МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.
«_____» _____2021 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: Програмний модуль системи підтримки прийняття рішень в умовах пандемії

Виконавець: Сокуренко В.А.

Керівник: Росінська Г.П.

Нормоконтролер: Тупота Є.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп’ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо–професійна програма «Системне програмування»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри
Литвиненко О.Є.
«____»_____2021 р.

ЗАВДАННЯ

на виконання дипломної роботи (проєкту)

Сокуренка Володимира Андрійовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проєкту) Програмний модуль системи підтримки прийняття рішень в умовах пандемії

затверджена наказом ректора від «04» лютого 2021 р. № 135/ст.

2. Термін виконання роботи (проєкту): з 17.05.2021 по 20.06.2021

3. Вихідні дані до роботи (проєкту): *Python*, середовище розробки *PyCharm*

4. Зміст пояснювальної записки:

1) опис предмету дослідження;

2) алгоритмічна розробка програмного додатку;

3) практична реалізація програмного модулю.

5. Перелік обов’язкового графічного (ілюстративного) матеріалу:

1) плакат «Отримання статі користувача»;

2) плакат «Отримання захворювань користувача»;

3) плакат «Отримання даних про симптоми користувача»;

4) плакат «Отримання даних про контакт з COVID–інфокаманими людьми»;

5) плакат «Отримання даних про додаткові симптоми користувача».

6. Календарний план–графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Провести аналіз літератури за темою дипломної роботи та аналіз аналогів на ринку	18.05.2021 - 20.05.2021	
2	Написати розділ 1	21.05.2021 - 22.05.2021	
3	Провести аналіз та розробку оптимальних алгоритмів роботи програмного модуля	23.05.2021 - 25.05.2021	
4	Написати розділ 2	25.05.2021 - 26.05.2021	
5	Розробити програмний модуль	26.05.2021 - 01.06.2021	
6	Провести тестування отриманого модуля	01.06.2021	
7	Написати розділ 3	01.06.2021 - 02.06.2021	
8	Оформити пояснювальну записку	02.06.2021 - 03.06.2021	
9	Підготувати графічний матеріал	04.06.2021	
10	Отримання допуску до захисту та подача роботи в ДЕК	09.06.2021	

7. Дата видачі завдання: «17» травня 2021 р.

Керівник дипломної роботи (проєкту) _____ Росінська Г.П.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Сокуренко В.А.
(підпис випускника) (П.І.Б)

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту «Програмний модуль»: 48 с., 10 рис., 15 літературних джерел, 1 додаток.

Ключові слова: ПРОГРАМНИЙ МОДУЛЬ, СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ, ПАНДЕМІЯ, *PYTHON*.

Об’єкт дослідження – система підтримки прийняття рішень в умовах пандемії.

Предмет дослідження – програмний модуль системи підтримки прийняття рішень в умовах пандемії.

Мета дослідження – створити програмний модуль системи підтримки прийняття рішень в умовах пандемії, який дасть рекомендації користувачу, щодо подальших дій в залежності від симптомів, віку, статі, інших захворювань та контактку з *COVID*-хворими.

Методи дослідження – теоретичне ознайомлення з існуючими технологіями, які використовують при розробці програмного модулю, мова програмування *Python*.

Результати дипломного проєктування полягають у зменшенні навантаження на систему охорони здоров'я, економії бюджету країни, через зменшення звернень до державних поліклінік.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ 6
ВСТУП 7
РОЗДІЛ 1 ОПИС ПРЕДМЕТУ ДОСЛІДЖЕНЬ 10
1.1. Поняття програмного модулю 10
1.2. Технічні засоби реалізації проєкту 11
1.3. *PyCharm* як середовище розробки 19
1.4. Висновки до розділу 22
РОЗДІЛ 2 АЛГОРИТМІЧНА РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ 23
2.1. Ідея проєкту 23
2.2. Технічне завдання до програмного модуля 24
2.3. Схеми алгоритму роботи функцій програмного модулю 24
2.4. Висновки до розділу 31
РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО МОДУЛЮ 32
3.1. Опис структури розроблювального програмного модулю 32
3.2. Вираховування коєфіцієнтів 33
3.3. Список можливих даних 35
3.4. Валідація даних 36
3.5. Надання рекомендації 37
3.6. Приклад використання 38
3.7. Висновки до розділу 44
ВИСНОВКИ 45
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ 47
ДОДАТОК А 49

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Python – інтерпретована об'єктно–орієнтована мова програмування високого рівня зі строгою динамічною типізацією.

Пандемія – вищий ступінь розвитку епідемічного процесу.

C++ – компільований, статично типізований мова програмування загального призначення.

float – широко поширений комп'ютерний формат подання дійсних чисел, що займає в пам'яті 32 *біта* (4 *байта*). Як правило, під ним розуміють формат числа з плаваючою комою стандарту *IEEE* 754.

IDE – комплекс програмних засобів, який використовується програмістами для розробки програмного забезпечення.

ПЗ – програмне забезпечення.

ВСТУП

Протягом всієї історії людство постійно стикається з пандеміями. Вони забирають тисячі людських життів, завдають колосальних економічних втрат і в одну мить знищують все, що було створено роками, десятиліттями і навіть століттями.

Як приклад, іспанський грип, або ж як йог ще називають – «іспанка», став справжньою медичної катастрофою. Пандемія торкнулася майже всі країни. В одній Великобританії померло 228 тисяч осіб, в Сполучених Штатах – 675 тисяч, в Японії – 400 тисяч. Дослідники підрахували, що в одній Індії загинули від 12 до 17 мільйонів людей. Їм заразилася третина населення землі, загинули 25 мільйонів людей, хоча деякі дослідники припускають, що вона досягала 45-50 мільйонів. Їм заразилася третина населення землі, загинули 25 мільйонів людей, хоча деякі дослідники припускають, що вона досягала 45-50 мільйонів. Така висока смертність пояснюється тим, що, якщо у населення не вироблений імунітет проти нового штаму вірусу. Іспанський грип показав, що з вірусними захворюваннями потрібно боротися на національному рівні. Починаючи з 1920-х років країни почали міняти систему охорони здоров'я. Багато країн створили або реорганізували міністерства охорони здоров'я, створили кращі системи по контролю захворювань і прийняли концепцію безкоштовної медицини. Ця пандемія показала: якби люди дотримувалися соціального дистанціювання і влада скасувала масові заходи, багатьох жертв можна було уникнути. Коли грип тільки почав поширюватися, в Філадельфії (США) влаштували парад на 200 тисяч чоловік в підтримку боротьби в Першій світовій війні. Через кілька днів в місті від іспанки померли 4 500 осіб. Влада Сент-Луїса (США) вчинили інакше: були закриті школи, театри, скасовані спортивні заходи. Церкви змушені були призупинити проведення недільних служб. В результаті кількість смертей в Сент-Луїсі було в кілька разів менше.

Одною із пандемій, з якою зіткнулося людство в 21 столітті є *COVID*-19, або ж коронавірус.

Спалах коронавирусной хвороби почалася в Ухані, сьомому за величиною місті Китаю з населенням більше 11 мільйонів чоловік. Це головний транспортний вузол в центральному Китаї, приблизно в 1000 *км* на південь від Пекіна, в 600 *км* на захід від Шанхаю і в 900 *км* на північ від Гонконгу. Прямі рейси пов'язували Ухань з країнами Європи (6 рейсів до Парижа, 3 в Лондон і 5 в Рим щотижня), Азію, Австралію, Північну Америку.

Вірус передається в основному при тісному контакті і повітряно-крапельним шляхом, коли людина кашляє або чхає. Людина може заразитися, доторкнувшись до інфікованої поверхні (дверна ручка, стіл), а потім обличчя (очей, носа, рота). Самою заразною є людина з симптомами, але також можливе зараження від людини без симптомів. Часовий інтервал між моментом зараження і появою перших симптомів зазвичай становить 5 днів, але може становити від 2 до 14 днів. Загальні симптоми включають жар, кашель і задишку. Ускладнення можуть включати пневмонію і гострий респіраторний дистрес-синдром.

За даними Міністерства охорони здоров’я та Ради національної безпеки та оборони, станом на 18 квітня 2021 року в Україні було підтверджено 1 946 510 випадків зараження *COVID*-19, з яких 39 786 людей загинули, 1 482 079 одужали. Станом на 18 квітня 2021 року Україна посідала 16 місце у світі за кількістю зафіксованих випадків зараження та вісімнадцяте у світі (восьме місце в Європі) – за кількістю смертей. Кількість заражених 1 000 000 людей становить 44 722 (шістдесят сьоме у світі та тридцять шосте в Європі), а кількість смертей від *COVID*-19 на мільйон населення становить 914.

Настільки тяжкий стан з пандемією в Україні, як і у всьому світі, виник через список причин, однією з яких є незнання людьми звичайних правил гігієни та нездатність системи охорони здоров’я надання консультацій, рекомендацій та лікування такій кількості людей.

Метою дипломної роботи є створення програмного модулю системи підтримки прийняття рішень в умовах пандемії, що буде надавати рекомендації користувачу щодо подальших дій в залежності від симптомів, статі, віку, хвороб

та контактку з *COVID*-хворими.

Об’єкт дослідження – система підтримки прийняття рішень.

Предмет дослідження – програмний модуль система підтримки прийняття рішень в умовах пандемії.

Методи дослідження – теоретичне ознайомлення з існуючими технологіями, які використовують при розробці програмного модуля, мова програмування *Python*.

Розроблений проєкт має наступні функції:

– валідація вхідних даних, таких як: стать, вік, захворювання користувача, основні симптоми, додаткові симптоми, навідних запитаннь, температури людини та контактку з *COVID*-хворими;

– вираховування рекомендації в залежності від вхідних даних;

– консольний додаток, який демонструє працю модуля;

Для досягнення поставленої мети необхідно вирішити наступні задачі:

– дослідити поняття програмного модулю;

– розглянути наявні аналоги та дослідити їх;

– вивчити технології необхідні для розробки програмного модулю, такі як: *Python*, модуль *decimal* для коректної роботи с *float*-числами, модуль *pandas* для обробки даних;

– розробити програмний модуль системи підтримки прийняття ріщень в умовах пандемії, що надасть користувачеві отримати рекомендацію щодо подальших дій в залежності від симптомів, статі, віку, хвороб та контакту з *COVID*–хворими;

– розробити консольний інтерфейс для демонстрації роботи програмного модулю.

РОЗДІЛ 1
ОПИС ПРЕДМЕТУ ДОСЛІДЖЕНЬ

1.1. Поняття програмного модулю

Програмний модуль - це функціонально завершена частина програми, зроблена у вигляді відокремого файлу з вихідним кодом або ж його іменованої частиною, призначена для експлуатації в інших програмах. Модулі дозволяють ділити складні завдання на більш легкі за принципом модульності. Зазвичай, програмний модуль призначений для надання програмістам декількох функцій.

Отже, модулі включають опис вихідних даних, операцій обробки даних і структури зв'язків з іншими програмними модулями. Бібліотека є окремим програмним продуктом, тобто кожен програмний модуль розробляється, компілюється і розглядається окремо від інакших програмних модулів. Таким чином, програмну бібліотеку можна розглядати як інстумент розділення на модулі складних програм, а також як засіб накопичення і повторного використання знань про програмне забезпечення. Розмір модуля вимірюється кількістю операторів в ньому. Розмір модулю повинен бути в межах розумного: не бути занадто маленьким або занадто великим. Надійність модуля є мірою його внутрішніх з'єднань. Надійність модуля залежить відз'єднань, які приховано від зовнішнього впливу програми.

1.2. Технічні засоби реалізації проєкту

У наступних розділах будут розглянуті наступні технології:

– мова програмування *Python*;

– бібліотека *decimal*;

– бібліотека *pandas*;

– інтегроване середовище розробки *PyCharm*.

1.2.1. *Python*

Python – високорівнева мова програмування загального призначення з динамічної строгою типізацією і автоматичним управлінням пам'яттю, орієнтований на підвищення продуктивності розробника, читання коду і його якості, а також на забезпечення переносимості написаних на ньому програм.

Гвідо ван Россум почав створювати мову програмування *Python* на прикінці 1980-х. Розподілена ОС *Amoeba* вимагала розширюваної мови сценаріїв, і Гвідо почав розробляти *Python* в вільний від праці час, взявши частину свого досвіду з *ABC*. У лютому 1991 року Гвідо опублікував вихідний код у групі новин *alt.sources*. Від початку *Python* розроблявся як об'єктно-орієнтована мова програмування.

Відмінні мовні характеристики:

– низький поріг вступу, після одного дня навчання можливо починати розробляти не складні програми;

– мінімалістична мова, з малим набором конструкцій;

– короткий код;

– ідеально йде для розробки програм-обгортки, підтримується підключення бібліотек *C*;

– є багато версій: *CPython* (основна реалізація) *Jython* (реалізація для *JVM*) *IronPython* (*CLR*) *PyPy*;

– високого рівня підтримка математичних обчислень (бібліотеки *NumPy*, *SciPy*);

– використовується для обробки природної мови (*NLTK*);

– велика кількість розроблених веб-фреймворків (*Django*, *TurboGear*, *CherryPy*, *Flask*).

Вплив інших мов програмування.

З'явившись пізно, *Python* зазнав впливу різних мов програмування:

– *ABC* – відступи для операторів групування, високорівневі структури даних (*Python* фактично створювався як спроба виправити помилки, допущені при проєктуванні *ABC*);

– *Modula*-3 – пакети, модулі, використовуючи *else* з *try* і *Osim*, названі аргументи функції (на це також впливає *Common Lisp*);

– *C/C++* – деякі синтаксичні структури;

– *Smalltalk* – об'єктно-орієнтоване програмування;

– *Lisp* – деякі особливості функціонального програмування (лямбда, карта, зменшення, фільтрація та інші);

– *Fortran* – зрізи масивів, складна арифметика;

– *Java* – протоколювання модулів, *unittest*, потокова передача (деякі можливості оригінального модуля не реалізовані), *xml.sax* стандартної бібліотеки, спільне використання, нарешті, за винятком обробки винятків, використання @ для декораторів;

– *Icon* – генератори;

– більшість інших функцій *Python* (наприклад, компіляція вихідних байтів) також раніше були реалізовані на інших мовах.

Причини використання *Python*.

– якість програмного забезпечення:

Для багатьох головною перевагою мови *Python* є його простота читання, ясність і більш високу якість, яке відрізняє його від інших інструментів в світі мов програмування. Код *Python* легше читати, що означає, що його набагато простіше повторно використовувати і підтримувати, ніж використовувати код на

інших мовах сценаріїв. Однаковість дизайну програмного коду на *Python* полегшує його розуміння навіть тим, хто не брав участі в його створенні. Крім того, *Python* підтримує сучасні механізми повторного використання коду, які представляють собою об'єктно-орієнтоване програмування (ООП);

– висока швидкість створення програм:

У порівнянні з компільованими або строго типізованими мовами, такими як *C/C++* і *Java, Python* має значно нижчий час розробки. Додатки на *Python* запускаються негайно, минаючи компіляцію і зв'язування, необхідні в інших мовах програмування, що ще більше підвищує продуктивність програміста;

– переносимість програм:

Більшість програм на *Python* працюють є кросплатформенними. Перенесення коду з *Linux* в *Windows* зазвичай являє собою просте копіювання програмних файлів з однієї машини на іншу. Крім того, *Python* надає масу можливостей для створення портативних графічних інтерфейсів, програм доступу до баз даних, веб-додатків і багатьох інших типів програм. Навіть інтерфейси операційних систем, включаючи спосіб запуску програм і обробку каталогів, переносимі в *Python*;

– підтримка бібліотек:

Python поставляється з великою кількістю зібраних та портативних функцій, відомих як стандартна бібліотека. Ця бібліотека надає безліч можливостей, необхідних додаткам, від зіставлення шаблонів до мережевих функцій. Крім того, *Python* розширюється як за допомогою власних бібліотек, так і за допомогою сторонніх бібліотек. Сторонні розробки включають інструменти для створення веб-сайтів, математичне програмування, доступ до послідовного порту, розробку ігор та багато іншого. Наприклад, розширення *NumPy* продається як безкоштовний і більш потужний еквівалент системи математичного програмування *Mathlab*;

– інтеграція компонентів:

Скрипти *Python* можуть легко взаємодіяти з іншими частинами програми за допомогою різних механізмів інтеграції. Ця інтеграція дозволяє використовувати *Python* для налаштування і розширення функціональності програмних продуктів.

Сьогодні програмний код на *Python* має можливість викликати функції з бібліотек на *C/C++*, сам викликатися з програм, написаних на *C/C++*, інтегруватися з програмними компонентами на *Java*, взаємодіяти з такими платформами, як *COM* і *.NET*, і взаємодіяти через послідовний порт або мережу з використанням таких протоколів, як *SOAP*, *XML-RPC* і *CORBA*;

– функціональність:

З точки зору функціональності *Python* можна назвати гібридом. Його інструменти варіюються від традиційних мов сценаріїв (таких як *Tcl*, *Scheme* і *Perl*) до мов розробки програмного забезпечення (таких як *C/C++* і *Java*). *Python* забезпечує простоту і легкість мови сценаріїв, а також міць, яку зазвичай можна знайти в скомпільованих мовах. Перевершуючи можливості інших мов сценаріїв, ця комбінація робить *Python* зручним інструментом для розробки великомасштабних проєктів. Нижче наведено список ключових функцій, доступних в арсеналі *Python*:

– динамічна типізація:

Python сам відстежує типи об'єктів, що використовуються в програмі, тому вам не потрібно писати довгі і складні оголошення в програмному коді. Насправді в *Python* відсутнє поняття типу і необхідність оголошувати змінні. Оскільки код *Python* не обмежений типами даних, він може автоматично обробляти цілий ряд об'єктів;

– автоматичне управління пам'яттю:

Python автоматично виділяє пам'ять об'єктам і звільняє її, коли об'єкти стають непотрібними. Більшість об'єктів можуть збільшувати і зменшувати обсяг використовуваної ними пам'яті в міру необхідності;

– модульне програмування:

для створення великих систем *Python* надає такі функції, як модулі, класи та винятки. Вони дозволяють розбивати систему на компоненти, використовувати ООП для створення багаторазового програмного коду і елегантно обробляти виникаючі події і помилки;

– вбудовані типи об'єктів:

Python надає найпоширеніші структури даних, такі як списки, словники та рядки, як функції, специфічні для самої мови програмування. Ці типи відрізняються високою гнучкістю і зручністю. Наприклад, вбудовані об'єкти можуть бути розширені і стиснуті в міру необхідності, можуть бути об'єднані один з одним для представлення даних зі складною структурою;

– вбудовані інструменти:

Python має потужні і стандартні інструменти для роботи з усіма цими типами об'єктів, включаючи такі операції, як об'єднання (об'єднання колекцій), Витяг фрагментів (Витяг частини колекції), сортування, зіставлення і багато іншого.

Утиліти сторонніх розробників.

– *Python* – це відкритий програмний продукт і тому розробники можуть створювати свої попередньо скомпільовані інструменти підтримки завдань, вирішити які внутрішніми засобами неможливо;

– перевага *Python* перед іншими мовами високого рівня;

– має більше функцій, ніж *Tcl*. *Python* підтримує "програмування в цілому", що робить його придатним для розробки великих систем;

– має більш чіткий синтаксис і більш просту архітектуру, ніж *Perl*, що робить програмний код більш легким для читання, простим в обслуговуванні і знижує ймовірність помилок;

– простіше і зручніше, ніж *Java*. *Python* – це мова сценаріїв, а *Java* успадкувала складний синтаксис від мов програмування, таких як *C++*;

– простіше і зручніше, ніж *C++*, але часто він не може конкурувати з *C++*, тому що, як мова сценаріїв, *Python* призначений для вирішення інших завдань;

– більш потужний і більш портативний, ніж *Visual Basic*. Відкритий характер *Python* також означає, що немає жодної компанії, яка б його контролювала;

– більш зрозумілий і універсальний, ніж *PHP*. Іноді *Python* використовується для створення веб-сайтів, але він може вирішувати набагато ширший спектр завдань, від управління роботами до створення анімаційних фільмів;

– більш зрілий і має більш чіткий синтаксис, ніж *Ruby*. На відміну від *Ruby* і *Java*, об'єктно-орієнтований стиль програмування є необов'язковим в *Python* – він не вимагає використання ООП в проєктах, де цей стиль не підходить;

– має динамічні функції таких мов, як *SmallTalk* і *Lisp*, але має більш простий і традиційний синтаксис, доступний як розробникам, так і кінцевим користувачам настроюються систем.

1.2.2. *decimal*

decimal – це *Python* модуль, що забезпечує підтримку швидкої правильно округленої десяткової арифметики з плаваючою комою. Він пропонує кілька переваг перед *float*:

– *decimal* "заснований на моделі з плаваючою точкою, яка була розроблена з урахуванням людей і обов'язково має головний керівний принцип – комп'ютери повинні забезпечувати арифметику, яка працює так само, як арифметика, яку люди вивчають у школі". – витяг із специфікації *decimal*;

– десяткові числа можна представити точно. На відміну від них, числа, такі як 1.1 та 2.2, не мають точних подань у двійковій системі з плаваючою комою. Кінцеві користувачі зазвичай не очікують, що 1.1 + 2.2 відображатиметься як 3.3000000000000003, як це робиться з *float*;

– точність переходить в арифметику. У десятковій плаваючій точці 0,1 + 0,1 + 0,1 – 0,3 точно дорівнює нулю. У двійковій системі з плаваючою комою результат – 5.5511151231257827*e*-17. Хоча майже до нуля, але різниці можуть накопичуватися. З цієї причини десятковий знак є кращим у бухгалтерських програмах, які мають суворі інваріанти рівності;

– *decimal* включає поняття значущих місць так, що 1,30 + 1,20 дорівнює 2,50. Нульовий результат зберігається, щоб вказати значимість;

– на відміну від апаратного заснованого двійкового числа з плаваючою комою, десятковий модуль має точність зміни користувача (за замовчуванням до 28 значень);

– і двійкові, і десяткові числа з плаваючою комою реалізовані з точки зору опублікованих стандартів. Хоча вбудований *float* надає лише незначну частину своїх можливостей, *decimal* надає всі необхідні частини стандарту. За потреби програміст має повний контроль над округленням та обробкою сигналів. Сюди входить можливість накласти точну арифметику, використовуючи винятки для блокування будь–яких неточних операцій;

– *decimal* був розроблений для підтримки "без упередження як точної некруглої десяткової арифметики (яку іноді називають арифметикою з фіксованою крапкою), так і арифметики із заокругленою плаваючою комою". витяг із десяткової арифметичної специфікації.

1.2.3. *pandas*

pandas – програмний модуль мовою програмування *Python* для обробки і аналізу даних. Робота *pandas* з даними базується на бібліотеці *NumPy*, що є інструментом нижчого рівня. Надає особливі структури даних і операції для роботи з числовими таблицями і тимчасовими рядами. Назва бібліотеки походить від економетричного терміна «панельні дані», використовуваного для опису багатовимірних структурованих наборів інформації. *pandas* поширюється під новою ліцензією *BSD*.

Історія.

Розробка пакету розпочато в 2008 році співробітником *AQR Capital Management* Уесом Маккіні. Перед відходом з *AQR* йому вдалося переконати керівництво дозволити опублікувати вихідний код бібліотеки під вільною ліцензією.

Інший працівник *AQR* – Чан Ше – приєднався до проєкту в 2012 році, ставши другим головним розробником бібліотеки. Приблизно в той же час бібліотека набрала популярність в середовищі *Python*-розробників, і до проєкту приєдналася велика кількість нових учасників.

Область застосування.

Основна область застосування – забезпечення роботи в рамках середовища *Python* не тільки для збору і очищення даних, але для задач аналізу та моделювання даних, без перемикання на більш специфічні для статобработкі мови.

Також активно ведеться робота по реалізації «рідних» категоріальних типів даних.

Бібліотека перш за все призначений для обробки і оцінки даних за загальними ознаками, наприклад середнім значенням, статистичними пакетом він в повному розумінні не є, однак набори даних типів *DataFrame* і *Series* застосовуються в якості вхідних в більшій кількості бібліотек аналізу даних і машинного навчання.

Основні можливості бібліотеки:

– об'єкт *DataFrame* для роботи з індексованими масивами двовимірних даних;

– засоби для обміну даними між структурами в пам'яті і файлами різних форматів;

– засоби поєднання даних і способи обробки відсутньої інформації;

– зміна формату наборів даних, в тому числі створення зведених таблиць;

– зріз даних за значеннями індексу, розширені можливості індексування, вибірка з великих наборів даних;

– вставка і видалення стовпців даних;

– можливості групування надають можливість виконувати трьохетапні операції типу «поділ, зміна, об'єднання»;

– злиття і об'єднання наборів даних;

– ієрархічне індексування дозволяє працювати з даними високої розмірності в структурах меншої розмірності;

– робота з тимчасовими рядами: формування тимчасових періодів і зміна інтервалів і так далі.

Бібліотека розроблена для високої продуктивності, самі важливі частини коду написані на *Cython* і *C*.

1.3. *PyCharm* як середовище розробки

PyCharm – інтегроване середовище розробки для мови програмування *Python*. Надає засоби для аналізу коду, графічний відладчик, інструмент для запуску юніт-тестів і підтримує веб-розробку на *Django*. *PyCharm* розроблена компанією *JetBrains* на основі *IntelliJ IDEA*.

PyCharm – це крос-платформна середовище розробки, яка сумісна з *Windows, macOS, Linux*.

Можливості *PyCharm*:

– допомога при написанні коду:

PyCharm робить розробку максимально продуктивною завдяки функціям автодоповнення і аналізу коду, миттєвої підсвічуванні помилок і швидких виправлень. Автоматичні рефакторингом допомагають ефективно редагувати код, а зручна навігація дозволяє миттєво переміщатися по проєкту;

– допомога при редагуванні:

Розумний редактор *PyCharm* призначений для максимально продуктивною розробки на *Python*, *JavaScript*, *CoffeeScript*, *TypeScript*, *CSS* і популярних мовах шаблонів. Функції автодоповнення, виявлення помилок і швидкі виправлення враховують особливості кожного з підтримуваних мов;

– зручна навігація:

Розумний пошук дозволяє швидко перейти до будь-якого класу, файлу або символу, а також на відповідне вікно або дії *IDE*. Перехід до вищестоящого методу, тесту, оголошенню, входженню або реалізації здійснюється в одне натискання;

– швидкі та безпечні рафакторінги:

PyCharm надає широкі можливості реорганізації коду за допомогою рефакторингов *Rename* і *Delete*, *Extract Method*, *Introduce Variable*, *Inline Variable*, *Inline Method* і багатьох інших. Рефакторингом враховують особливості конкретного мови або фреймворка, допомагаючи вносити зміни по всьому проєкту;

– вбудовані інструменти для розробників:

PyCharm пропонує великий набір інструментів з коробки: вбудований відладчик і інструмент запуску тестів, профілювальник *Python*, повнофункціональний вбудований термінал, інструменти для роботи з базами даних. *IDE* інтегрована з популярними системами контролю версій, містить вбудований *SSH*-термінал, підтримує можливості віддаленої розробки та віддалені інтерпретатори, а також інтеграцію з *Docker* і *Vagrant*;

– налагодження, тестування і профілювання:

Використовуйте візуальний відладчик *Python* і *JavaScript*. Створюйте і запускайте тести, використовуючи функції розумного редагування, і переглядайте звіти про запуск в зручному графічному інтерфейсі. Повністю контролюйте свій код завдяки інтеграції з профілювальником *Python*. Повністю контролюйте свій код завдяки інтеграції з профілювальником *Python*;

– контроль версій, розгортання і віддалена розробка:

Використовуйте універсальний інтерфейс для роботи з *Git*, *SVN*, *Mercurial* і іншими системами контролю версій. Запускайте і налагоджувати код на віддаленій машині. Налаштуйте автоматичне розгортання на віддалений хост або віртуальній машині і керуйте інфраструктурою за допомогою *Vagrant* і *Docker*;

– інструменти для роботи з базами даних:

Доступ до *Oracle*, *SQL Server*, *PostgreSQL*, *MySQL* та інших баз даних здійснюється прямо з *IDE*. PyCharm допомагає редагувати *SQL*-код, виконувати запити, переглядати дані і змінювати схеми;

– інструменти для наукових обчислень:

З *PyCharm* ви зможете працювати з ноутбуками *Jupyter*, запускати команди в інтерактивній консолі *Python*, підключати бібліотеки *Anaconda*, а також працювати з іншими бібліотеками для наукових обчислень і аналізу даних, включаючи *Matplotlib* і *NumPy*;

– інтерактивна консоль для *Python*:

Ви можете запустити консоль *REPL* для *Python*, яка має багато переваг над стандартною консоллю. Серед них перевірка синтаксису на льоту за допомогою інспекцій, зіставлення дужок і лапок і, звичайно, автодоповнення;

– підтримка наукових бібліотек:

PyCharm підтримує *Pandas*, *Numpy*, *Matplotlib* і інші бібліотеки для наукових обчислень. *IDE* забезпечує розумне редагування, дозволяє переглядати набори даних у вигляді графіків і в табличній формі;

– інтеграція з *Conda*:

Створюйте окреме оточення *Conda* і інсталюйте тільки потрібні бібліотеки для кожного проєкту. *PyCharm* дозволяє легко створювати і вибирати правильний оточення;

– настроюється крос-платформна *IDE*:

PyCharm можна встановити на *Windows*, *macOS* і *Linux* за допомогою одного ліцензійного ключа. Налаштуйте робоче середовище так, як вам подобається: виберіть відповідну колірну схему і зручні поєднання клавіш, включіть режим емуляції *VIM*;

– налаштовується інтерфейс:

Який розробник не любить налаштовувати інструменти під свої переваги? Ми таких ще не зустрічали, тому зробили інтерфейс нашої *IDE* максимально адаптованим під потреби користувача. Налаштуйте робоче середовище так, як вам подобається: виберіть відповідну колірну схему і зручні поєднання клавіш;

– модулі:

За роки розробки платформи *IntelliJ* було створено більше 50 плагінів для *PyCharm*, які забезпечують підтримку додаткових систем контролю версій, інтеграцію з інструментами і фреймворками, а також розширюють можливості редактора, наприклад за рахунок емуляції *Vim*;

– крос-платформна *IDE*:

PyCharm можна встановити на декількох комп'ютерах в операційних системах *Windows*, *Mac OS* або *Linux* – всі можливості *IDE* будуть доступні на будь-якому з них;

1.4. Висновки до розділу

В даному розділі було розглянуто основні технології, що доцільно використовувати при розробці програмного модулю, а саме:

– мова програмування – *Python*;

– бібліотека *pandas*;

– бібліотека *decimal*;

– інтегроване середовище розробки *PyCharm*.

Функціонально закінчений фрагмент програми, оформлений у вигляді окремого файлу з сирцевим кодом, називається програмним модулем.

Модулі дозволяють розбивати складні задачі на менші відповідно до принципу модульності. Зазвичай проєктується таким чином, щоб надати програмістам зручну для багаторазового використання функціональність.

Python є однією із мов програмування, що дозволяє створувати складні системи за допомогою програмних модулів.

PyCharm допомагає у створенні правильно вистроеного за логікою програмного модуля.

РОЗДІЛ 2
АЛГОРИТМІЧНА РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

2.1. Ідея проєкту

За основу ідеї було взято таке явище як он-лайн експертні медичні системи. Воно полягає у забезпеченні оперативної та систематичної допомоги медичному персоналу у вирішенні проблемних ситуацій та прийнятті рішень з питань лікування пацієнтів. Експертні системи будуються на базі законів, регламентів, національних рекомендацій і правил, прийнятих в медичній установі. Схожа ідея виникла і у мене, так як зіткнувся з пандемією та захворюванням *COVID*-19. З такою думкою я почав шукати сервіси, що надають можливість отримати рекомендацію в залежності від симптомів. Провівши доступний аналіз, я виявив, що така ідея ще не була реалізована українською мовою, то я вирішив її реалізувати.

Для зручності використання я вирішив зробити програмний модуль максимально простим. Для отримання рекомендації потрібно лиш передати в головну функцію такі параметри, як: стать, вік, захворювання користувача, основні симптоми, додаткові симптоми, навідних запитаннь, температури людини та контактку з *COVID*-хворими.

Для того щоб в подальшому можливо було модифікувати модуль, при розробці я продумав основу модуля та за допомогою можливостей *Python* створив легко розширюваний модуль.

На сьогоднішній день зареєстрованих хворих в Україні налічується більше 2 000 000. Кожного дня фіксується не менш 2 000 хворих. Це колосальне навантаження на систему охорони здоров’я. Звичайна людина, яка погано себе почуває, але не має *COVID*-19 при відвідуванні лікарні лише підвищує навантаження на перенавантажену систему охорони здоров’я та ще має ризик захворіти *COVID*-19. За для уникнення цією ситуації я розробив програмний модуль підтримки прийняття рішень в умовах пандемії.

Альтернативи програмного модулю системи підтримки прийняття рішень в умовах пандемії не існує, але існують сервіси, які надають рекомендації, наприклад, такі як: *https://www.sberbank.ru/promo/eco/covid/,* що за допомогою моделі ІІ, побудована на світових даних з дослідження коронавируса надає рекомендації.

2.2. Технічне завдання до програмного модуля

Цілі і задачі програмного модуля: Програмний модуль є *Python*-модулем, що надає можливість надання рекомендацій пацієнтам. Програмний модуль має полегшувати створеня он-лайн експертної медичної системи.

Функціональні вимоги:

1. Валідація статі, віку, захворювань користувача, основних симптомів, додаткових симптомів, наведних питань, температури людини ти контакту з *COVID*–хворими;
2. Аналіз статі, віку, захворювань користувача, основних симптомів, додаткових симптомів, наведних питань, температури людини ти контакту з *COVID*-хворими;

2.3. Схеми алгоритму роботи функцій програмного модулю

Блок-схема – це схема системи, в якій основні частини або функції представлені блоками, з'єднаними лініями, що показують взаємозв'язки блоків.

Вони широко використовуються в інженерії в дизайні апаратних засобів, електронному дизайні, розробці програмного забезпечення та діаграмах технологічних процесів.

Блок-схеми зазвичай використовуються для вищого рівня, менш детальних описів, які призначені для уточнення загальних концепцій, не турбуючись про деталі реалізації.

2.3.1. Валідація

На блок-схемі валідації даних (рис. 2.1) зображено алгоритм, що проводить модуль для того аби надати користувачу можливість отримати рекомендації.

Аналізуючи технології, що надають розробникам інструменти валідації було вирішено створити власний інструмент валідації, який базується на заздалегіть визначеному списку даних.

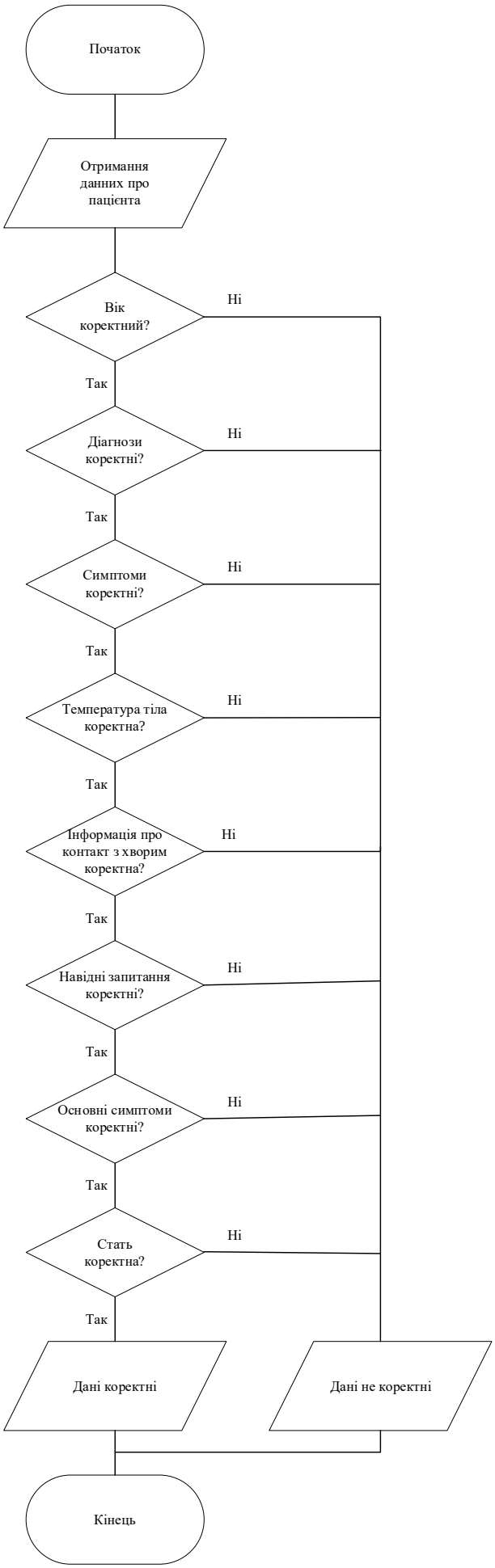

Рис. 2.1. Схема алгоритму валідації даних

2.3.2. Отримання рекомендації

Отримання рекомендації є основною функцією, що надає даний програмний модуль. Саме через це існує нагальна потреба максимально можливої правильної і точної її фізичної реалізації. Для цього перед її розробкою необхідно описати алгоритм, за яким ця функція буде діяти. Для цього було розроблено блок-схему отримання рекомандації (рис. 2.2).

Головною ціллю розробки цієї функції є надання користувачам найзручнішої та найпростішої можливості отримання рекомендації.

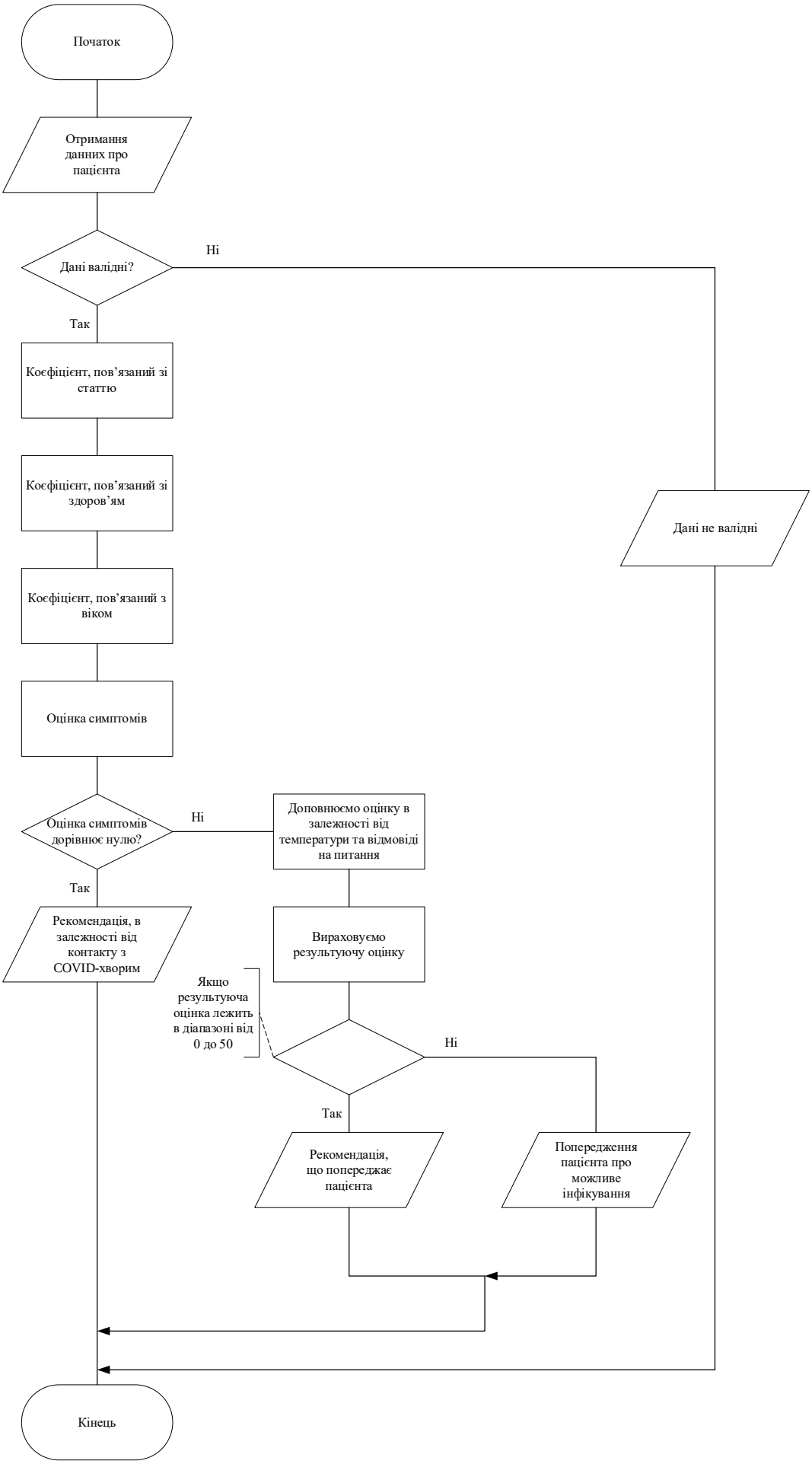

Рис. 2.2. Схема алгоритму отримання рекомендацій

2.3.3. Приклад використання за допомогою консольного додатку

Для наглядної демонстрації праці програмного модулю системи підтримки прийняття рішень в умовах пандемії був створений алгоритм консольного додатку (рис. 2.3).

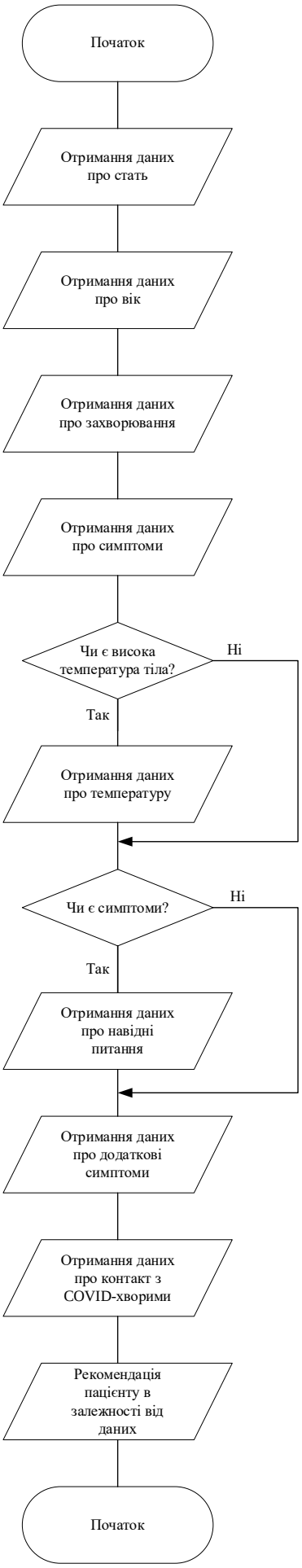

Рис. 2.3. Схема алгоритму консольного додатку

2.4. Висновки до розділу

Блок-схеми зазвичай використовуються для вищого рівня, менш детальних описів, які призначені для уточнення загальних концепцій, не турбуючись про деталі реалізації.

В розділі розглянуті та досліджені основні функції проєкту.

Для полегшення практичної розробки програмного модулю створені блок–схеми алгоритмів роботи основних функцій модулю:

– схема алгоритму валідації даних;

– схема алгоритму отримання рекомендації;

– схема алгоритму консольного додатку.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО МОДУЛЮ

3.1. Опис структури розроблювального програмного модулю

Для розробки програмний модулю системи підтримки прийняття рішень в умовах пандемії було застосовано наступні технології:

– інтегроване середовище розробки *PyCharm*;

– мова програмування *Python;*

– *Python*-бібліотека *decimal*;

– *Python*-бібліотека *pandas.*

Коди програмного модуля знаходяться у файлах:

– *recommendation.py*;

– *calculating.py*.

Коди валідації даних знаходятся у файлах:

– *validation.py*;

– *suggestive_questions_validation.py*;

– *main_symptoms_validation.py*;

– *gender_validation.py*;

– *contact_with_an_infected_in_last_14_days_validation.py*;

– *body_temperature_validation.py*;

– *age_validation.py*;

– *additional_symptoms_validation.py*;

– *additional_diseases_validation.py*.

Коди вираховування коєфіцієнтів знаходяться у файл:

– *calculate.py*.

Коди коєфіцієнтів знаходяться у файл:

– *data.py*.

Коди інтерфейсу знаходяться у файл:

– *interface.py*.

3.2. Вираховування коєфіцієнтів

Для коректного функціонування програмного модулю потрібно визначити статистично ризик захворювання *COVID*-19 в залежності від віку та статі. Для аналізу даних буде викоритсовуватися набір даних, у якому фіксували дані людей, що захворіли *COVID*-19. Набір даних взятий з відкритого джерела *https://www.kaggle.com/*. Назва набору данних: «*COVID19_open_line_list.csv*».

Для аналізу данних будемо використовувати бібліотеку pandas. Спочатку потрібно створити дві колекції, в яких будемо вести підрахунок хворих жіночої та чоловічої статей та вікового діапазону.

```
sex = {'male': 0, 'female': 0}
ages = {
    '0, 10': 0,
    '10, 20': 0,
    '20, 30': 0,
    '30, 40': 0,
    '40, 50': 0,
    '50, 60': 0,
    '60, 70': 0,
    '70, 120': 0
}
```

Далі зберігаємо доступ до даних за допомогою функції *pandas*. Для використаня бібліотеки pandas її потрібно імпортувати. Для зручності переіменуємо її в *pd*:

```
import pandas as pd
```

Далі підраховуємо кількість чоловіків та жінок в наборі даних. Після чого вираховуємо відсоток чоловіків та жінок від усіх суми статей.

```
for gender in df['sex'].dropna():
    if gender in sex.keys():
        sex[gender] += 1

all_count = sex['male'] + sex['female']
man = float(decimal.Decimal((sex['male'] * 100) / all_count))
woman = float(decimal.Decimal((sex['female'] * 100) / all_count))
```

Далі аналогічно підраховуємо кількість людей, що відносяться до діапазону віку та вираховуємо відсоток від загального.

```
for age in df['age'].dropna():
    if age.isdigit():
        if 0 <= int(age) < 10:
            ages['0, 10'] += 1
        elif 10 <= int(age) < 20:
            ages['10, 20'] += 1
        elif 20 <= int(age) < 30:
            ages['20, 30'] += 1
        elif 30 <= int(age) < 40:
            ages['30, 40'] += 1
        elif 40 <= int(age) < 50:
```

```
            ages['40, 50'] += 1
        elif 50 <= int(age) < 60:
            ages['50, 60'] += 1
        elif 60 <= int(age) < 70:
            ages['60, 70'] += 1
        elif 70 <= int(age) < 120:
            ages['70, 120'] += 1


    all_ages = 0

    for age in ages:
        all_ages += ages[age]

    # result
    proc_ages = {}

    for age in ages:
        proc_ages[age] = float(decimal.Decimal((ages[age] * 100) / all_ages))
```

3.3. Список можливих даних

Дані про можливі симптоми, хвороби, що упрощують інфікування вірусом та рекомендації були взяті зі звіту ВООЗ, що знаходятся за посиланням: *https://www.cdc.gov/coronavirus/2019–ncov/symptoms–testing/symptoms.html*. Навідні питання були створені на основі рекомендацій.

3.4. Валідація даних

Для коректної роботи програмного модуля відбувається перевірка коректності вхідних даних. Якщо дані про пацієнта є підмножиною можливих даних програмний модуль проведе аналіз та надасть рекомендації.

```
def validation(
    sex: str,
    age: int,
    additional_diseases: list,
    main_symptoms: list,
    additional_symptoms: list,
    suggestive_questions: list,
    body_temperature: str,
    contact_with_an_infected_in_last_14_days: str
):

    if age_validation(age) and \
        additional_diseases_validation(additional_diseases) and \
        additional_symptoms_validation(additional_symptoms) and \
        body_temperature_validation(body_temperature) and \

contact_with_an_infected_in_last_14_days_validation(contact_with_an_infected_in_l
ast_14_days) and \
        gender_validation(sex) and \
        main_symptoms_validation(main_symptoms) and \
        suggestive_questions_validation(suggestive_questions):
    return True
return False
```

3.5. Надання рекомендації

Спочатку відбувається перевірка на коректність вхідних даних.

```
if validation(
        sex,
        age,
        additional_diseases,
        main_symptoms,
        additional_symptoms,
        suggestive_questions,
        body_temperature,
        contact_with_an_infected_in_last_14_days
    )
```

Якщо дані валідні відбувається вирахування коєфіцієнтів статі та віку, підрахунку симптомів.

```
gender_rate = data.sex[sex]
    health_problem_rate = calc.health_rate(additional_diseases)
    age_rate = calc.age(age)
    symptom_score = calc.main_symptom(main_symptoms)
    symptom_score += calc.additional_symptom(additional_symptoms)
```

Якщо симптоми відсутні – алгоритм орієнтується на контакт з *COVID*-інфікованими.

```
    if symptom_score == 0:
        return
calc.contact_with_the_sick(contact_with_an_infected_in_last_14_days)
```

В інакшому випадку підраховуємо залишок даних та даємо рекомендацію в задежності від результату.

```
symptom_score += calc.high_fever(main_symptoms, body_temperature)

    symptom_score += calc.questions(suggestive_questions)

    result = calc.result(gender_rate, age_rate, health_problem_rate, symptom_score)

    if 0 < result <= 50:
        return data.result['some_problems']
    else:
        return data.result['big_problems']
```

3.6. Приклад використання

Для демонстрування використання програмного модулю був створений консольний додаток. Спочатку отримуємо стать користувача (рис. 3.1).

```
print("Вкажіть Вашу стать: ")
    print("1) Жінка")
    print("2) Чоловік")
    user_chose = int(input())
    if user_chose == 1:
        sex = 'woman'
    else:
        sex = 'man'
```

```
Вкажіть Вашу стать:
1) Жінка
2) Чоловік
2
```

Рис. 3.1. Отримання статі користувача

Далі отримуємо вік (рис. 3.2).

```
age = int(input("Введіть Ваш вік: "))
    while age >= 120 or age <= 0:
        age = int(input("Введіть Ваш вік: "))
```

```
Введіть Ваш вік: 20
```

Рис. 3.2. Отримання віку користувача

Далі отримуємо захворювання пацієнта (рис. 3.3).

```
print("Відзначте всі положення, які відносяться до Вас:")
    while True:
        diagnosis_id = int(input("1) В даний час є злоякісна пухлина\n"
                                 "2) Ослаблений імунітет\n"
                                 "3) Ожиріння\n"
                                 "4) Цукровий діабет\n"
                                 "5) Сердечно – судинні захворювання\n"
                                 "6) Є хронічне захворювання легенів\n"
                                 "7) Хронічне захворювання печінки в анамнезі\n"
                                 "8) Ниркова недостатність в анамнезі\n"
                                 "0) Далі\n"))

        if diagnosis_id == 0:
            break
```

```
diagnosis = list(data.additional_diseases.keys())[diagnosis_id – 1]
if diagnosis not in additional_diseases:
    additional_diseases.append(diagnosis)
else:
    print('Ви вже додали це захворювання')
```

```
Відзначте всі положення, які відносяться до Вас:
1) В даний час є злоякісна пухлина
2) Ослаблений імунітет
3) Ожиріння
4) Цукровий діабет
5) Сердечно - судинні захворювання
6) Є хронічне захворювання легенів
7) Хронічне захворювання печінки в анамнезі
8) Ниркова недостатність в анамнезі
0) Далі
```

Рис. 3.3. Отримання захворювань користувача

Отримуємо дані про симптоми (рис. 3.4).

```
print("Чи є у Вас якісь із наступних симптомів? Виберіть тільки симптоми, які з\'явилися зараз і не пов\'язані з "
      "наявними у Вас хронічним захворюванням.")
for key in data.main_symptoms.keys():
    print(key)
    user_chose = int(input('1) Так\n2) Ні\n'))
    if user_chose == 1:
        main_symptoms.append(key)
```

```
Чи є у Вас якісь із наступних симптомів?
Виберіть тільки симптоми, які з'явилися зараз і не пов'язані з наявними у Вас хронічним захворюванням.
Висока температура тіла
1) Так
2) Ні
2
Кашель
1) Так
2) Ні
2
Задишка
1) Так
2) Ні
2
```

Рис. 3.4. Отримання даних про симптоми користувача

Якщо одним з симптомів є висока температура – користувач вказує діпазон температури.

```
if 'Висока температура тіла' in main_symptoms:
    print('Яка у Вас температура тіла?')
    for index, temp in enumerate(data.body_temperature.keys()):
        print(f'{index + 1}) {temp}')
    user_chose = int(input())
    body_temperature = list(data.body_temperature.keys())[user_chose – 1]
```

Якщо симптоми у людини присутні – задаємо навідні питання.

```
if main_symptoms:
    for suggestive_question in data.suggestive_questions.keys():
        print(suggestive_question)
        user_chose = int(input('1) Так\n2) Ні\n'))
        if user_chose == 1:
            suggestive_questions.append(suggestive_question)
```

Отримуємо додаткові симптоми (рис. 3.5).

```
print("Чи є у Вас якісь із наступних симптомів?")
    while True:
        symptom_id = int(input("1) Патологічна стомлюваність\n"
                               "2) Біль у м\'язах\n"
```

```
                "3) Головний біль\n"
                "4) Діарея\n"
                "5) Нудота\n"
                "6) Біль в горлі\n"
                "7) Нежить\n"
                "8) Відсутність апетиту\n"
                "0) Далі\n"))

        if symptom_id == 0:
            break

        symptom = list(data.additional_symptoms.keys())[symptom_id – 1]
        if symptom not in additional_symptoms:
            additional_symptoms.append(symptom)
        else:
            print('Ви вже додали це захворювання')
```

```
Чи є у Вас якісь із наступних симптомів?
1) Патологічна стомлюваність
2) Біль у м'язах
3) Головний біль
4) Діарея
5) Нудота
6) Біль в горлі
7) Нежить
8) Відсутність апетиту
0) Далі
```

Рис. 3.5. Отримання даних про додаткові симптоми користувача

Отримуємо дані про контакт з *COVID*-інфокаманими людьми (рис. 3.6).

```
print('Контактували Ви в останні 14 днів з людиною з підозрою на інфекцію COVID-19?')
    for index, item in enumerate(data.contact_with_an_infected_in_last_14_days):
        print(f'{index + 1}) {item}')
    user_chose = int(input())
    contact_with_an_infected_in_last_14_days = data.contact_with_an_infected_in_last_14_days[user_chose – 1]
```

```
Контактували Ви в останні 14 днів з людиною з підозрою на інфекцію COVID-19?
1) Я спільно проживаю або доглядаю за особою, у якого підозрюється інфекція COVID-19
2) Я знаходився (лась) в загальному приміщенні (клас, робочий кабінет, спортивний зал) або перебував (лась) під час поїздки або перельоту в безпосередній близькості (1 м) від людини, у якого підозрюється COVID-19
3) У мене був контакт лицем до лиця тривалістю понад 15 хвилин з людиною, у якого підозрюється COVID-19
4) Інше
5) Нічого з перерахованого вище
```

Рис. 3.6. Отримання даних про контакт з *COVID*-інфокаманими людьми

Надаємо рекомендації користувачу (рис. 3.7).

```
print(calculating_a_recommendation(
    sex,
    age,
    additional_diseases,
    main_symptoms,
    additional_symptoms,
    suggestive_questions,
    body_temperature,
    contact_with_an_infected_in_last_14_days
))
```

```
Наявні у Вас симптоми не характерні для інфекції COVID-19. Дотримуйтесь звичайні заходи профілактики, для того щоб не заразитися.
```

Рис. 3.7. Надання рекомендацій користувачу

3.7. Висновки до розділу

В розділі проведено розробку та зроблено опис, щодо ключових компонентів функціоналу програмного модуля. Зокрема було розроблено такі фундаментальні елементи проєкту, як перевірка вхідних даних на коректність. Було проведено аналіз даних для отримання закономірності захворень в залежності від віку та статі. Розроблено метод аналізу вхідних даних та надання рекомендації. Для прикладу використання програмного модулю був створений консольний додаток.

Для повного уявлення про роботу програмного модулю був наведений список файлів, у яких описані усі необхідні функції програмного модулю.

ВИСНОВКИ

Метою виконання дипломного проєкту було створення програмного модулю системи підтримки прийняття рішень в умовах пандемії, що надає рекомендації в залежності від фізичного стану людини. Для створення програмного модулю були вирішені наступні задачі:

– досліджено поняття програмного модулю;

– проаналізовано ринок на наявність аналогічних ресурсів;

– проведено аналіз функціоналу існуючих аналогічних ресурсів;

– розглянуто та вивчено необхідні для створення модулю технології;

– розглянуто модуль *decimal*;

– розглянуто модуль *pandas*;

– створено програмний модуль для надання рекомендацій;

– створено консольний додаток для демонстрації роботи програмного модулю.

У дипломному проєкті було ознайомлено із роботою програмних модулів та створено власний зразок. Сьогодні існує безліч програмних модулів, що вирішуть різноманітні проблеми. Перелік проблем, що вирішують програмні модулі починаються з модифікації виводу інформації та закінчується модулями для створення штучного інтелекту.

В першому роздалі роботи було визначено що програмний модуль являє собою функціонально завершений фрагмент програми, оформлений у вигляді окремого файлу з сирцевим кодом або його іменованої частини, призначений для використання в інших програмах.

В другому розділі для створення програмного модулю було створено технічне завдання та реалізовані функції програмного модулю у вигляді блок–схем.

В третьому розділі роботи було створено безкоштовний, простий у використанні програмний модуль для отримання рекомендацій в залежності від стану людини.

Розроблений проєкт має наступні функції:

– валідація вхідних даних;

– аналіз даних для вираховування коєфіцієнтів;

– аналіз вхідних даних та надання рекомендацій;

– інтерфейс для використання програмного модулю.

Аналізуючи виконання проєкту, описаного у третьому розділі, можна зробити висновок, що програмний модуль можна використовувати в подальшому для створення відчизняної он–лайн експертної медичної системи.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проєкти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.
2. ДСТУ ГОСТ 3008–95 «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.»
3. Марк Лутц Изучаем *Python* – Київ: Диалектика, 2020. – 832 с.
4. *Decimal – Decimal fixed point and floating point arithmetic* [Електронний ресурс]. – 2021. – Режим доступу: *https://docs.python.org/3/library/decimal.html* (дата звернення 22.05.2021). – Назва з екрана
5. *pandas* documentation [Електронний ресурс]. – 2021. – Режим доступу: *https://pandas.pydata.org/docs/* (дата звернення 22.05.2021). – Назва з екрана.
6. Моя шпаргалка по *pandas* [Електронний ресурс]. – 2020. – Режим доступу: *https://habr.com/ru/company/ruvds/blog/494720/* (дата звернення 22.05.2021). – Назва з екрана.
7. РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ [Електронний ресурс]. – 2020. – Режим доступу: *http://bourabai.kz/alg/pro08.htm* (дата звернення 19.05.2021). – Назва з екрана.
8. Г.Майерс. Надежность программного обеспечения. – М.: Мир, 1980. – С. 127–154.
9. Э.Дейкстра. Заметки по структурному программированию / У.Дал, Э.Дейкстра, К.Хоор. Структурное программирование. – М.: Мир, 1975. – С. 24–97.
10. Н.Вирт. Систематическое программирование. – М.: Мир, 1977. – С. 94–164.
11. Разработка программного модуля. Структурное программирование [Електронний ресурс]. – 2018. – Режим доступу:

https://vuzlit.ru/1010733/razrabotka_programmnogo_modulya_strukturnoe_programmirovanie (дата звернення 19.05.2021). – Назва з екрана.

12. *Quick Documentation* [Електронний ресурс]. – 2021. – Режим доступу: *https://www.jetbrains.com/pycharm/guide/tips/quick–docs/* (дата звернення 19.05.2021). – Назва з екрана.
13. *PyCharm* — эффективная разработка на *Python* [Електронний ресурс]. – 2020. – Режим доступу: *https://chel-center.ru/python-yfc/2019/10/01/pycharm-mdash-instrument-effektivnoj-razrabotki-na-python/* (дата звернення 19.05.2021). – Назва з екрана.
14. Встроенные инструменты для разработчиков [Електронний ресурс]. – 2021. – Режим доступу: *jetbrains.com/ru-ru/pycharm/features/tools.html* (дата звернення 19.05.2021). – Назва з екрана.
15. Введение в *pandas* на русском языке [Електронний ресурс]. – 2018. – Режим доступу: *https://dfedorov.spb.ru/pandas/* (дата звернення 19.05.2021). – Назва з екрана.

ДОДАТОК А

Лістинг коду програмного модуля

Лістинг файлу *calculating.py*

```
import decimal
import data
def age(age_value: int):
    for age_range in data.ages:
        if int(age_range.split(', ')[0]) <= age_value < int(age_range.split(', ')[1]):
            return data.ages[age_range]


def health_rate(additional_diseases: list):
    health_problem_rate = 1
    for additional_diagnosis in additional_diseases:
        health_problem_rate +=
decimal.Decimal(data.additional_diseases[additional_diagnosis])
    return health_problem_rate


def main_symptom(main_symptoms: list):
    symptom_score = 0
    if main_symptoms:
        for symptom in main_symptoms:
            symptom_score += data.main_symptoms[symptom]
    return symptom_score


def additional_symptom(additional_symptoms: list):
    symptom_score = 0
    if additional_symptoms:
        for symptom in additional_symptoms:
            symptom_score += data.additional_symptoms[symptom]
    return symptom_score


def questions(suggestive_questions: list):
    symptom_score = 0
    if suggestive_questions:
        for suggestive_question in suggestive_questions:
            symptom_score += data.suggestive_questions[suggestive_question]
    return symptom_score
```

```
def result(gender_rate, age_rate, health_problem_rate, symptom_score):
    return float(decimal.Decimal(gender_rate) + \
                 decimal.Decimal(age_rate) + \
                 decim        al.Decimal(health_problem_rate)) * symptom_score


def high_fever(main_symptoms: list, body_temperature: str):
    symptom_score = 0
    if 'Висока температура тіла' in main_symptoms:
        symptom_score += data.body_temperature[body_temperature]
    return symptom_score


def contact_with_the_sick(contact_with_an_infected_in_last_14_days: str):
    if contact_with_an_infected_in_last_14_days == 'Нічого з перерахованого
вище':
        return data.result['okay']
    else:
        return data.result['have_contact_with_covid']
```

Лістинг файлу *recommendation.py*

```
from input_validation.validation import validation
import data
import calculating.calculating as calc


def calculating_a_recommendation(
        sex: str,
        age: int,
        additional_diseases: list,
        main_symptoms: list,
        additional_symptoms: list,
        suggestive_questions: list,
        body_temperature: str,
        contact_with_an_infected_in_last_14_days: str
):
    if validation(
            sex,
            age,
            additional_diseases,
            main_symptoms,
```

```
            additional_symptoms,
            suggestive_questions,
            body_temperature,
            contact_with_an_infected_in_last_14_days
        ):
            gender_rate = data.sex[sex]
            # коефіцієнт пов\'язаний з статтю
            health_problem_rate = calc.health_rate(additional_diseases)
            # коефіцієнт проблем зі здоров\'ям

            age_rate = calc.age(age)

            symptom_score = calc.main_symptom(main_symptoms)

            symptom_score += calc.additional_symptom(additional_symptoms)

            if symptom_score == 0:
                return
calc.contact_with_the_sick(contact_with_an_infected_in_last_14_days)
            else:
                symptom_score += calc.high_fever(main_symptoms, body_temperature)

                symptom_score += calc.questions(suggestive_questions)

                result = calc.result(gender_rate, age_rate, health_problem_rate,
symptom_score)

                if 0 < result <= 50:
                    return data.result['some_problems']
                else:
                    return data.result['big_problems']
        else:
            return 'Error validation'
```

Лістинг файлу *calculate.py*

```
import decimal

import pandas as pd


sex = {'male': 0, 'female': 0}
```

```
ages = {
    '0, 10': 0,
    '10, 20': 0,
    '20, 30': 0,
    '30, 40': 0,
    '40, 50': 0,
    '50, 60': 0,
    '60, 70': 0,
    '70, 120': 0
}
df = pd.read_csv("data/COVID19_open_line_list.csv", low_memory=False)

# sex
for gender in df['sex'].dropna():
    if gender in sex.keys():
        sex[gender] += 1

all_count = sex['male'] + sex['female']
man = float(decimal.Decimal((sex['male'] * 100) / all_count))
woman = float(decimal.Decimal((sex['female'] * 100) / all_count))

# ages

for age in df['age'].dropna():
    if age.isdigit():
        if 0 <= int(age) < 10:
            ages['0, 10'] += 1
        elif 10 <= int(age) < 20:
            ages['10, 20'] += 1
        elif 20 <= int(age) < 30:
            ages['20, 30'] += 1
        elif 30 <= int(age) < 40:
            ages['30, 40'] += 1
        elif 40 <= int(age) < 50:
            ages['40, 50'] += 1
        elif 50 <= int(age) < 60:
            ages['50, 60'] += 1
        elif 60 <= int(age) < 70:
            ages['60, 70'] += 1
        elif 70 <= int(age) < 120:
```

```
        ages['70, 120'] += 1


all_ages = 0

for age in ages:
    all_ages += ages[age]

# result
proc_ages = {}

for age in ages:
    proc_ages[age] = float(decimal.Decimal((ages[age] * 100) / all_ages))

print(man, woman)
print(proc_ages)
```

Лістинг файлу *additional_diseases_validation.py*

```
import data


def additional_diseases_validation(additional_diseases: list):
    """
    Перевірка корректності даних про захворювання пацієнта
    :param additional_diseases: list
    :return: boolean
    """
    if not additional_diseases:
        return True

    for additional_diagnosis in additional_diseases:
        if additional_diagnosis not in data.additional_diseases:
            return False

    return True
```

Лістинг файлу *additional_symptoms_validation.py*

```
import data


def additional_symptoms_validation(additional_symptoms: list):
    """
    Перевірка коректності даних про захворювання пацієнта
    :param additional_symptoms: list
    :return: boolean
    """
    if not additional_symptoms:
        return True

    for additional_symptom in additional_symptoms:
        if additional_symptom not in data.additional_symptoms:
            return False

    return True
```

Лістинг файлу *age_validation.py*

```
def age_validation(age: int):
    """
    Перевірка коректності даних про вік пацієнта
    :param age: int
    :return: boolean
    """
    if 0 <= age <= 120:
        return True
    return False
```

Лістинг файлу *body_temperature_validation.py*

```
import data


def body_temperature_validation(body_temperature: str):
    """
    Перевірка коректності даних про захворювання пацієнта
    :param body_temperature: str
    :return: boolean
    """
    if body_temperature == '' or body_temperature in data.body_temperature:
        return True
    return False
```

Лістинг файлу *contact_with_an_infected_in_last_14_days_validation.py*

```
import data


def
contact_with_an_infected_in_last_14_days_validation(contact_with_an_infected_in_l
ast_14_days: str):
    """
    Перевірка коректності даних про захворювання пацієнта
    :param contact_with_an_infected_in_last_14_days: str
    :return: boolean
    """
    if contact_with_an_infected_in_last_14_days == '' or \
            contact_with_an_infected_in_last_14_days in
data.contact_with_an_infected_in_last_14_days:
```

```
        return True
    return False
```

Лістинг файлу *gender_validation.py*

```
def gender_validation(sex: str):
    """
    Перевірка коректності даних про стать пацієнта
    :param sex: str
    :return: boolean
    """
    if sex == 'man' or sex == 'woman':
        return True
    return False
```

Лістинг файлу *main_symptoms_validation.py*

```
import data


def main_symptoms_validation(main_symptoms: list):
    """
    Перевірка коректності даних про захворювання пацієнта
    :param main_symptoms: list
    :return: boolean
    """
    if not main_symptoms:
        return True

    for main_symptom in main_symptoms:
        if main_symptom not in data.main_symptoms:
            return False

    return True
```

Лістинг файлу *suggestive_questions_validation.py*

```
import data


def suggestive_questions_validation(suggestive_questions: list):
    """
    Перевірка корректності даних про захворювання пацієнта
    :param suggestive_questions: list
    :return: boolean
    """
    if not suggestive_questions:
        return True

    for suggestive_question in suggestive_questions:
        if suggestive_question not in data.suggestive_questions:
            return False

    return True
```

Лістинг файлу *validation.py*

```
from input_validation.additional_diseases_validation import
additional_diseases_validation
from input_validation.additional_symptoms_validation import
additional_symptoms_validation
from input_validation.age_validation import age_validation
from input_validation.body_temperature_validation import
body_temperature_validation
from input_validation.contact_with_an_infected_in_last_14_days_validation
import \
    contact_with_an_infected_in_last_14_days_validation
from input_validation.gender_validation import gender_validation
from input_validation.main_symptoms_validation import
main_symptoms_validation
from input_validation.suggestive_questions_validation import
suggestive_questions_validation


def validation(
    sex: str,
    age: int,
    additional_diseases: list,
    main_symptoms: list,
```

```
        additional_symptoms: list,
        suggestive_questions: list,
        body_temperature: str,
        contact_with_an_infected_in_last_14_days: str
    ):
        """
        Валіданія усіх вхідних даних
        :param sex: str
        :param age: int
        :param additional_diseases: list
        :param main_symptoms: list
        :param additional_symptoms: list
        :param suggestive_questions: list
        :param body_temperature: str
        :param contact_with_an_infected_in_last_14_days: str
        :return: boolean
        """
        if age_validation(age) and \
            additional_diseases_validation(additional_diseases) and \
            additional_symptoms_validation(additional_symptoms) and \
            body_temperature_validation(body_temperature) and \

contact_with_an_infected_in_last_14_days_validation(contact_with_an_infected_in_last
_14_days) and \
            gender_validation(sex) and \
            main_symptoms_validation(main_symptoms) and \
            suggestive_questions_validation(suggestive_questions):
        return True
    return False
```

Лістинг файлу *data.py*

```
sex = {'man': .56, 'woman': .44}


ages = {
    '0, 10': .022,
    '10, 20': .028,
    '20, 30': .14,
    '30, 40': .22,
    '40, 50': .20,
    '50, 60': .17,
    '60, 70': .15,
    '70–120': .07
```

```
}

additional_diseases = {
    'В даний час є злоякісна пухлина': .125,
    'Ослаблений імунітет': .125,
    'Ожиріння': .125,
    'Цукровий діабет': .125,
    'Сердечно – судинні захворювання': .125,
    'Є хронічне захворювання легенів': .125,
    'Хронічне захворювання печінки в анамнезі': .125,
    'Ниркова недостатність в анамнезі': .125
}
# коєфіцієнти, що впливають на результат в залежності від додаткових захворювань пацієнта

main_symptoms = {
    'Висока температура тіла': 10,
    'Кашель': 10,
    'Задишка': 10
}
# умовні бали длярозрахунку результату

additional_symptoms = {
    'Патологічна стомлюваність': 5,
    'Біль у м\'язах': 5,
    'Головний біль': 5,
    'Діарея': 5,
    'Нудота': 5,
    'Біль в горлі': 5,
    'Нежить': 5,
    'Відсутність апетиту': 5
}
# умовні бали длярозрахунку результату

suggestive_questions = {
    'У Вас сильно прискорене дихання?': 20,
    'При виконанні повсякденних справ Ви відчуваєте задишку, запаморочення або сильну втому?': 20,
    'Ви не можете спокійно говорити повними реченнями через проблеми з диханням?': 20,
    'У Вас є відчуття сплутаності свідомості або дезорієнтації, наприклад, Ви не можете зрозуміти, '
    'де ви знаходитеся, або відчуваєте надмірну сонливість?': 20,
    'Ви недавно втратили здатність відчувати запах чи смак без жодної видимої причини для цього?': 20
}
```

```
body_temperature = {
    '37,5 – 38': 10,
    '38,1 – 40': 20,
    '>40': 30
}

contact_with_an_infected_in_last_14_days = [
    'Я спільно проживаю або доглядаю за особою, у якого підозрюється інфекція COVID–19',
    'Я знаходився (лась) в загальному приміщенні (клас, робочий кабінет, спортивний зал) або перебував (лась) під час\n'
    'поїздки або перельоту в безпосередній близькості (1 м) від людини, у якого підозрюється COVID–19',
    'У мене був контакт лицем до лиця тривалістю понад 15 хвилин з людиною, у якого підозрюється COVID–19',
    'Інше',
    'Нічого з перерахованого вище'
]

result = {
    'okay': 'Наявні у Вас симптоми не характерні для інфекції COVID–19. Дотримуйтесь звичайні заходи профілактики, '
            'для того щоб не заразитися.',
    'big_problems': 'Наявні у Вас симптоми дуже серйозні. Можливо, Ви інфіковані COVID–19.',
    'have_contact_with_covid': 'Наявні у Вас симптоми не характерні для інфекції '
                               'COVID–19. Однак відповідно до рекомендацій ВООЗ і Центрів по контролю і профілактиці\n'
                               'захворювань США, '
                               'настійно рекомендується протягом наступних 14 днів ізолювати себе від оточуючих і\n'
                               'стежити за можливою появою '
                               'симптомів COVID–19.',
    'some_problems': 'Відповідно до рекомендацій Всесвітньої організації охорони здоров\'я, наявні у Вас симптоми не\n'
                     'підтверджують наявність у Вас інфекції COVID–19. Все ж буде краще, якщо Ви залишитеся вдома. Це\n'
                     'буде корисніше для Вашої '
                     'безпеки і безпеки інших людей. Якщо Ви стурбовані вагою наявних у Вас симптомів, зверніться до '
                     'лікаря. '

}
```

Лістинг файлу *interface.py*

```
import data
from calculating.recommendation import calculating_a_recommendation


def interface():
    sex = str
    age = int
    additional_diseases = []
    main_symptoms = []
    additional_symptoms = []
    suggestive_questions = []
    body_temperature = ''
    contact_with_an_infected_in_last_14_days = ''

    print("Вкажіть Вашу стать: ")
    print("1) Жінка")
    print("2) Чоловік")
    user_chose = int(input())
    if user_chose == 1:
        sex = 'woman'
    else:
        sex = 'man'

    age = int(input("Введіть Ваш вік: "))
    while age >= 120 or age <= 0:
        age = int(input("Введіть Ваш вік: "))

    print("Відзначте всі положення, які відносяться до Вас:")
    while True:
        diagnosis_id = int(input("1) В даний час є злоякісна пухлина\n"
                                 "2) Ослаблений імунітет\n"
                                 "3) Ожиріння\n"
                                 "4) Цукровий діабет\n"
                                 "5) Сердечно – судинні захворювання\n"
                                 "6) Є хронічне захворювання легенів\n"
                                 "7) Хронічне захворювання печінки в анамнезі\n"
                                 "8) Ниркова недостатність в анамнезі\n"
                                 "0) Далі\n"))

        if diagnosis_id == 0:
            break

        diagnosis = list(data.additional_diseases.keys())[diagnosis_id – 1]
        if diagnosis not in additional_diseases:
```

```
            additional_diseases.append(diagnosis)
        else:
            print('Ви вже додали це захворювання')

    print("Чи є у Вас якісь із наступних симптомів? Виберіть тільки симптоми, які з\'явилися зараз і не пов\'язані з "
          "наявними у Вас хронічним захворюванням.")
    for key in data.main_symptoms.keys():
        print(key)
        user_chose = int(input('1) Так\n2) Ні\n'))
        if user_chose == 1:
            main_symptoms.append(key)

    if 'Висока температура тіла' in main_symptoms:
        print('Яка у Вас температура тіла?')
        for index, temp in enumerate(data.body_temperature.keys()):
            print(f'{index + 1}) {temp}')
        user_chose = int(input())
        body_temperature = list(data.body_temperature.keys())[user_chose – 1]

    if main_symptoms:
        for suggestive_question in data.suggestive_questions.keys():
            print(suggestive_question)
            user_chose = int(input('1) Так\n2) Ні\n'))
            if user_chose == 1:
                suggestive_questions.append(suggestive_question)

    print("Чи є у Вас якісь із наступних симптомів?")
    while True:
        symptom_id = int(input("1) Патологічна стомлюваність\n"
                               "2) Біль у м\'язах\n"
                               "3) Головний біль\n"
                               "4) Діарея\n"
                               "5) Нудота\n"
                               "6) Біль в горлі\n"
                               "7) Нежить\n"
                               "8) Відсутність апетиту\n"
                               "0) Далі\n"))

        if symptom_id == 0:
            break

        symptom = list(data.additional_symptoms.keys())[symptom_id – 1]
        if symptom not in additional_symptoms:
            additional_symptoms.append(symptom)
        else:
```

```
            print('Ви вже додали це захворювання')

        print('Контактували Ви в останні 14 днів з людиною з підозрою на
інфекцію COVID–19?')
        for index, item in enumerate(data.contact_with_an_infected_in_last_14_days):
            print(f'{index + 1}) {item}')
        user_chose = int(input())
        contact_with_an_infected_in_last_14_days =
data.contact_with_an_infected_in_last_14_days[user_chose – 1]

        print(calculating_a_recommendation(
            sex,
            age,
            additional_diseases,
            main_symptoms,
            additional_symptoms,
            suggestive_questions,
            body_temperature,
            contact_with_an_infected_in_last_14_days
        ))
```

Лістинг файлу *main.py*

```
from interface import interface

if __name__ == '__main__':
    interface()
```