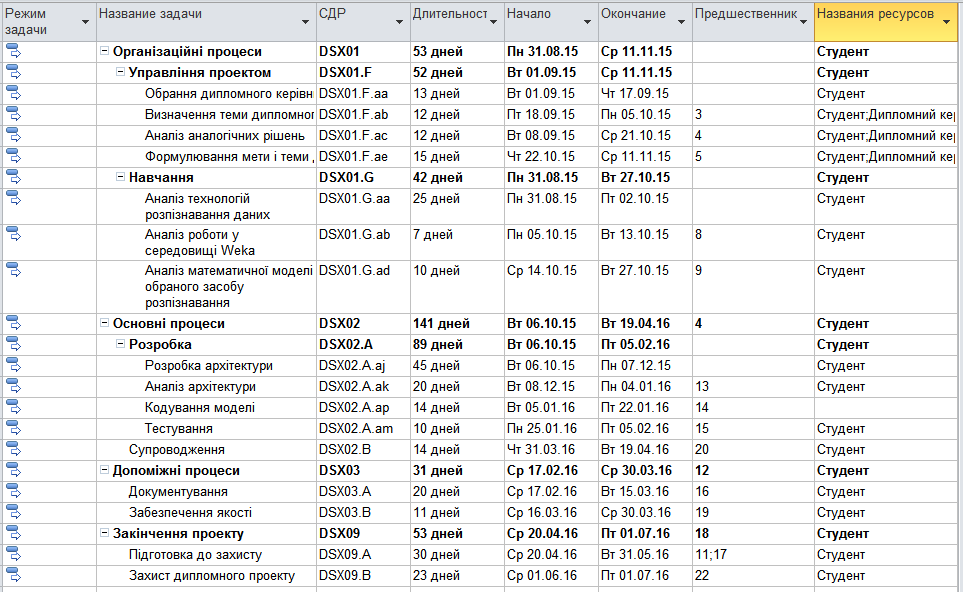


Рисунок 1.9 - Структурна декомпозиція робіт

На основі отриманих даних побудуємо діаграму Ґантта (рис. 2.2) Критичний шлях зображено червоним (сіреньким) кольором. Критичними операціями є операції лежать на критичному шляху З використанням паралельних процесів можна скоротити час роботи на добу приблизно втричі. Така можливість також додає буфер часу, на випадок недодержання плану.

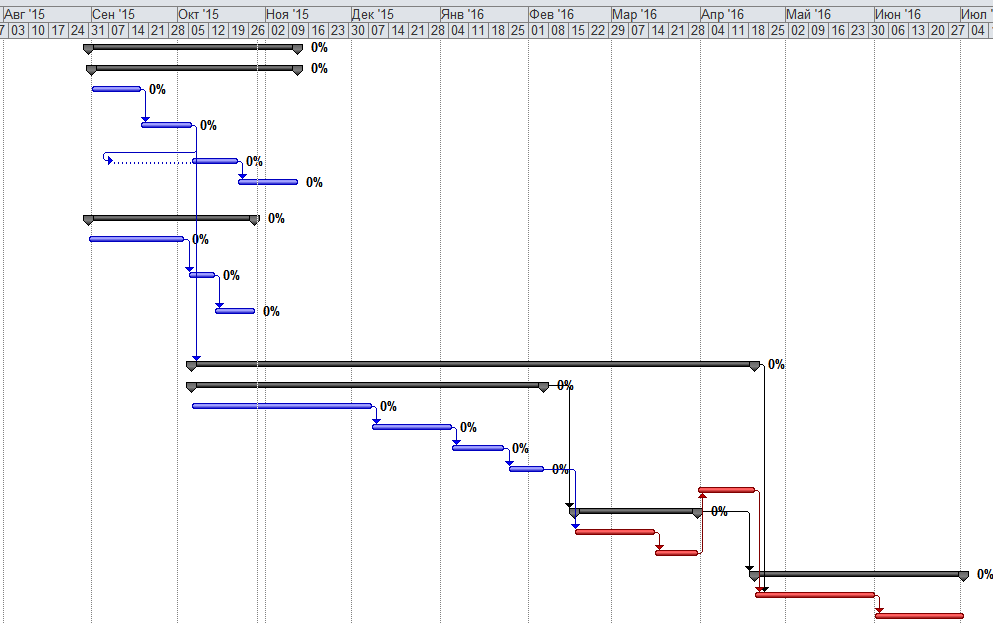


Рисунок 1.10 – Діаграма Ґантта

**5.2) Определение объемов работ по одному из методов оценки -**

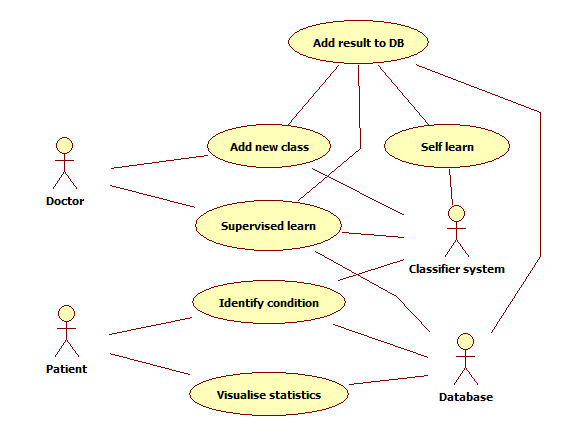


Рисунок 1.11 - Діаграма варіантів використання

## Етап 1. Оцінка акторів.

Таблиця 1.1 – Оцінка акторів

|  |  |  |  |
| --- | --- | --- | --- |
| Індивідуальні актори | | Множник | Назва |
| 1 | Простий | 1 | Database |
| 2 | Середньої складності | 2 | Classifier system |
| 3 | Складний | 3 | Doctor |
| 4 | Складний | 3 | Paient |

UAW = 2 \* 3 + 2 \* 1 + 1\*1 = 9

Складні актори – взаємодія з користувачем, Прості – інша система з визначеним API (у даному випадку взаємодія з СУБД).

## Етап 2. Оцінка варіантів використання.

Для зручності оцінювання буду використовувати підхід до визначення типу варіанту використання, що базується на кількості кроків, що будуть задіяні при реалізації варіантів використання.

Простий – менше за 3 кроки

Середньої складності – від 4 до 7 кроків

Складний – більше за 7 кроків

Таблиця 1.2 – Оцінка варіантів використання

|  |  |  |  |
| --- | --- | --- | --- |
| Складність | | Множник | Варіант використання |
| 1 | Простий | 5 | Visualize statistics |
| 2 | Середньої складності | 10 | Identify condition |
| 3 | Складний | 15 | Self learn |
| 4 | Складний | 15 | Add new class |
| 5 | Складний | 15 | Supervised learn |
| 6 | Простий | 5 | Add result to databse |

UUCW = 5\*2+10\*1+15\*3 = 65

UCP = UAW + UUCW

UCP = 9 + 65 = 74

Проект використовує нові технології, для реалізації яких немає готових компонентів.

Припустимо, що %REUSE = 10%

UCP = (100 – 10)/100 \* 74 = 66.6

## Етап 3. Оцінка технічних факторів

Таблиця 1.3 – Оцінка технічних факторів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Технічний фактор | | Множник | Відносний вплив | Опис |
| 1 | Розподілена система | 2 | **3** | Система не потребує розподілених обчислень, але їх можна застосувати |
| 2 | Час відклику важливий | 1 | **3** | Час відгуку сильно впливає на розробку системи |
| Продовження таблиці 1.3 | | | | |
| 3 | Ефективність кінцевого користувача | 1 | **1** | Ефективність користування має середній вплив на розробку |
| 4 | Складні внутрішні обчислення | 1 | **5** | Застосовуються алгоритми класифікації |
| 5 | Фокус на повторному використанні коду | 1 | **2** | При оновленні або додаванні даних до групи повторно використовується код |
| 6 | Простота інсталяції | 0,5 | **2** | При встановленні програми знадобиться допомога спеціаліста |
| 7 | Зручність використання | 0,5 | **5** | Система зосереджена на взаємодії з користувачем, тому |
| 8 | Кросплатформенність | 2 | **4** | Система є платформо-незалежною, але це не має важливий вплив на розробку |
| 9 | Легке внесення змін | 1 | **5** | При розробці системи береться до уваги той факт, що система повинна бути здатною до легкого внесення змін |
| 10 | Паралельні обчислення | 1 | **1** | Паралельні обчислення можуть мати місце, але не використовуються на даний момент |
| 11 | Виділений захист | 1 | **2** | Система потребує захист персональних даних пацієнтів |
| 12 | Залежність від чужого коду | 1 | **3** | Система може використовуватися зовнішніми системами/акторами |
| 13 | Навчання користувача | 1 | **2** | Потрібно провести тренінг користувача |
| Обчислений TCF | |  | **1,015** |  |

## 

## Етап 4. Оцінка зовнішніх факторів

Таблиця 1.4 – Оцінка зовнішніх факторів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Зовнішній фактор | | Множник | Відносний вплив | Опис |
| 1 | Ознайомленість з проектом | 1,5 | **3** | Знання команди предметної області має значний вплив на розробку |
| 2 | Досвід розробки | 0,5 | **2** | Досвід команди має середній вплив |
| 3 | Знання ООП | 1 | **4** | Бажаними є знання ООП |
| 4 | Здібності аналітика | 0,5 | **4** | Важливо правильно отримати вимоги |
| 5 | Мотивація | 1 | **3** | Мотивація має середній вплив |
| 6 | Стабільність вимог | 2 | **4** | Вимоги не часто будуть змінюватись, тому це не дуже впливає на розробку |
| 7 | Участь частково зайнятих працівників | -1 | **3** | Всі розробники – студенти, тому частково зайняті |
| 8 | Складність мови програмування | -1 | **3** | Мова програмування – Java, також потрібні знання SQL. Для проекту вистачить місяця на опанування мов |
| Обчислений EF | |  | **0,905** |  |

## Етап 5. Результуючі оцінки.

Скоректовані UCP обчислюються за формулою AUCP = UCP \* TCF \* EF.

AUCP = 66.6 \* 1.015 \* 0.905 = 61.18.

У даному випадку 1 UCP відповідає 28 робочим годинам. Отже, загальний час на розробку дорівнює: 28 \* 61.18 = 1714 (год)

# 2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

## 2.1 Концептуальне проектування

Розглянемо ER-діаграму (Рис. 2.1 та Рис. 2.2).



Рисунок 2.1 – ER-діаграма (частина перша)

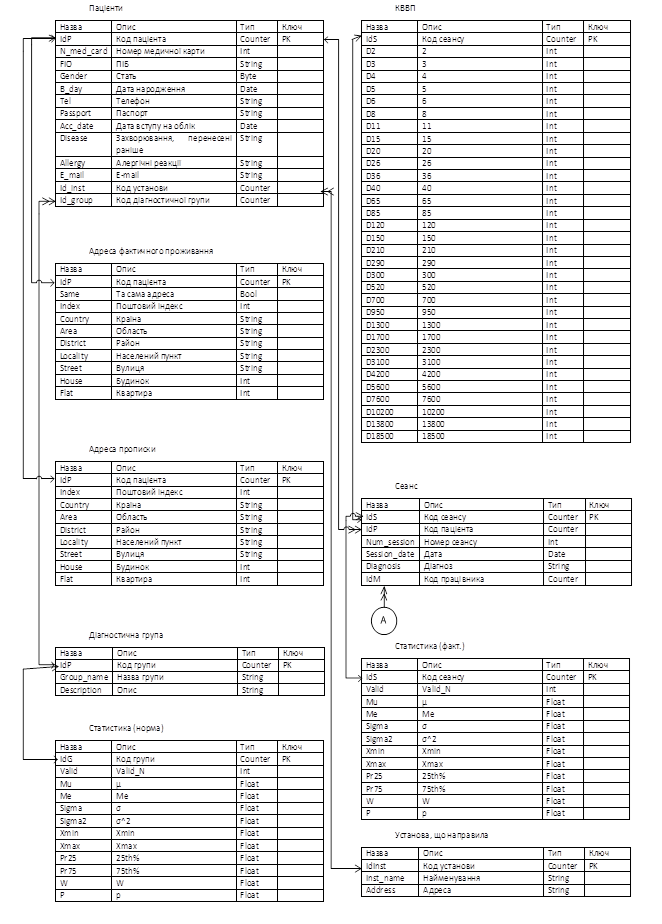


Рисунок 2.2 – ER-діаграма (частина друга)

Логічне проектування (без алгоритмики): діаграма програмних класів; опис структур даних.

Діаграма програмних класів зображена на Рис. 2.3.



Рисунок 2.3 – Діаграма програмних класів

Діаграма класів (спрощений варіант) дещо нагадує шаблон проектування MVC і є більш доречним варіантом для створення програмного коду для даної системи.

Інтерфейс програми відображається у класі View, база даних усієї системи представляється у вигляді двох класів MedCard – клас, який предоставляє інформацію про пацієнтів, DataStorage – дані про медичний персонал та діагностичні групи. Усі обчислення щодо класифікації відбуваються у класі Classificator.

## 2.2 Логічне проектування

### 2.2.1 Алгоритми

Функція 1: Навчання нейронної мережі. ІІ алгоритм зображено на рис. 2.1.:

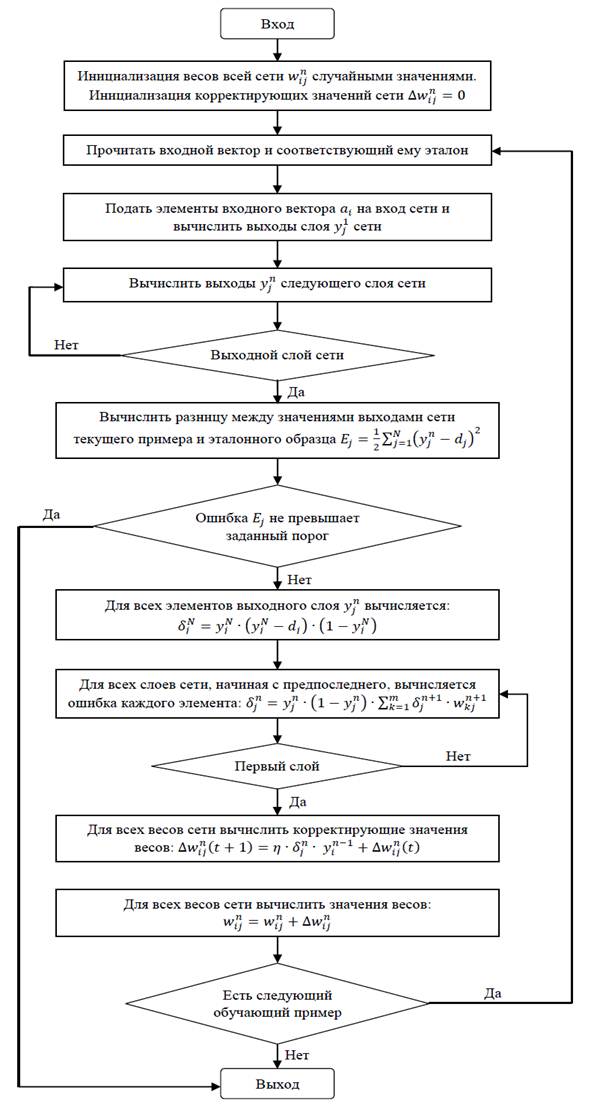


Рисунок 2.4 – Алгоритм навчання методом зворотного розповсюдження помилки

На рис. 2.5. зображено алгоритм Функції 2, що представляє собою навчання дерева розв’язків:

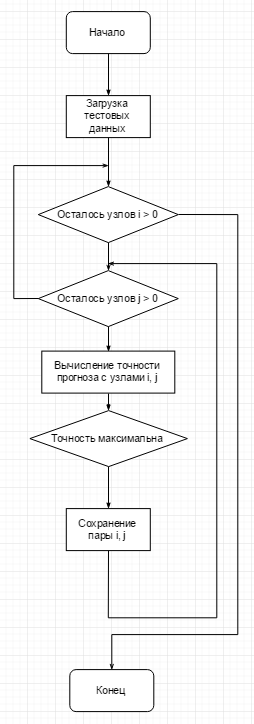


Рисунок 2.5 – Формування дерева ров’язків

Оцінка обчислювальної складності простого блока: О(1), константний час;

Перевірка складної умови: О(n), де n – кількість вузлів

Алгоритмічна складність циклів: О(n), лінійний час

Загальна оцінка обчислювальної складності:

**Функція** **1** – Квадратична складність О(n^2), де n – кількість нейронів в мережі

**Функція 2** – Квадратична складність О(n^2), де n – кількість вузлів дерева

### 2.2.2 Інтерфейс користувача

У цьому пункті розглянуто декілька основних графічних форм (інтерфейс користувача) для ознайомлення з системою.

Після запуску програмного продукту спочатку бачимо вікно авторизації (Рис. 2.6).

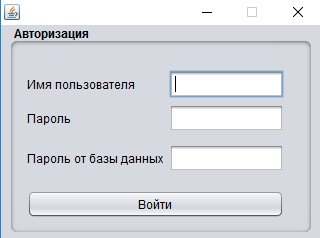


Рисунок 2.6 – Авторизація

Розглянемо Процес постановки діагнозу. Спочатку потрібно ввести КВВП-дані (Рис. 2.7)

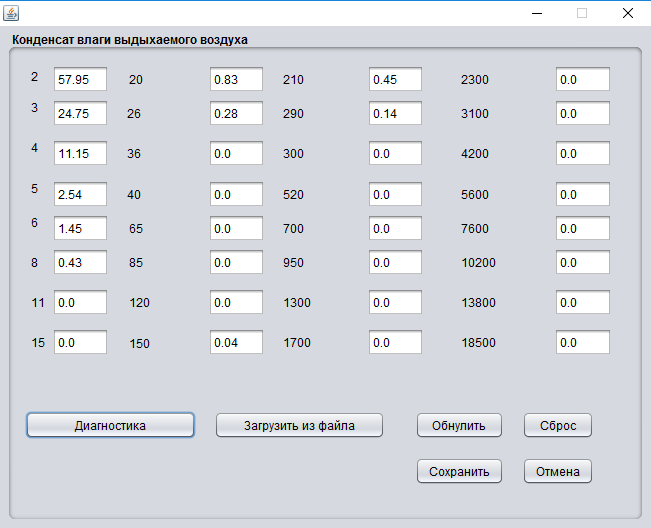


Рисунок 2.7 – Ввод КВВП

Далі зображено карту пацієнта, де розміщена вся необхідна інформація про нього для лікаря та лікарні в цілому (Рис. 2.8)

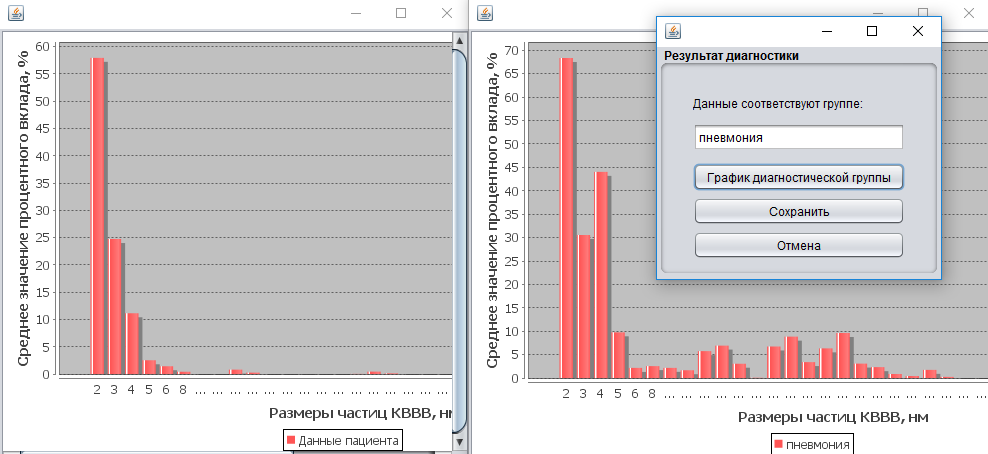


Рисунок 2.8 – Гістограми

Наступна форма демонструє додавання нового сеанса в сесію (Рис. 2.9).

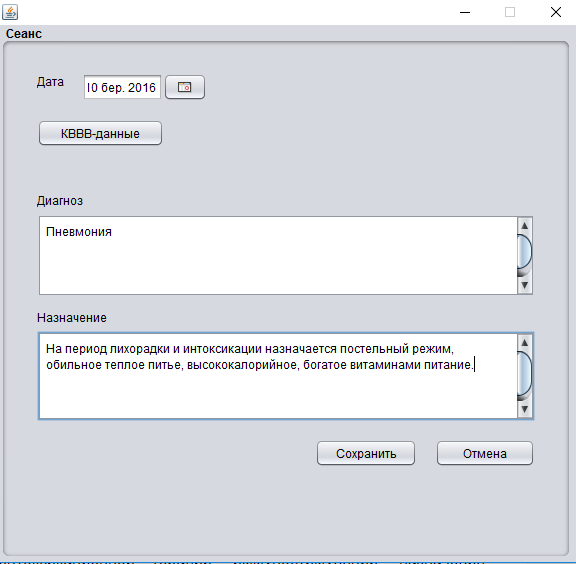


Рисунок 2.9 – Додавання нового сеанса в сесію

Також можна додавати та змінювати існуючі діагностичні групи (Рис. 2.10)

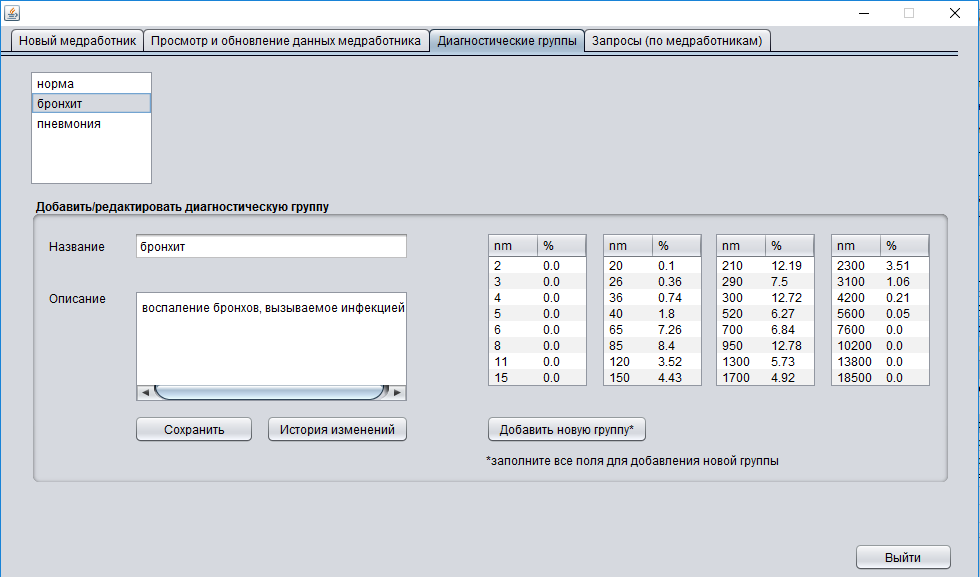


Рисунок 2.10 – Додавання та редагування діагностичних груп

У режимі оператора можна знаходити необхідних пацієнтів за заданими критеріями пошуку та оновлювати їх картки (Рис. 2.10)

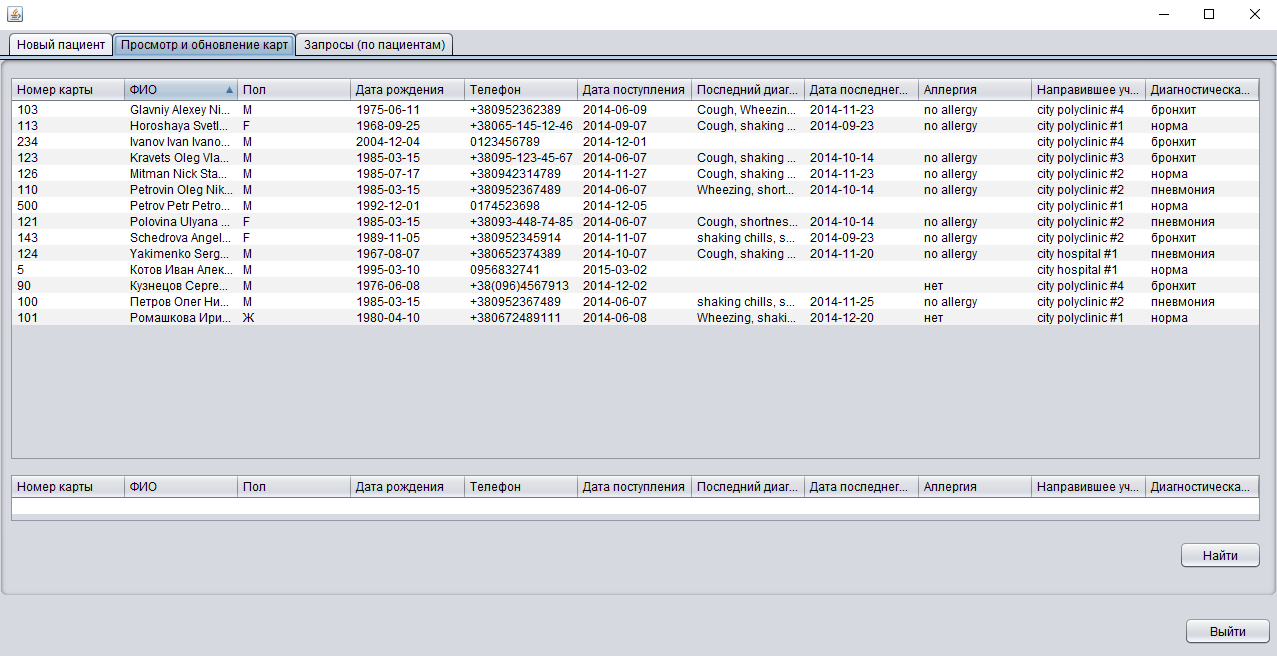


Рисунок 2.11 – Перегляд усіх карток пацієнтів

Працює на Windows XP і вище (перевірено на них, повинно працювати і на інших ОС), використовується jar-файли і бібліотека FANN, необхідний клієнт СУБД MySql.

# 3 КОНСТРУЮВАННЯ ПРОГРАМНОГО ПРОДУКТУ

## 3.1 Опис програмних технологій

Для створення ПО були використані наступні програмні технології: IDE Netbeans (мова програмування - Java), СУБД MySql2.

## 3.2 Опис програмних бібліотек

Були використані наступні програмні бібліотеки: java.awt. \* - Для промальовування елементів інтерфейсу і прослуховування зовнішніх сигналів, java.io. \* - введення / виведення для дебагу, java.sql. \* - Для виконання запитів до БД, java.util . \* - для використання структур даних, javax.swing. \* - для промальовування більш складних елементів інтерфейсу, javax.mail. \* - для роботи з відправкою інформації по email.

Для побудови нейронних мереж було використано бібліотеку FANN, а також JNI АРІ для підключення програми на С

## 3.3 Особливості створення програмних модулів з урахуванням мови програмування

В даному проекті кожен програмний модуль представлений окремим файлом у вигляді класу. Зв'язок між даними файлами (класами) забезпечений за допомогою import (підключення бібліотек Java і внутрішніх файлів для видимості). Так як Java є повністю об'єктно-орієнтованою, то всюди притаманні класи.

## 3.4 Особливості створення структур даних

Основні структури даних являють собою залежності у вагахкласифікаторів. Данные зависимости аггрегируются из тренировочных данных в процессе обучения нейронных сетей.

У програмному коді були використані також такі структури даних як масиви, списки, словники (Map, HashMap).

## 3.5 Модульне тестування

Оскільки нейронні мережі не мають людиночитаємого звіту щодо їхної внутрішньої структури, будемо користуватися методом чорного ящика.

Модель: Використання єдиної нейронної мережі

На роль першого класифікатора було вирішено обрати простий багатошаровий перцептрон Розенблатта завдяки простоті реалізації та наявності багатьох програмних бібліотек, що реалізують необхідний функціонал.

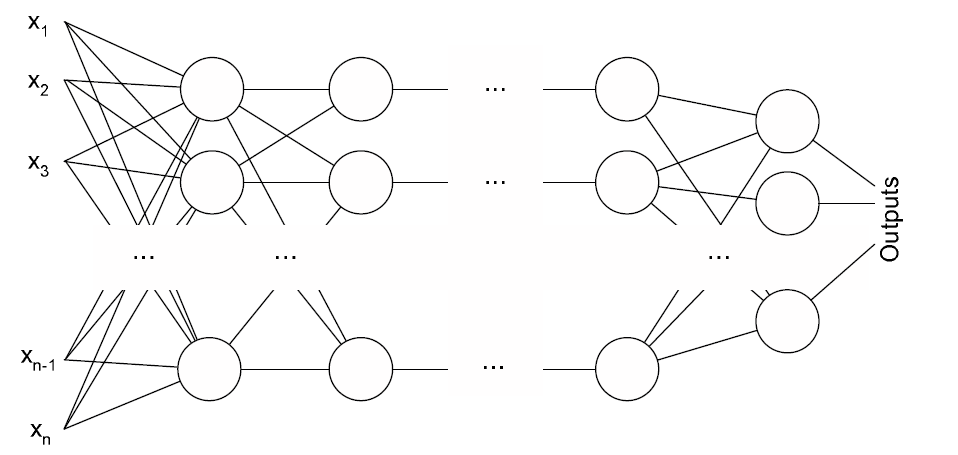


Рисунок 3.1 – Багатошаровий перцептрон

Для реалізації було обрано бібліотека FANN (Fast Artificial Neural Network), що спеціалізована на побудові багатошарових нейронних мереж. Оскільки вона написана на мові програмування С, є великий потенціал для портування на різні мови. Ця можливість була використана для біндингу з Java-кодом системи DiaSpectrEx для участі в конкурсі IT-Еврика 2014.

Була обрана повнозв’язкова топологія з кількістю вхідних нейронів, що відповідають кількості ознак в векторі даних та кількістю вихідних нейронів, що відповідають кількості класів. Кількість проміжних шарів та оптимальні функції активації були обрані жадібним алгоритмом на малій частині вихідного набору даних. Класи кодуються таким чином, щоб на виході найбільш ймовірного класу була логічна «1», а на всіх інших виходах – логічний «0».

Для використання перцептрону всі наявні дані були нормалізовані та подані на вхід мережі для її навчання. Також було проведено декілька серій навчання з різними налаштуваннями алгоритмів, та обрано оптимальний з результатів.

## 3.5.1 Тестування моделі

Для тестування обраної технології та її якості навчання на основі набору тренувальних даних було сформовано набір тестових даних за наступним принципом:

https://latex.codecogs.com/gif.latex?v%27_i%20%3D%20v_i%20&plus;%20%5Cdelta_i%2C%20%5Cdelta_i%20%5Cin%20%5B-v_i%5Cdiv25%2C%20v_i%5Cdiv25%5D (9)

де https://latex.codecogs.com/gif.latex?v%27_i - тестовий вектор, https://latex.codecogs.com/gif.latex?v_i - вектор з тренувального набору

Тестування було проведене запуском нейронної мережі на тестових даних з аналізом співвідношення кількості правильно розпізнаних векторів та неправильно розпізнаних векторів

Точність класифікації на тестових векторах складає **від 49.92% до 73.11%** в залежності від тестових даних.

Означений підхід мав декілька суттєвих недоліків, що вимусило відмовитися від подальшої його реалізації. Серед них:

1. Неможливість додавати класи до моделі без перенавчання всієї нейронної мережі;
2. Низька точність класифікації, що неприпустимо в медичних системах;
3. Стабільність моделі залежить від багатьох налаштувань, які нетривіально вибрати оптимально та які залежать від характеристик тренувальної вибірки;
4. Немає можливості детектувати, що клас не належить до жодного з відомих моделі
5. Висока ймовірність перенавчання нейронної мережі через її складність.

## Модель: Використання декількох нейронних мереж

Для того, щоб позбавитися від проблем єдиної нейронної мережі, було вирішено використати декілька менших нейронних мереж у ролі класифікаторів типу «один-проти-всіх».

Кожний з класифікаторів є малим перцептроном, що має 50% зв’язків між нейронами, кількість входів, що відповідає кількості ознак, та лише одному виходу. Таким чином формується N класифікаторів, де N – кількість класів. Кожний класифікатор може відрізнити «свій» клас від усіх інших.

Використовуючи такий підхід з’являється можливість суттєво зменшити кожний з малих класифікаторів, порівняно з одним «великим», особливо у разі великої кількості даних. Класифікатори навчаються за допомогою стандартного інкрементального алгоритму на спеціально адаптованих даних, кожний з яких являє собою файл, в якому дані потрібного класу промарковані символом «1», а дані усіх інших класів – символом «0».

Після побудови класифікаторів вони зорганізуються у глибоке дерево прийняття рішень за допомогою жадібного алгоритму:

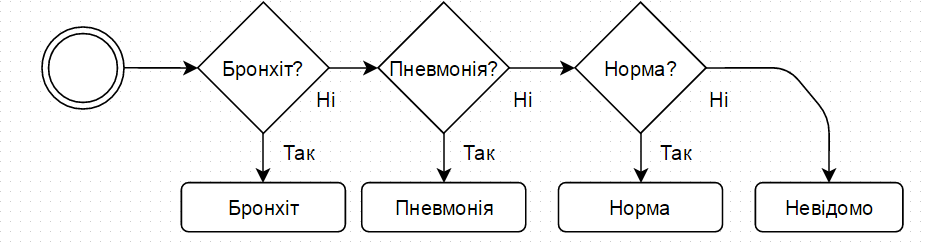


Рисунок 3.2 – Примітивне глибоке дерево прийняття рішень

Оскільки кожний з підкласифікаторів представляє собою окрему нейронну мережу, для нього можуть бути використані індивідуальні налаштування, що дозволять виокремлювати «свій» клас найбільш ефективно.

Алгоритм побудови являє собою повний перебір можливих формацій дерева з подальшим тестуванням отриманого дерева на кожній ітерації перестановок блоків-класифікаторів. Використана стратегія «розподіляй і володарюй» допомогла в суттєвому зменшенні складності кожного з класифікаторів та всієї моделі в цілому.

## 3.5.2 Тестування моделі

Модель добре проявила свої властивості на тестових даних, що були згенеровані так само, як і дані для тестування єдиної мережі (9). Точність розпізнавання складає **94.83%** на тестових даних. Для сучасних медичних досліджень це є досить високим результатом, оскільки ставить технологію на один рівень точності з такими неінвазійними методами діагностики як рентген та флюороскопія.

Однак, не зважаючи на високу точність алгоритму, він має декілька суттєвих недоліків.

З боку навчання:

* Висока обчислювальна складність повного перебору, що унеможливить побудову моделі для великої кількості класів, або для великої кількості даних в класах.
* Неможливість коригування окремих підкласифікаторів без повторного навчання усіх підкласифікаторів моделі.

З боку виконання:

* Неможливість додавання нових класів при роботі алгоритму.
* Нестабільність моделі: побудова моделі залежить від якості навчання кожного з обраних класифікаторів.

Оскільки на даний момент кількість класів досить мала через невелику кількість клінічних досліджень, алгоритм, що використовує дерево рішень є оптимальним с точки зору співвідношення «витрати – якість розпізнавання».

# 4 ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

## Функціональне тестування

На табл. 4.1 – 4.13 зображені усі варіанти для функціонального тестування (тестування варіантів використання). Test cases розписані для першого варіанту використання (як приклад). Спеціалісту з QA достатньо подивитися на таблицю та покрити тестами усі стовпці в кінцевому вигляді тестування.

ВВ «Registration of medicals»

Передумова: Програма запущена, є права для реєстрації у профілі.

1. Запуск режиму реєстрації
2. Адміністратор вводить дані
3. Перевірка правильності даних

3а. Дані невірні, перехід до п. 2

1. Збереження даних в DataStorage

Таблиця 4.1 – Покриття умов ВВ «Реєстрація медпрацівника»

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ВВ1 – реєстрація медпрацівника | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Ум.1 –  є права для реєстрації | F | F | F | F | T | T | T | T |
| Ум.2 – дані надані адміністратору | F | F | T | T | F | F | T | T |
| Ум.3 – дані вірні | F | T | F | T | F | T | F | T |
| Д1 – запуск режиму реєстрації |  |  |  |  | ok | ok | ok | ok |
| Д2 – адміністратор вводить дані |  |  |  |  |  |  | ok | ok |
| Д3 – виправлення даних |  |  |  |  |  |  | ok |  |
| Д4 – збереження даних в БД |  |  |  |  |  |  |  | ok |

Таблиця 4.2 – Покриття умов ВВ «Реєстрація медпрацівника» (скорочене)

|  |  |  |  |
| --- | --- | --- | --- |
| ВВ1 – реєстрація медпрацівника | 1 | 2 | 3 |
| Ум.1 –  є права для реєстрації | T | T | T |
| Ум.2 – є дані | F | T | T |
| Ум.3 – дані вірні | X | F | T |
| Д1 – запуск режиму реєстрації | ok | ok | ok |
| Д2 – адміністратор вводить дані |  | ok | ok |
| Д3 – виправлення даних |  | ok |  |
| Д4 – збереження даних в БД |  |  | ok |

Test cases:

Права – адміністратор, нема даних. Результат – Запуск режиму реєстрації.

Права – адміністратор, дані є: ПІБ (англійською), інші поля заповнені вірно. Результат – необхідно виправити дані в ПІБ.

Права – адміністратор, усі умові виконані вірно (всі поля заповнені коректно: ПІБ – кирилиця, без зайвих символів, правильний формат номеру телефону та інше). Результат – введення та збереження даних у БД.

ВВ «Authorization»

Передумова: Запущений режим «Авторизація»

1. Користувач вводить логін та пароль
2. Перевірка правильності введених даних в DataStorage

2а. Невірний логін або пароль. Перехід до п.1

1. Вхід до головного меню програми

Таблиця 4.3 – Покриття умов ВВ «Авторизація»

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ВВ2 – авторизація | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Ум.1 –  вірний логін | F | F | F | F | T | T | T | T |
| Ум.2 – вірний пароль | F | F | T | T | F | F | T | T |
| Ум.3 – вірний ключ до БД | F | T | F | T | F | T | F | T |
| Д1 – повідомлення про невірний логін |  | Ok |  | Ok |  |  |  |  |
| Д2 – повідомлення про невірний |  |  |  |  |  | Ok |  |  |
| Д3 – повідомлення про невірний ключі до БД | Ok |  | Ok |  | Ok |  | Ok |  |
| Д4 – Вхід в систему |  |  |  |  |  |  |  | Ok |

Таблиця 4.4 – Покриття умов ВВ «Авторизація» (скорочене)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ВВ2 – авторизація | 1 | 2 | 3 | 4 |
| Ум.1 –  вірний логін | T | X | T | F |
| Ум.2 – вірний пароль | T | X | F | X |
| Ум.3 – вірний ключ до БД | T | F | T | T |
| Д1 – повідомлення |  |  |  | Ok |

Продовження таблиці 4.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Д2 – повідомлення про невірний пароль |  |  | Ok |  |
| Д3 – повідомлення про невірний ключі до БД |  | Ok |  |  |
| Д4 – Вхід в систему | Ok |  |  |  |

ВВ «Input CMiEA»

Передумова: Запущений режим «Сеанс»

1. Лікар вводить значення КВВП-даних
2. Система перевіряє коректність вводу

2а. Дані некоректні, виведення повідомлення про помилку на екран. Перехід до п.1

1. Збереження введених даних у MedCard

Таблиця 4.5 – Покриття умов ВВ «Введення КВВП-даних»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ВВ3 – введення КВВП-даних | 1 | 2 | 3 | 4 |
| Ум.1 –  дані коректні | F | F | T | T |
| Ум.2 – сума дорівнює 100 | F | T | F | T |
| Д1 – повідомлення про некоректні дані | Ok | Ok |  |  |
| Д2 – повідомлення про неприпустиму суму |  |  | Ok |  |
| Д3 – збереження даних |  |  |  | Ok |

Таблиця 4.6 – Покриття умов ВВ «Введення КВВП-даних» (скорочене)

|  |  |  |  |
| --- | --- | --- | --- |
| ВВ3 – введення КВВП-даних | 1 | 2 | 3 |
| Ум.1 –  дані коректні | F | T | T |
| Ум.2 – сума дорівнює 100 | X | F | T |
| Д1 – повідомлення про некоректні дані | Ok |  |  |
| Д2 – повідомлення про неприпустиму суму |  | Ok |  |
| Д3 – збереження даних |  |  | Ok |

ВВ «Send report via email»

Передумова: Адміністратор авторизувався. Сформований звіт про діагностування пацієнта

1. Адміністратор обирає необхідного пацієнта зі списку всіх пацієнтів

1а. Пацієнт не знайдений, перехід до п.1

1. Адміністратор обирає звіт зі списку звітів
2. Адміністратор відправляє обраний звіт на email пацієнта

Таблиця 4.7 – Покриття умов ВВ «Відправка звіту по email»

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ВВ4 – відправка звіту по email | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Ум.1 –  Наявність email пацієнта в базі | F | F | F | F | T | T | T | T |
| Ум.2 – наявність звіту в списку звітів | F | F | T | T | F | F | T | T |
| Ум.3 – підключення до Інтернету | F | T | F | T | F | T | F | T |

Продовження таблиці 4.7

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Д1 – повідомлення про відсутність email пацієнта | Ok | Ok | Ok | Ok |  |  |  |  |
| Д2 – повідомлення про відсутність звіту при пошуку |  |  |  |  | Ok | Ok |  |  |
| Д3 – повідомлення про те, що немає підключення до Інтернету |  |  |  |  |  |  | Ok |  |
| Д4 - Звіт успішно відправлений |  |  |  |  |  |  |  | Ok |

Таблиця 4.8 – Покриття умов ВВ «Відправка звіту по email» (скорочене)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ВВ4 – відправка звіту по email | 1 | 2 | 3 | 4 |
| Ум.1 –  Наявність email пацієнта в базі | F | T | T | T |
| Ум.2 – наявність звіту в списку звітів | X | F | T | T |
| Ум.3 – підключення до Інтернету | X | X | F | T |
| Д1 – повідомлення про відсутність email пацієнта | Ok |  |  |  |
| Д2 – повідомлення про відсутність звіту при пошуку |  | Ok |  |  |
| Д3 – повідомлення про те, що немає підключення до Інтернету |  |  | Ok |  |
| Д4 - Звіт успішно відправлений |  |  |  | Ok |

ВВ «Identify patient’s condition»

Передумова: Лікар обрав режим «Сеанс» та ввів КВВП-дані

1. Система виконує класифікацію введених даних

1а. Дані не належать жодному з відомих класів. Повернення помилки.

1. Система виводить передбачувану хворобу пацієнта

Таблиця 4.9 – Покриття умов ВВ «Визначення стану пацієнта»

|  |  |  |
| --- | --- | --- |
| ВВ5 – визначення стану пацієнта | 1 | 2 |
| Ум.1 –  Дані належать одному з класів | F | T |
| Д1 – повідомлення щодо відсутності приналежності до відомого класу | Ok |  |
| Д2 – виведення  передбачуваної хвороби |  | Ok |

ВВ «Query execution»

Передумова: Медпрацівник авторизувався

1. Медпрацівник обирає необхідний запит зі списку запитів
2. Вводить додаткові дані до запиту
3. Система виконує запит і виводить результат на екран

Таблиця 4.10 – Покриття умов ВВ «Виконання запитів»

|  |  |  |  |
| --- | --- | --- | --- |
| ВВ6 – виконання запитів | 1 | 2 | 3 |
| Ум.1 –  запит некоректний | X | F | T |
| Ум.2 – шуканий запит присутній в списку запитів | F | T | T |
| Д1 – додавання нового запиту | Ok |  |  |
| Д2 – повідомлення про некоректність запиту |  | Ok |  |
| Д3 – виконання запиту і виведення результату |  |  | Ok |

ВВ «Patient registration»

Передумова: Оператор авторизувався

1. Оператор обирає режим «Зареєструвати пацієнта»
2. Оператор вводить дані про пацієнта (ПІБ, телефон, паспорт, історія хвороб, алергія, дата народження, дата вступу на облік, організація, що направила)
3. Перевірка заповненості необхідних полей

3а. Необхідно заповнити деякі дані, перехід до п. 2

1. Перевірка коректності вводу

4а. Дані введені некоректно, перехід до п. 2

1. Збереження даних в MedCard

Таблиця 4.11 – Покриття умов ВВ «Реєстрація пацієнтів»

|  |  |  |  |
| --- | --- | --- | --- |
| ВВ7 – реєстрація пацієнтів | 1 | 2 | 3 |
| Ум.1 –  Заповнені всі необхідні поля | F | T | T |
| Ум.2 – коректність всіх заповнених полів | X | F | T |
| Д1 – повідомлення про необхідність заповнити поля | Ok |  |  |
| Д2 – повідомлення про некоректне введення в конкретному полі |  | Ok |  |
| Д3 – збереження даних |  |  | Ok |

ВВ «Input medical data»

Передумова: Створена мед карта пацієнта, у неї внесені обов’язкові дані про пацієнта

1. Оператор вводить медичні дані про пацієнта (передбачувана діагностична група, організація, що направила, алергія, історія хвороб)
2. Перевірка коректності введених даних

2а. Дані некоректні, перехід до п.1

1. Збереження даних у MedCard

Таблиця 4.12 – Покриття умов ВВ «Введення медичних даних»

|  |  |  |
| --- | --- | --- |
| ВВ8 – введення медичних даних | 1 | 2 |
| Ум.1 – коректність всіх заповнених полів | F | T |
| Д1 – повідомлення про некоректне введення в конкретному полі | Ok |  |
| Д2 – збереження даних |  | Ok |

ВВ «Update statistics for groups»

Передумова: Лікар авторизувався та обрав режим «Оновлення статистики»

1. Перевірка кількості даних у MedCard

1а. Недостатня кількість для оновлення, вивід повідомлення на екран.

1. Вивід оновленої статистики на екран.

Таблиця 4.13 – Покриття умов ВВ «Оновлення статистик для груп»

|  |  |  |
| --- | --- | --- |
| ВВ9 – оновлення статистик для груп | 1 | 2 |
| Ум.1 – кількість даних відповідає необхідному (задано константно) | F | T |
| Д1 – повідомлення про недостатню кількість даних | Ok |  |
| Д2 – вивід оновленої статистики |  | Ok |

Таблиця тестових варіантів

|  |  |  |
| --- | --- | --- |
| Подія | Умова T | Умова F |
| Ввод ПІБ | Строкові значення, що складаюсться з українських та англійських літер та типографських знаків, нек більше 60 символів в довжину.  Тестове значенне:  Вильгельмина Д’Абдурахмангаджи Христорождественская-Петренко | Числа, спеціальні символи, типографські символи  Тестове значенне:  Кон©тантин 1 Радомирович Тесленко~Павлов |
| Ввод КВВП-даних | Дробні значення в діапазоні від 0 до 100, не більше 4 знаків після коми. Десятковий розподільник -- кома  Тестове значення:  99,3544 | Строки, специальні символи  Тестове значенне:  Девяносто девять,3544 |
| Ввод e-mail | Тестове значенне:  [tim4bor@gmail.com](mailto:tim4bor@gmail.com) | Специальні символи  [tim~4bor@gmail.com](mailto:Tim~4bor@gmail.com)  [tim4bor@.com](mailto:Tim4bor@.com) |
| Ввод номера телефона | Строка, состояща з цифр та знака «+», довжиною 13 символів  Тестове значення:  +380664471908 | Буквенно-цифрові строки, специальні символи  Тестовое значение:  +38066ЧЧ71908  +3966445585  +369535465© |
| Ввод номера паспорта | Тестове значенне:  АА123456 | Специальні символи  Тестове значенне:  С©Ё!»№;% |
| Ввод історії хвороби | Текст  Тестове значенне:  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras imperdiet volutpat massa vel bibendum. Etiam viverra tristique lorem vel tempor. Donec magna neque, condimentum sit amet tincidunt venenatis, faucibus in risus. Nam felis dui, dapibus non viverra vel, lacinia commodo velit. Nam ac sollicitudin nisi. Class aptent taciti sociosqu ad. | Нетипографські символи |
| Ввод дати | Дата в форматі ДД-ММ-ГГГГ  Тестове значення:  23-12-1925 | Буквенні символи, специальні символи,  дата, більше за поточну, дата раніше за поточну бульше, ніж на 150 лет  Тестове значеня:  АЧ~44-9252  23-02-1800  23-02-2016 |

# 5 РОЗГОРТАННЯ ПРОГРАМНОГО ПРОДУКТУ

## 5.1 Інструкція з встановлення

Для роботи з програмою перед її установкою необхідно встановити:

1) Jdk 7 версії або вище

2) СУБД MySQL (створивши схему patient2 і запустивши дамп-файли sql (файли знаходяться на github))

3) jar-файл і fannfloat.dll необхідно помістити в одну папку

## 5.2 Інструкція з використання

1) Логіни / паролі для входу

Режим лікаря - d1 / d1

Режим оператора - op / op

Режим адміністратора - ad1 / ad1

2) Розглянемо роботу з найпершою версією програми (далі будуть виконані зміни, поки більша частина функцій робоча, але це ще прототип) на Рис. 5.1.

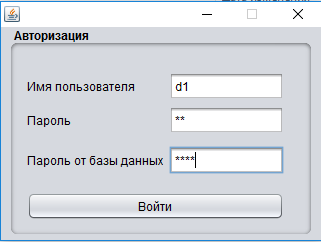


Рисунок 5.1 – Авторизація

Далі йде перехід в головне меню.

Для прикладу візьмемо існуючу карту під номером 100 (Рис. 5.2)

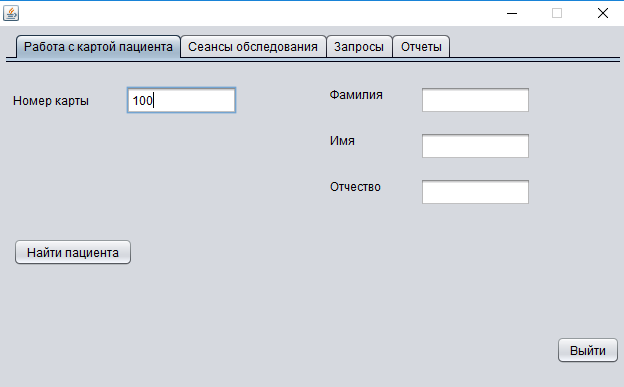


Рисунок 5.2 – Робота з картою пацієнта

Натискаємо «Знайти пацієнта» (Рис. 5.3)

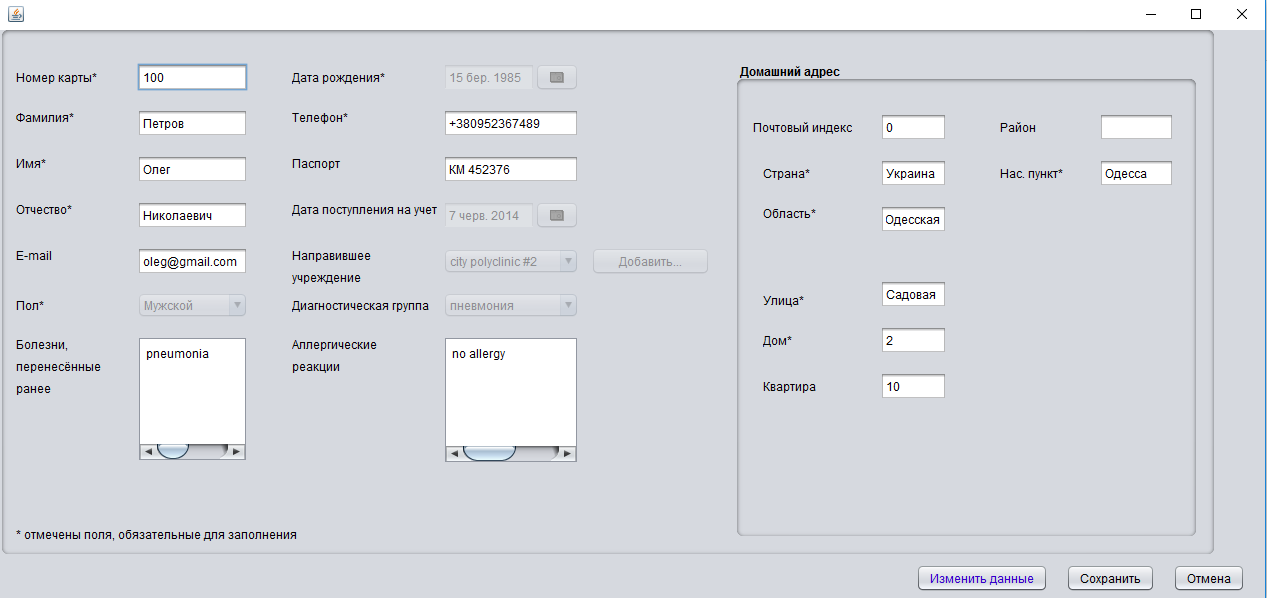


Рисунок 5.3 – Карта пацієнта

Тут можна змінити дані (треба натиснути синю кнопку, буде доступна зміна, після необхідно зберегти).

Натискаємо «Скасування», тепер дана карта є активною для лікаря на поточний момент.

Переходимо на вкладку «Сеанси обстеження», вибираємо «Перегляд сеансів», не створюємо новий сеанс, натиснувши кнопку «Обследование»(Рис. 5.4).

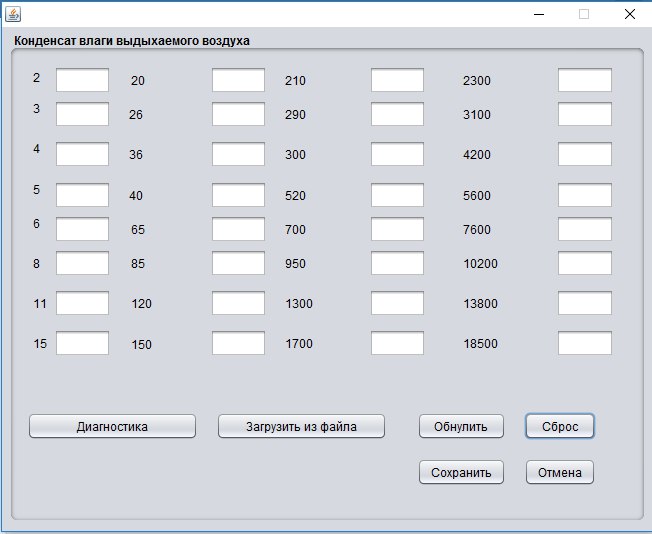


Рисунок 5.4 – Вікно вводу КВВП даних

Відкривається вікно, в якому можна ввести КВВП дані, а також обрати файл напряму зі спектрометра:

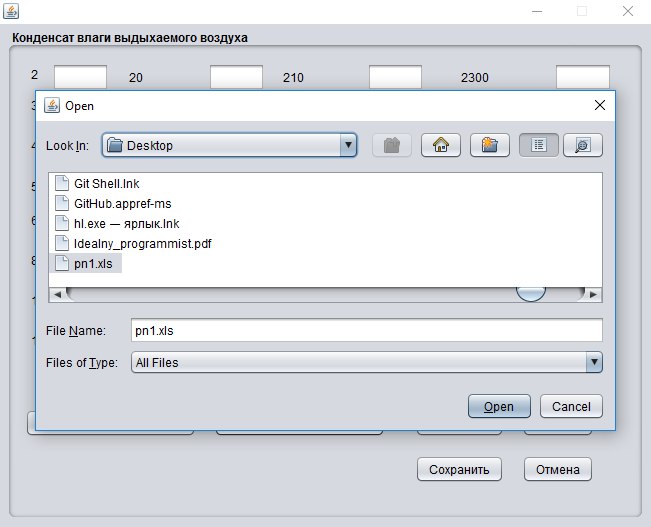


Рисунок 5.5 – Завантаження файлу зі спектрометра

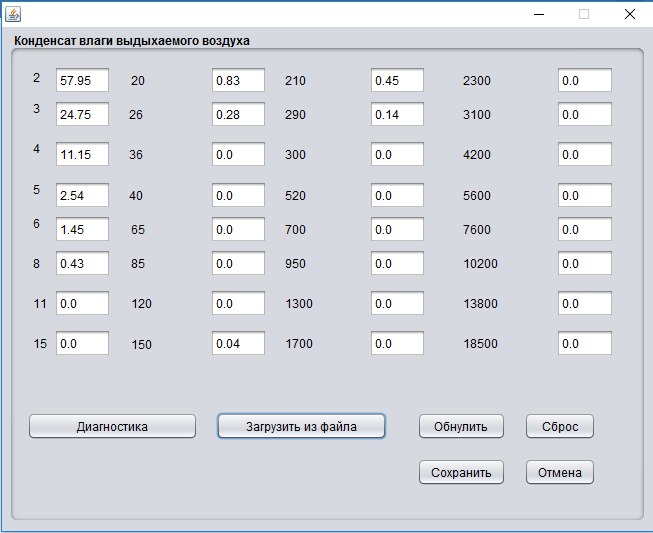


Рисунок 5.6 – Заповнена форма КВВП

Після вводу даних необхідно натиснути кнопку «діагностика», система класифікує введені дані та виведе найвірогідніший діагноз на екран:

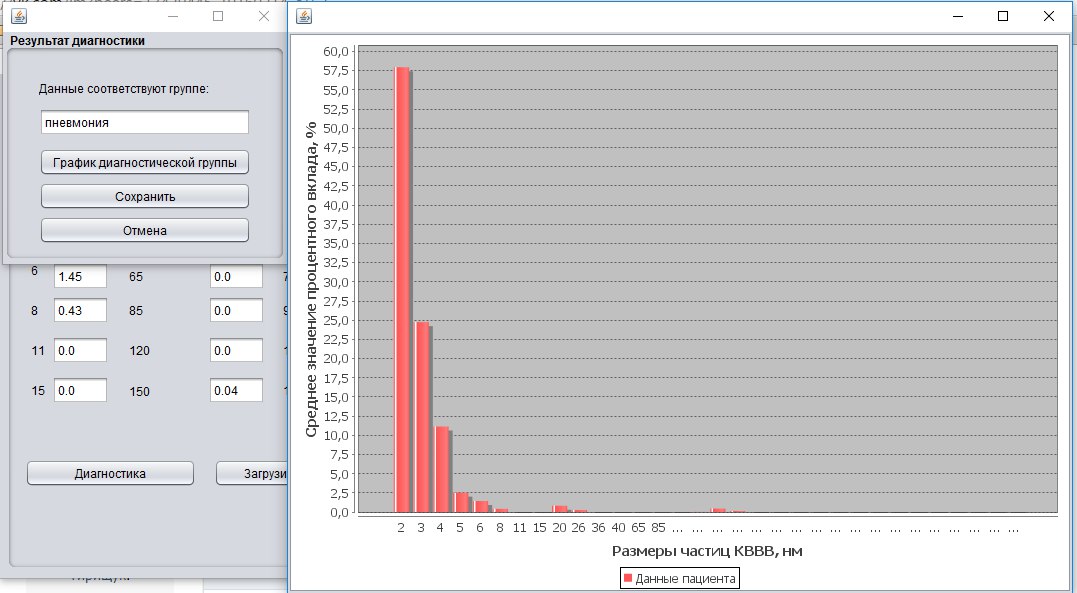


Рисунок 5.7 – Вірогідний діагноз

А також, середнє значення КВВП для цієї групи захворювань:

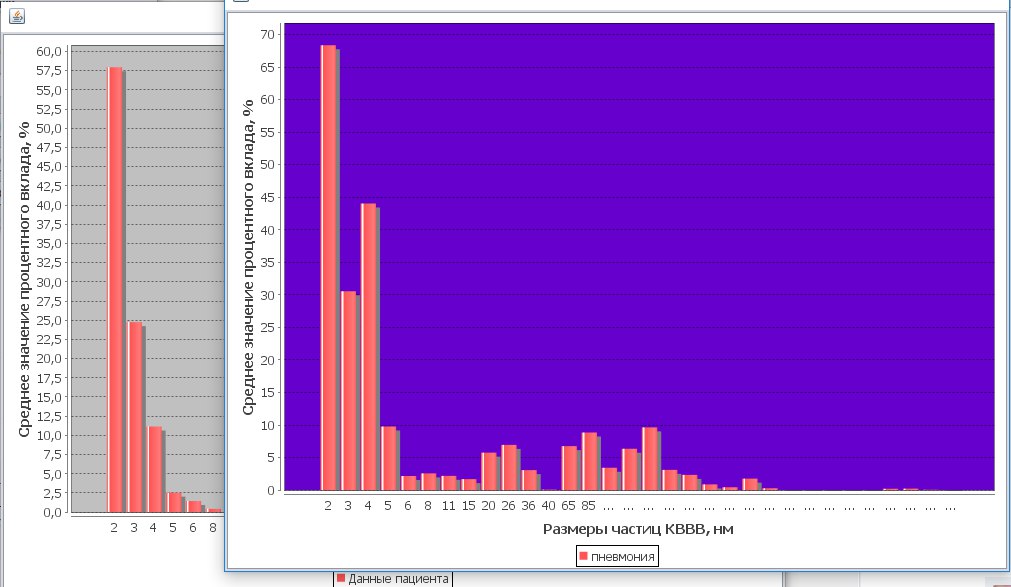


Рисунок 5.8 – Гістограми та середнє значення для групи

Відповідно до отриманих даних, лікар ставить діагноз пацієнтові.