НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Програмне забезпечення для моніторингу та аналізу психічного стану пацієнта»

Виконав:
студент IV курсу, групи КП-62
Пономарчук Дмитро Юрійович _____

Керівник:
Старший викладач кафедри ПЗКС, к.ф.-м.н.,
Гречко Анастасія Валеріївна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
доц. каф. СПСКС, доц., к.т.н., доцент,
Боярінова Юлія Євгенівна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ Іван ДИЧКА

«___»_____2019 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Пономарчуку Дмитру Юрійовичу

1. Тема проєкту «Програмне забезпечення для моніторингу та аналізу психічного стану пацієнта», керівник проекту Гречко Анастасія Валеріївна, ст. викл., затверджені наказом по університету від «25» травня 2020 р. №1181-с

2. Термін подання студентом проєкту «11» червня 2020 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - огляд наявних програмних рішень;
   - огляд математичних методів психологічного аналізу;
   - опис реалізації;
   - аналіз реалізації програмного застосунку.

5. Перелік обов'язкового графічного матеріалу:
   - UML діаграма прецендентів (креслення);
   - структура бази даних. ER діаграма (креслення);
   - робота service-workers (плакат);
   - робота модулів cypress.io (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	14.11.2019	
2.	Розроблення та узгодження технічного завдання	28.11.2019	
3.	Розроблення структури web-ресурсу	15.12.2019	
4.	Підготовка матеріалів першого розділу дипломного проєкту	30.12.2019	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2020	
6.	Підготовка матеріалів другого розділу дипломного проєкту	20.02.2020	
7.	Програмна реалізація web-ресурсу	10.03.2020	
8.	Тестування web-ресурсу	17.03.2020	
9.	Підготовка матеріалів третього розділу дипломного проєкту	30.03.2020	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	11.04.2020	
11.	Підготовка графічної частини дипломного проєкту	21.04.2020	
12.	Оформлення документації дипломного проєкту	26.05.2020	

Студент Дмитро ПОНОМАРЧУК

Керівник проєкту Анастасія ГРЕЧКО

АНОТАЦІЯ

Даний дипломний проєкт присвячений розробленню web-сервісу для моніторингу та аналізу психічного стану пацієнта.

У роботі виконано порівняльний аналіз існуючих рішень для моніторингу та аналізу психічного стану пацієнта, проаналізовано методи для моніторингу та аналізу психічного стану пацієнта, обґрунтовано вибір технологій та допоміжних бібліотек серверної та клієнтської частин для реалізації даного web-сервісу. Розроблений web-сервіс надає терапевтам можливість здійснювати збір та аналіз інформації щодо конкретного пацієнта для подальшого аналізу та моніторингу його психічного стану. Збір інформації здійснюється за допомогою автоматизованих інструментів. Процес аналізу інформації здійснюється автоматично, відносно попередньо встановлених правил аналізу. Результатом аналізу є рекомендаційні настанови, а також розвернута характеристика відносно психічного стану пацієнта.

В даному проєкті розроблено та досліджено: архітектуру серверної та клієнтської частини web-сервісу, алгоритм автоматичного та ручного збору інформації, алгоритм обробки та підбору правил аналізу інформації, алгоритм аналізу інформації, а також графічні елементи та дизайн web-сторінок.

ABSTRACT

This diploma project is dedicated to the development of web-service for monitoring and analysis of the patient's mental state.

During the writing of the diploma project was carried out a comparative analysis of the existing solutions for monitoring and analysis of the patient's mental state, analyzed the methods used for monitoring and analysis of the patient's mental state, justified the choice of technologies and supporting libraries of server and client parts for this web-service. Developed web-service that therapists the opportunity to collect and analyze information about a particular patient for further analysis and monitoring of his mental state. Information is collected using automated tools. The process of gathering information can be fulfilled by automated instruments. The process of analyzing the information is carried out automatically, with respect to predefined rules. The result of an analysis are recommended steps, as well as detailed description of the mental state of the patient.

In this diss was developed: the architecture of server and client side of web-service, the algorithm of automatic and manual information gathering, the algorithm of selection and processing rules of gathering information, information analysis algorithm, graphic elements and design of web-pages.

ДП.045440-01-90 Програмне забезпечення для моніторингу та аналізу психічного стану пацієнта. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Програмне забезпечення	5	
	для моніторингу та		
	аналізу психічного стану		
	пацієнта. Технічне		
	завдання		
ДП.045440-03-81	Програмне забезпечення	5	
	для моніторингу та		
	аналізу психічного стану		
	пацієнта. Пояснювальна		
	записка		
ДП.045440-04-51	Програмне забезпечення	4	
	для моніторингу та		
	аналізу психічного стану		
	пацієнта. Програма та		
	методика тестування		
ДП.045440-05-34	Програмне забезпечення	8	
	для моніторингу та		
	аналізу психічного стану		
	пацієнта. Керівництво		
	користувача		
ДП.045440-06-99	Програмне забезпечення	1	
	для моніторингу та		
	аналізу психічного стану		
	пацієнта. UML діаграма		
	прецедентів		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-07-99	Програмне забезпечення	1	
	для моніторингу та		
	аналізу психічного стану		
	пацієнта. Структура		
	бази даних. ER діаграма		
ДП.045440-08-98	Програмне забезпечення	1	
	для моніторингу та		
	аналізу психічного стану		
	пацієнта. Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___»_____2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ТА АНАЛІЗУ ПСИХІЧНОГО СТАНУ ПАЦІЄНТА

Технічне завдання

ДП.045440-02-91

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Анастасія ГРЕЧКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитро ПОНОМАРЧУК

2019

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: програмне забезпечення для моніторингу та аналізу психічного стану пацієнта.

Галузь застосування: психологія.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості допоміжного інструменту для генерування опитувальників та аналізу психічного стану пацієнта психологами з метою зменшити ризики втрати часу та точності оцінювання.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмна система повинна забезпечувати такі основні функції:

1. Система повинна надавати можливість проглядати, створювати та редагувати опитувальники для пацієнтів.
2. Система надаватиме можливість для пацієнтів проходити та проглядати опитувальники.

3. Застосунок повинен мати функціонал для генерування висновку та змогу перегляду згенерованого висновку.
4. Застосунок матиме функціонал для моніторингу динаміки стану пацієнта та відображення цієї динаміки на стороні клієнта.
5. Наявність функціоналу для генерування висновку та моніторингу динаміки стану та інтерфейсу для показу висновків.
6. Змога порівнювати висновки, результати опитувальників з іншими пацієнтами або з тим же.

Розроблення серверної частини виконати на мові програмування, використовуючи веб-фреймворк. Розроблення клієнтської частини виконати, використовуючи бібліотеки для забезпечення інтерактивності та динамічності веб-сторінки.

Додаткові вимоги:

1. Застосунок повинен підтримуватися у всіх сучасних браузерах.
2. Застосунок надаватиме можливість реєструватися користувачам, як терапевт та пацієнт.
3. Застосунок надаватиме можливість реєструвати інших користувачів, користувачу з правами адміністратора.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1. Пояснювальна записка.
2. Програма та методика тестування.
3. Керівництво користувача.
4. Креслення.

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою проєкту 16.11.2019
Розроблення та узгодження технічного завдання..... 09.12.2019
Підготовка матеріалів першого розділу дипломного проєкту 30.12.2019
Розроблення алгоритмів для аналізу та моніторингу 16.01.2020
Підготовка матеріалів другого розділу дипломного проєкту..... 10.02.2020
Програмна реалізація та тестування програмної системи 20.02.2020
Підготовка третього розділу дипломного проєкту..... 10.03.2020
Підготовка четвертого розділу дипломного проєкту 11.04.2020
Підготовка графічної частини дипломного проєкту 19.05.2020
Оформлення документації дипломного проєкту 26.05.2020

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ І.А. Дичка
«___» _____ 2020 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ТА АНАЛІЗУ ПСИХІЧНОГО СТАНУ ПАЦІЄНТА

Пояснювальна записка

ДП.045440-03-81

«ПОГОДЖЕНО»

Керівник проєкту:
_____ Анастасія ГРЕЧКО

Нормоконтроль:
_____ Микола ОНАЙ

Виконавець:
_____ Дмитро ПОНОМАРЧУК

2020

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення;

БД – база даних;

Веббраузер – прикладне програмне забезпечення для перегляду сторінок, змісту веб-документів, комп'ютерних файлів і їх каталогів;

Python – інтерпретована об’єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією;

Парадигма програмування – система правил для визначення стилю написання програм;

JavaScript (JS) – динамічна, об'єктно-орієнтована прототипна мова програмування;

ECMAScript – це скриптова мова, яка створена для стандартизації мови програмування JavaScript;

Java — об'єктно-орієнтована мова програмування;

Фронтенд – frontend – клієнтська частина – клієнтська сторона користувацького інтерфейсу до програмно-апаратної частини сервісу;

Бекенд – backend – серверна частина – невидима частина сайту, яка відповідає за роботу сайту;

HyperText Markup Language (HTML) – це один з основних будівельних блоків будь-якого сайту, диктує організацію і контент сайту;

Cascading Style Sheets (*CSS)* – це один з основних будівельних блоків будь-якого сайту, містить код для кожного графічного елемента – від фонів до шрифтів – який становить зовнішній вигляд веб-сайту;

Скрипт – програмні інструкції виконання;

Фреймворк – framework – програмне забезпечення, яке полегшує процес розроблення на об’єднання різних модулів програмного проєкту;

Психологічний тест – тест-опитувальник – опитувальник – стандартизована методика, спрямована на вимір індивідуальних

властивостей і якостей респондента (психофізіологічних і особистісних характеристик, здібностей, знань і навичок, станів);

Progressive web app (PWA) – технологія в web-розробці, яка візуально і функціонально трансформує сайт в додаток (мобільний додаток в браузері);

Single page application (SPA) – це веб-додаток або веб-сайт, який використовує єдиний HTML-документ як оболонку для всіх веб-сторінок і організує взаємодію з користувачем через динамічно підкачуємі HTML, CSS, JavaScript, зазвичай за допомогою AJAX;

Asynchronous Javascript and XML (AJAX) – це сукупність методів веб-розробки, що використовують багато веб-технологій на стороні клієнта для створення асинхронних веб-додатків;

Personal Home Page Tools (PHP) – популярна скриптована мова загального призначення, яка особливо підходить для веб-розробки.

ВСТУП

Психологія дуже давня наука, яка слідувала за людством майже все його життя. З розвитком психології з’являлося багато відгалужень психології, видів та способів досліджень людини, але майже всі процеси в цій науці залишилися не автоматизовані. Одним з найпопулярніших методів дослідити людську душу – провести опитування. Опитування дуже часто націлені не на конкретну людину та проводяться в групі опитуваних людей.

Психологічне опитування відноситься до розділу психодіагностики і займається вивченням психологічних якостей і властивостей особистості через застосування психологічних опитувань. Цей метод часто застосовується в консультуванні, психотерапії, а також роботодавцями при прийомі на роботу. Психологічні опитування потрібні, коли потрібно дізнатися про особистості людини більш детально, чого не можна зробити за допомогою бесіди або опитування.

Ці опитування проводяться в основному в ручному режимі людиною, але це займає багато часу, та не завжди є точним через людські помилки. Але автоматизоване психологічне опитування вирішує цю проблему та має великий потенціал для полегшення роботи психолога, психіатра або, справді, кожного, хто використовує психологічні опитування.

1. ОГЛЯД НАЯВНИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Огляд проблеми, яка вирішується ПЗ

Методи дослідження в психології – це ті прийоми та засоби, за допомогою яких психологи отримують достовірні відомості, що використовуються для побудови наукових теорій і вироблення практичних рекомендацій. У даній роботі автоматизуємо метод опитування, тому що це один з найпопулярніших методів дослідження у цій сфері діяльності, до того ж він єдиний метод, який можна автоматизувати [31, 1, 2]. Серед інших методів опитування виділяється такими перевагами:

Валідність – відповідність отриманих з опитувальника даних тієї характеристики, для якої опитування проводиться;

Надійність – відповідність отриманих результатів при повторному опитуванні;

Достовірність – властивість опитувальника давати справжні результати, навіть при навмисних або ненавмисних спробах їх спотворення випробовуваними;

Репрезентативність – відповідність нормам.

Порядок проведення наукового дослідження в галузі психології на сьогодні:

- відповідно до наукової проблеми та теми дослідження визначають його об’єкт і предмет, мету, гіпотезу та завдання;
- вибір наукових методів дослідження, які доповнюють один одного (наприклад, спостереження та експеримент);
- визначення умов дослідження (лабораторний або природний експеримент);
- розроблення плану експериментального дослідження.
- вибір методів оброблення емпіричних даних (кількісний, якісний аналіз);
- інтерпретація зібраного експериментального матеріалу;

- формулювання висновків і визначення сфери їх застосування.

Що заважає об’єктивно інтерпретувати одержані результати дослідження терапевту?

По-перше, певна упередженість дослідника, схильність бачити те, що він бажає бачити. Через це він інколи не помічає ті факти, які не збігаються з його очікуваннями (гіпотезою дослідження).

По-друге, інколи дослідникові бракує спостережливості, або ж він сприймає психологічний факт дуже вузько чи надто широко.

Така перевірка займає багато часу у психологів або психотерапевтів та може призвести до помилок в перевірці або у висновку через людський фактор. Тому головною проблемою наразі є відсутність автоматизованої системи для проведення опитувань, перевірки результатів та генерування висновку. Випливаючи з цього, мета даної роботи – створення програмного застосунку для автоматизованого моніторингу та аналізу психічного стану пацієнта, яка буде розв'язувати наступні проблеми:

- автоматизоване збирання і реєстрування у режимі реального часу одержаних результатів дослідження;
- зберігання зібраних даних та інформації;
- аналіз та моніторинг зібраних даних;
- генерування висновків на основі зібраної інформації.

1.2. Аналіз наявних систем-аналогів

Повних аналогів розроблюваної в даній роботі системи на ринку не представлено. Більшість знайдених застосувань або належать певним приватних компаніям або університетам, або це загальні застосунки для створювання опитувальників, тобто це не професійний інструменти направлені на те, щоб можна було слідкувати за статусом опитуваних. Проаналізуємо особливості деяких з цих систем.

1.2.1. *Questbase*

Веб-платформа, яка надає все необхідне для створення та управління вашими опитувальниками, опитуваннями та іспитами, як онлайн, так і друкованими варіантами. Розроблений як інструмент для навчання та дослідження, QuestBase також може використовуватися для вибіркових або психологічних опитувальників, опитування задоволеності та думки, дослідження ринку та для збирання відгуків клієнтів.

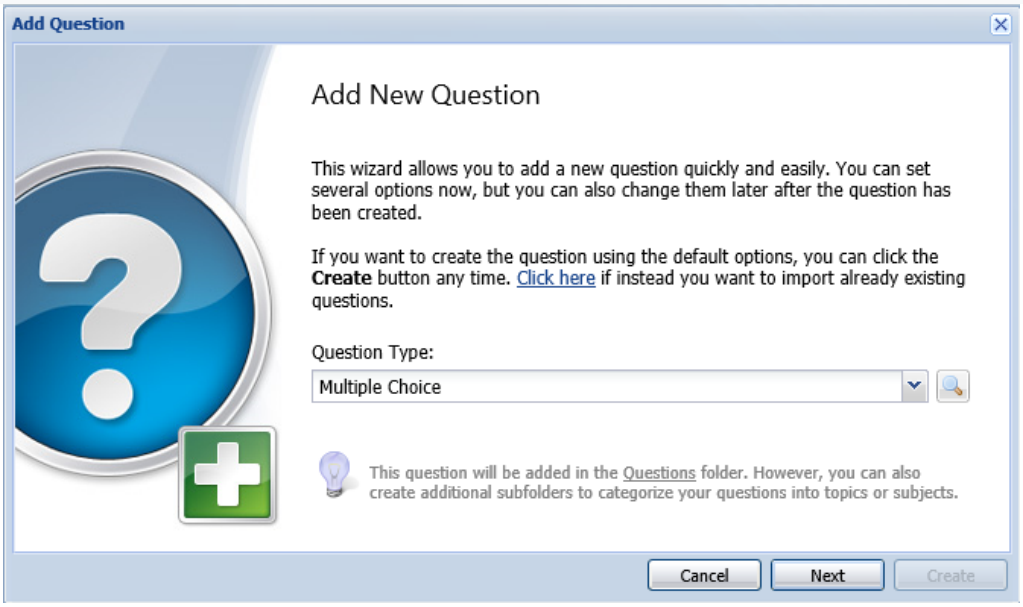


Рис. 1.1. Ілюстрація створення нового опитувальника, перший крок

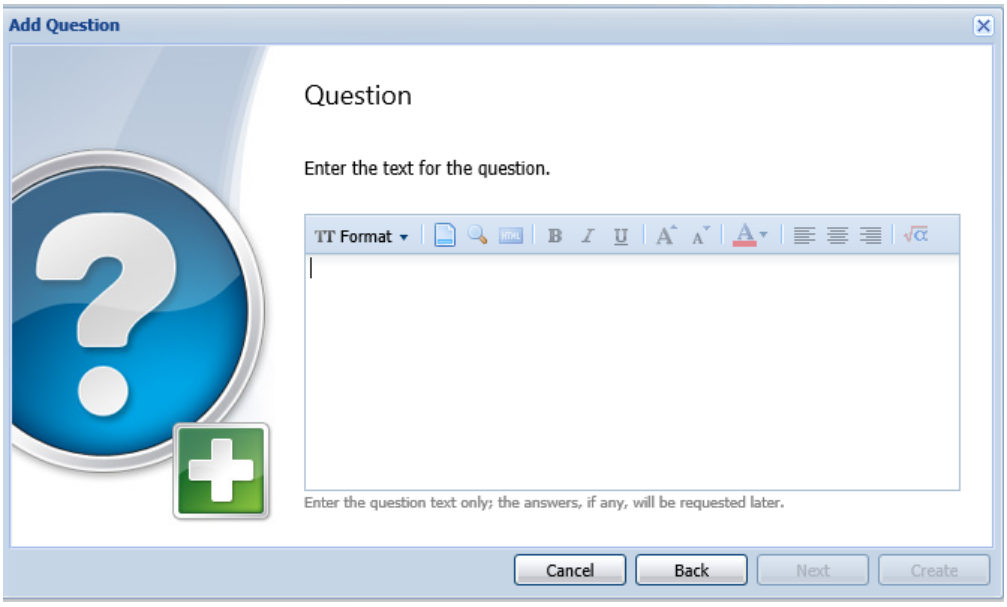


Рис. 1.2. Ілюстрація створення нового опитувальника, другий крок

Дане рішення добре підходить для створення простеньких опитувальників після проходження яких ми зможемо продивитися всі відповіді, які були зроблені користувачами. У застосунка не дуже достатньо зрозумілий інтерфейс для користувачів та мізерний функціонал (наведено в рис. 1.1 та рис. 1.2 на прикладі створення опитувальника) який не дозволяє в автоматизованому режимі зробити висновки на основі результатів опитувальників та порівняти відповіді інших користувачів.

Проаналізувавши дане програмне забезпечення ми можемо виділити наступне: переваги системи це підтримка тестів та опитувальників для різних напрямків, використання декілька типів питань. При цьому головна проблема залишається – не має функціоналу для автоматизації всього процесу та не конкретна націленість на психологічні напрямки, також деякі мінорні мінуси по типу не зрозуміло інтрефейсу.

1.2.2. *Psychology Today*

Веб-застосунок, де зібрали групи відомих психологів, науковців, психіатрів та письменників, щоб донести свої думки та ідеї. Присутнє проходження заздалегідь підготовлених психологічних опитувань (рис. 1.3) та генерування висновку на основі пройденого раніше опитувальника. Також є можливість знайти собі спеціаліста в психологічному напрямку для подальшого лікування.

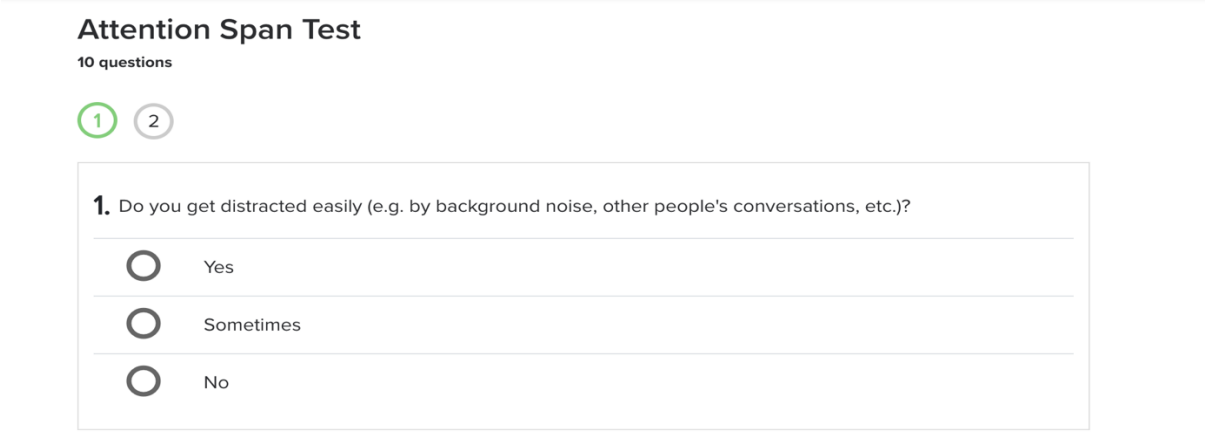


Рис. 1.3. Ілюстрація проходження опитувальника, та одне із питань

Це рішення уже більш професійне в плані психології та психіатрії, тут присутні онлайн консультації у відомих психологів та науковців, також в наявності опитувальники, які націлені на психоаналіз людини та генерування висновку на результаті цього, також більш зручний та зрозумілий інтерфейс.

По-перше, один із головних мінусів застосунку, те що опитувальники не можна створювати та редагувати для звичайних користувачів, всі створенні опитувальники заздалегідь підготовленні адміністраторами. По-друге, варіативність питань мала, маємо тільки один вид питань. Також ці опитування анонімні, тобто ви не зможете побачити хто відповів на ці питання, не зможете прослідкувати динаміку змін стану пацієнта. Якщо підсумувати все сказане вище цей застосунок не відповідає деяким вимогам: не маємо прив'язки пацієнт – терапевт, недостатній функціонал для проведення аналізу пацієнтів.

1.2.3. *Flexi Quiz*

FlexiQuiz надає ряд особливостей чіткого та простого дизайну, щоб зробити застосунок легким у використанні. Редагування опитувальника під власну фірму, школу чи університети, що підлаштовуються під кольори та зображення вашого бренду. Вікторина також може бути написана декількома мовами, щоб відповідати вашим вимогам. Існує декілька способів налаштування опитувальника від задання кількості запитань (рис. 1.4) на сторінці аж до встановлення обмеження часу на проходження опитувальнику. Використовуючи шаблони FlexiQuiz для створення власних привітальних сторінок, подяк та запитань із відгуками. Присутні прості способи публікації та розповсюдження вікторин. Економія часу через автоматизовану оцінку опитувальника. Ви у змозі вибирати правильні відповіді, кількість балів за кожне питання та загальну швидкість проходження. Існують також деякі більш вдосконалені функції, включаючи негативне оцінювання та ручне позначення питань для написання вільного

тексту після подання вікторини. Також існують три види користувачів респондент, адміністратор та тренер. Респонденти – ви можете створити приватний обліковий запис для учасників та для перегляду будь-яких опитувальників, які були призначені їм. Адміністратори – ви можете створити обліковий запис для інших людей у вашій команді. Вони матимуть доступ до всієї тієї самої інформації, що і тренера та респонденти та деякі додаткові функції. Вони також можуть створювати та ділитися власними опитувальниками. Тренери – матимуть доступ до обраних тестів та опитувальників, які ви призначили їм. Вони також зможуть створити власні опитувальники.

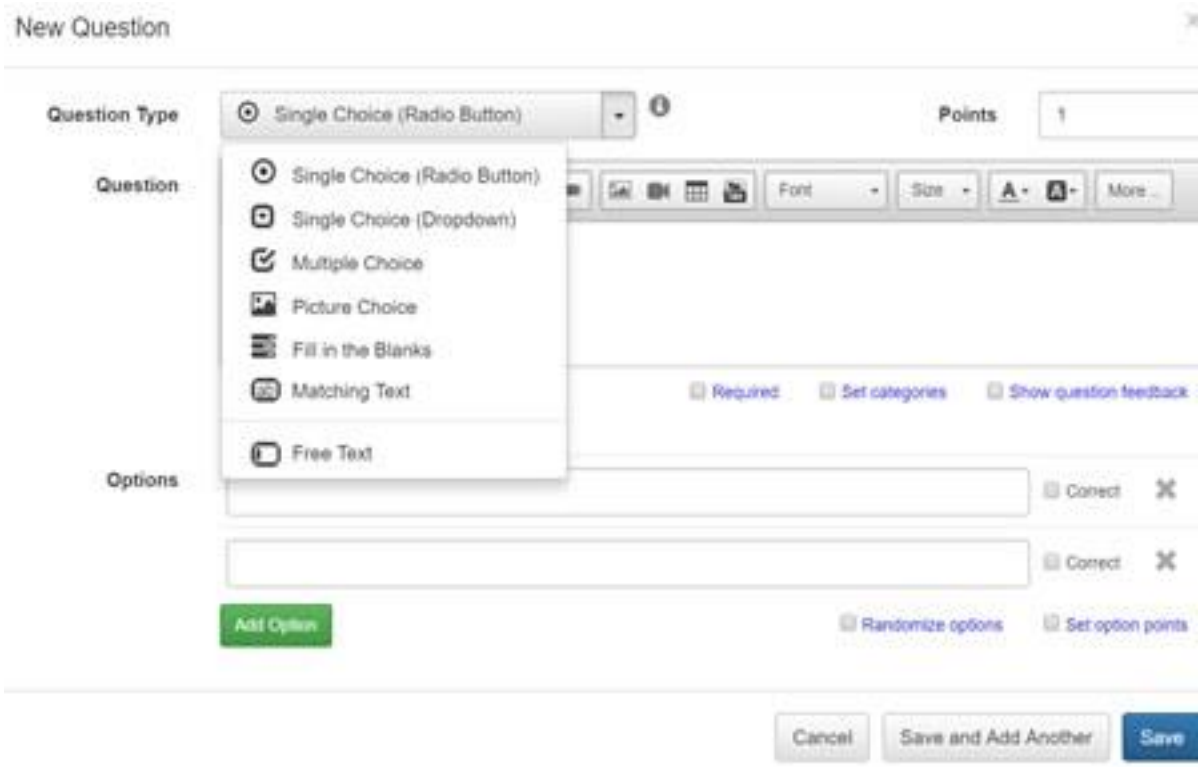


Рис. 1.4. Ілюстрація створення питання

З точки зору вимог для аналізу програмного забезпечення, данасистема має велику кількість переваг порівнюючи з вище згаданими варіантами, тут присутнє майже весь функціонал який потрібний для вимог. Присутні три ролі для користувачів, присутній функціонал для створення та редагування опитувальників. Але навіть враховуючи всі плюси цього

застосунку він не пристосований для професійного психологічного консультування, моніторингу та аналізу психічного стану, тому що тут націленість йде на навчання. Тут до кожного питання йде прив'язка оцінки і в кінці ми отримуємо результат з оцінкою, без аналізування даних відповідей.

1.2.4. *Typeform*

Один із найпопулярніших програмних застосунків для конструювання тестів, опитувальників. Вікторина дарує людям веселе та інтерактивне враження. Для вчителів – це чудовий інструмент навчання для студентів, який може покращити досвід роботи в класі. Зробити вікторину з Typeform легко, використовуючи функції Essentials (і вище), також спеціальні екрани подяки, калькулятор відповідей та багато іншого.

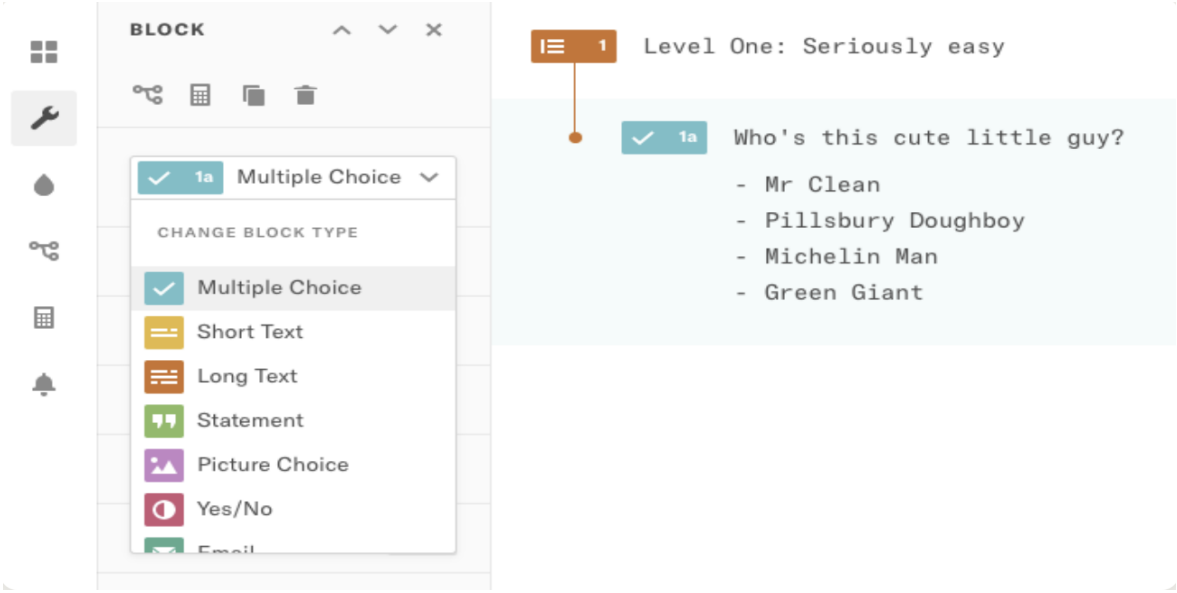


Рис. 1.5. Ілюстрація створення питання

Typeform доволі зручний конструктор для опитувальників з великим функціоналом для створення простих опитувальників (рис. 1.5) без аналізу. Застосунок майже не задовольняє наші вимоги через не зручну калькуляцію результатів, відсутність моніторингу та генерування висновків.

1.3. Аналіз вимог до розроблюваного ПЗ

Опитавши цільову аудиторію та проаналізувавши наявні системи-аналоги було сформовано вимоги до програмного забезпечення, яким має відповідати дане програмне рішення моніторингу та аналізу психічного стану. В системі буде два типи користувачів: лікарі та пацієнти. Лікарі в змозі створювати нові опитувальники на основі результатів яких потім буде згенерований висновок. Після генерування висновків та проходженням декількох опитувальників зі сторони пацієнта, система буде генерувати динаміку висновків та стану пацієнта. Лікарю також потрібно надати змогу порівнювати результати одного пацієнта або декількох. Також програмне рішення має бути реалізоване достатньо зрозуміло для всіх користувачів.

Таким чином сформуємо вимоги:

- застосунок повинен підтримуватися у всіх сучасних браузерах;
- застосунок надаватиме можливість реєструватися користувачам, як терапевт та пацієнт;
- застосунок надаватиме можливість реєструвати інших користувачів, користувачу з правами адміністратора;
- система повинна надавати можливість проглядати, створювати та редагувати опитувальники для пацієнтів;
- система надаватиме можливість для пацієнтів проходити та проглядати опитувальники;
- застосунок повинен мати функціонал для генерування висновку та змогу перегляду згенерованого висновку;
- застосунок матиме функціонал для моніторингу динаміки стану пацієнта та відображення цієї динаміки на стороні клієнта;
- наявність функціоналу для генерування висновку та моніторингу динаміки стану та інтерфейсу для показу висновків;
- змога порівнювати висновки, результати опитувальників з іншими пацієнтами або з тим же.

1.4. Постановка завдання

Проаналізувавши рішення від конкурентів, можна стверджувати, що їх програмне забезпечення не задовольняє усіх вимог. А ті програмні рішення, які реалізовані, націлені на звичайні опитування без автоматизації процесу генерування висновків або з частковою автоматизацією та мають деякі недоліки в генеруванні висновків та моніторингом за висновками або результатами, якщо це відноситься до більш професійного застосунку. При цьому, кожне з наявних рішень добре реалізує деяку конкретну частину функціонала.

В результаті аналізу наявних аналогів, їх особливостей, переваг та недоліків було сформульовано список бажаних функціональних особливостей розроблюваної системи:

- система надаватиме змогу реєстрації користувача як пацієнта;
- система надаватиме права адміністратору реєструвати користувачів: пацієнта та лікаря;
- система надаватиме можливість лікарю створювати, редагувати та видаляти опитувальник;
- система повинна надавати можливість об’єднувати опитувальники в траєкторії протягом якого проходить лікування;
- система повинна мати можливість проходити опитувальники, продивлятися висновки та результати;
- система матиме можливість генерувати висновки на результатах опитувальників;
- система надаватиме можливість порівнювати результати та висновки з другими опитувальниками.

2. ОГЛЯД МАТЕМАТИЧНИХ МЕТОДІВ ПСИХОЛОГІЧНОГО АНАЛІЗУ

У багатьох областях практичної психології широко використовуються вимірювальні психодіагностичні методики, до яких відносяться опитувальники на вимірювання досягнень, апаратурні методики, здібностей, засновані на стандартизованому самозвіті – опитувальники і техніки суб'єктивного шкалювання [5]. Коректність в застосуванні цих методик забезпечується не тільки змістовними уявленнями, а й виконанням особливих вимог психометрії [6] – вимог надійності, репрезентативності тестових норм, валідності.

Тестові методики призначені на вирішення певних обмежених завдань. Тут не виключені помилки в індивідуальних випадках, діагноз і прогноз даються лише з ймовірною точністю.

Конкретні варіанти емпірико-статистично налагодження опитувальників, якими психолог може скористатися як при наявності сучасної обчислювальної техніки, так і в її відсутність. Ці варіанти дозволяють перевірити, чи працює опитувальник у цілому, якщо він погано диференціює випробовуваних або в яких своїх частинах (завданнях, питаннях) опитувальник “не спрацьовує”. Застосування цих варіант корисно і необхідно не тільки при розробці нових опитувальників, але і при кожній зміні діагностичної ситуації в застосуванні старого опитувальника (наприклад, перехід від добровільного до примусового обстеження), при перенесенні опитувальника з однієї популяції (на якій встановлені тестові норми) на іншу. Це перш за все відноситься до техніки самозвіту, зокрема, до тест-опитувальника, повністю залежних від того, як конкретний пацієнт інтерпретує семантику питань і пов'язує її з суб'єктивної гіпотезою про мету обстеження. Але описані тут алгоритми застосовні й до будь-яких інших опитувальників: результати виконання тестових завдань

також можуть формально кодуватися у вигляді бінарної змінної – 1 (“вирішив”) або 0 (“не розв'язав”).

Тест-опитувальник складається, як правило, з так званих чи-питань, або “так-ні” питання: кожне питання містить твердження, з яким випробуваному пропонується погоджуватися або не погоджуватися. В результаті відповідь випробуваного кодується як дихотомічна змінна зі значеннями “+1” (“вірно”) і “–1” (“невірно”). Бал опитувальника підраховується підсумовуванням відповідей “правильно” на “прямі” пункти опитувальника (це питання, позитивно пов'язані з вимірюваної рисою) і відповідей “так” на “зворотні” пункти (питання, негативно пов'язані з вимірюваною рисою). Алгебраїчно такий найпростіший спосіб підрахунку тестового балу може бути описаний формулою:

$$X_{ik} = \frac{1}{m}\sum_{j=1}^{m} R_{ij} C_{jk} \qquad (1)$$

X_{ik} – бал і-того випробуваного по *k*-тій шкалі (межа);

R_{ij} – відповідь і-того випробуваного на *j*-тий пункт тест-опитувальника;

C_{jk} – ключ (шкальної значення) *j*-того пункту по *k*-тій шкалі;

m – кількість пунктів в *k*-тій шкалі (для яких C_{jk} № 0).

Зазвичай R_{ij} = {–1, + 1} і C_{jk} = {–1, 0, +1}.

Апарат емпірико-статистичного аналізу пунктів відкриває можливість для пристосування методики до конкретних умов її застосування. Ця можливість полягає в модифікації шкальних ключів для окремих пунктів (значень C_{jk}). Для обчислення C_{jk} існує декілька алгоритмів.

2.1. Алгоритм кореляції на основі чотирьох клітинної таблиці

Ідеї цього алгоритму в різноманітних модифікаціях використовувалися в безлічі робіт закордонних і вітчизняних авторів [23]. Розглянемо ситуацію, яка виникає під час відсутності зовнішнього критерію валідності.

В нашій системі терапевт має тест-опитувальник, орієнтований на вимір якоїсь однієї особистісної риси (одновимірний опитувальник) і містить M питань. Про опитуваний контингент не має ніякої апріорної емпіричної інформації, але виходить з припущення, що індивіди в цій вибірці значимо відрізняються між собою за ступенем вираженості даної риси k. Психолог хоче пристосувати тест-опитувальник до вимірювання риси k на даному контингенті за допомогою уточнення шкальних ключів C_{jk}.

Для цього йому доцільно взяти для попереднього психометричного дослідження випадкову вибірку з N індивідів. Для успішного аналізу пунктів потрібно використовувати $M = 50$ і $N = 100$. Менші значення M і N, як правило, призводять до занадто великого відсіву пунктів.

На сформованій вибірці система проводить підрахунок і отримує масив результатів у вигляді прямокутної матриці розмірністю $N \times M$, де по рядках випробовувані, по стовпцях – пункти, на перетині – значення R_{ij} відповідей i-го випробуваного на j-те питання опитувальника.

Для кожного стовпця (для кожного пункту) система на підставі вихідних припущень призначає C_{jk}^0 – вихідні шкальні ключі. Потім для кожного рядка (кожного випробуваного) матриці $|R|$ за формулою (1) підраховується сумарний тестовий бал X_{ik}. Серед всіх $\{X\}$ відшукують 30% випробовуваних з найбільшими значеннями тестового бала і 30% – з найменшими. Вони включаються відповідно в “високу” і “низьку” екстремальні групи. Надалі по кожному пункту враховуються лише відповіді випробовуваних з екстремальних груп. Шкальні ключі C_{jk} уточнюються за допомогою кореляції на основі чотирьох клітинної таблиці між відповідями на пункт і попаданням в екстремальну групу. Для кожного пункту будується матриця пов’язаності 2×2:

Таблиця 1

Матриця пов’язаності

	Екстремальна група	
Відповідь	Висока	Низька
“Вірно”	*a*	*b*
“Невірно”	*c*	*d*

У клітинах матриці 2 × 2 вказуються частоти. Наприклад, в клітці *a* підсумовується частота народження тих випробовуваних, які потрапили в “високу” групу і відповіли “вірно” на даний *j*-й пункт опитувальника, в клітці *b* – число досліджуваних з “низькою” групи, які відповіли “вірно” і т. д. Очевидно, що “хороший” пункт, що володіє високою діскрімінатівністю щодо екстремальних груп, повинен володіти високим контрастом значень *a* і *b*, з одного боку, і одночасно високим контрастом з і *d* – з іншого. При цьому контрасти повинні мати протилежний знак: якщо *a* більше *b* (“вірно” частіше відповідає “висока” група), то має бути менше *a* (“невірно” частіше відповідає “низька” група); навпаки, в разі “зворотних” пунктів, коли різниця *a* – *b* негативна, то різниця *c* – *d* повинна бути позитивна. Існують різні коефіцієнти, які враховують ці вимоги [5, 6].

Ми пропонуємо використовувати найбільш популярний з них φ-коефіцієнт [20]:

$$\varphi_j = \frac{ad - bc - N/2}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} \qquad (2)$$

де *N* – сума всіх елементів таблички: $N = a + b + c + d$ (у разі “зворотних” пунктів, коли різниця ($ad – bc$) негативна, величину, треба до чисельника, навпаки, додати, а не відняти, чисельник формули (2) отримує вид: $ad – bc – N / 2$)

Оскільки при використанні рівних і відомих за обсягом екстремальних груп $a + c = b + d = S$, формула (2) дещо спрощується:

$$\varphi_j = \frac{aN - PS - N/2}{\sqrt{P(N-P)}} \qquad (3)$$

де P – сума відповідей “правильно” на даний пункт: $P = a + b$.

Значимість φ-коефіцієнта встановлюється з наступного наближеного співвідношення:

$$\left|\varphi_{гр}\right| = \sqrt{X^2_{гр} / N} \qquad (4)$$

де: $c^2_{гр}$ – стандартний квантиль розподілу *xi*-квадрат з одним ступенем свободи. Як відомо $c^2 1;\ 0.05 = 3{,}84$ і $c^2 1;\ 0.01 = 6{,}63$.

Таким чином, якщо при вибірці N = 100 обчислене значення фі-коефіцієнта перевищує по модулю 0.26, то це означає, що потрібно знехтувати ймовірністю помилки в 1% можна робити висновок про те, що даний пункт вносить істотний внесок в сумарний бал. Якщо j і 0.26, пункт слід вважати “прямим” (відповідь “вірно” свідчить на користь вимірюваної риси), якщо j – *0.26*, пункт слід вважати зворотним (відповідь “так” свідчить на користь вимірюваної риси).

Після того як для всіх M пунктів підраховані j_j і перевірені на значимість (j зіставлені з $\left|j_{гр}\right|$), психолог виводить нові уточнені значення C^1_{jk}: якщо j_j и $j_{гр}$, то C^1_{jk} = +1; якщо $j_j - j_{гр}$, то C^1_{jk} = –1, якщо $-j_{гр} < j_j < j_{гр}$, то C^1_{jk} = 0 (при підрахунку сумарного балу пункт не враховується).

Якщо всі уточнення C^1_{jk} збіглися з вихідними передбачуваними ключами C^0_{jk}, то можна робити висновок про те, що психометрический експеримент повністю підтвердив, що перевірявся тест-опитувальник за всіма пунктами, з яких він складається.

За відсутності повного збігу потрібно виконати новий цикл обчислення. За уточненим вектор – ключу $<C^1>$ по формулі (1) знову підраховуються сумарні бали для всіх випробовуваних. Потім знову

визначаються екстремальні групи, знову підраховуються коефіцієнт фі для всіх M пунктів опитувальника. (Якщо відразу кілька випробуваних на кордоні між екстремальною і нейтральної групою набрали однаковий бал, то слід всіх випробовуваних з однаковим балом включить в одну групу, домагаючись, щоб екстремальна була ближче до 30%, тоді чисельності екстремальних груп можуть бути не однакові і слід скористатися формулою (2) при підрахунку j_j). Очевидно, що процедура досягає результативної зупинки, якщо $< C^t > = < C^t + 1 >$ тобто коли ключ, отриманий на черговому кроці обробки, збігається повністю (за всіма j) з ключем, отриманим на попередньому кроці.

В даному випадку цей момент можна визначити заздалегідь: шукана умова досягається вже тоді, коли стабілізуються склади «високої» і «низькою» груп. Для оцінки якості пунктів часто застосовують більш трудомістку в обчислювальному відношенні точково-бісеріальну кореляцію. В цьому випадку зупинитися можна при $C_{ik}^{t+1} = C_{ik}^{t}$.

2.2. Алгоритм роздільної кореляції відповідей

Якщо перший з описаних тут алгоритмів є традиційним (для світової психометрія) способом аналізу пунктів, придатним для ручних обчислень так і для автоматизованих обчислень. Даний алгоритм більш ефективний внаслідок меншої кількості ітераційних циклів підрахунку збіжності ключів.

При тому ж рівні структурованості масиву $|R|$ збіжність ключів досягається внаслідок 3-4 ітерацій. Виділений вектор знову ж високо конгруентна (збігається) з першою або з другою головною компонентою одновимірного опитувальника. При факторизації одновимірного опитувальника практично завжди виділяються два чинники: один відповідає вимірюваній властивості, інший – соціальної бажаності

відповіді; сила другого чинника залежить від діагностичної ситуації й рівня підозрілості контингенту досліджуваних.

Алгоритм роздільної кореляції відповідей, подібно дуже складним алгоритмам латентно-структурного аналізу [15], дозволяє враховувати при підрахунку сумарного балу з різною вагою відповіді “вірно” і “невірно”. Формула (1) кілька модифікується:

$$X_{ik} = \frac{1}{m}\sum_{j=1}^{m} C\left(R_{ij}\right) \tag{5}$$

де $C\left(R_{ij}\right)$ – ключ, заданий як вторинна змінна, що приймає різні значення в залежності від того, яке з заздалегідь передбачених значень R_{ij} реалізовано. Така модифікація дозволяє врахувати різну силу відповідей “правильно” і “неправильно”, їх різне діагностичне значення.

На кожному черговому t кроці обчислень по ключу $<C_{ik}^{t-1}>$ і по матриці $|R|$ за допомогою формули (5) підраховуються сумарні бали X_{ik}. Як і в першому алгоритмі, з $\{X^{+}\}$ виділяються “висока” і “низька” групи. Ключ для відповіді “вірно” визначається за формулою:

$$f_j^{+} = \frac{a-b}{a+b} \tag{6}$$

Зрозуміло, що f_j^{+} досягає +1, якщо “правильно” відповідають тільки представники “високої” групи, і – 1, якщо “правильно” відповідають тільки випробовувані з “низькою” групи. Важливість f_j^{+} можна оцінити за допомогою наступного наближеного співвідношення:

$$f_{гр} = \sqrt{X_{гр}^2/\left(x+b\right)} \tag{7}$$

Ключ для відповіді “невірно” визначається формулою:

$$f_j^{-} = \frac{c-d}{c+d} \tag{8}$$

Перевірка значущості f_j^{-} аналогічна з урахуванням підстановки $c + d = a + b$.

Плюс алгоритму в тому, що він дозволяє відшукати значущі зв'язку там, де перший алгоритм фіксує лише незначну кореляцію. Тому алгоритм можна вважати більш ефективним і при відборі пунктів по зовнішньому критерію. Не вимагаючи обчислення ніяких підкореневих виразів.

3. ОПИС РЕАЛІЗАЦІЇ

3.1. Вибір PWA для реалізації веб-застосунку

В реалізації через PWA [26], ми маємо найбільше переваг, якщо порівнювати з desktop-додатками чи web-додатками, як звичайними так і з SPA використовуючи фреймворк. PWA – це також web-додаток, але з використанням певних технологій для досягнення заданих цільових показників, які надають переваги, як над desktop так і над web додатками.

В Progressive Web Application можна виділити два основних пункти, які показують, що це саме PWA – це технології та цільові показники.

Технології:

- Service worker – серце PWA. Проксуючий шар між frontend та backend частинами, що знаходиться в браузері. Всі запити браузера йдуть через нього. Цей поділ на два незалежних шару дозволило зробити перехід звичайного веб сайту до PWA максимально простим.
- Web app manifest – JSON файл декларативно визначає для браузера назву програми, іконку, як буде виглядати PWA та деякі інші параметри. Дозволяє встановити PWA, як окремий додаток на домашній екран смартфона.
- HTTPS – PWA вимагає, щоб всі ресурси сайту передавалися по HTTPS протоколу. SSL сертифікат можна отримати безкоштовно, деякі хостери роблять це за вас. Але критично, щоб на сайті не було посилань на незахищені ресурси – деякі браузері просто не будуть відображати сайт в цьому випадку.
- App shell – це просто скелет графічного інтерфейсу, шаблон. Для прикладу візьмемо середній сайт з хедером, двома колонками та іншим. Грубо кажучи, вирізаємо з сайту контент поточної сторінки і всю динамічну інформацію, все що залишилося – це app shell. Суть в тому, що app shell зберігається

на клієнті і завантажується при запуску програми, а потім завантажується вся динамічна інформація. І поки йде процес завантаження app shell повинен виглядати красиво (лоадери на місяцях і т.п), щоб користувач зміг зрозуміти, що додаток працює та не закрив вкладку в браузері.

- Push notifications – якщо пройтись по сайтам в інтернеті з Chrome DevTools, з відкритим на вкладці Application, то ми побачимо, що більшість сайтів не використовуються PWA технологію. А 90% тих, які користуються технологією, то користуються заради Push Notifications.

Цільові показники:

- надійність – додаток завантажується та відображується одразу же, незалежно від статусу і якості інтернет з'єднання;
- швидкість – взаємообмін даними по мережі відбувається швидко;
- привабливість – робить для користувача досвід роботи з додатком комфортним.

Розглянувши, що таке PWA перейдемо до порівняння з другими вище зазначеними варіантами.

Таблиця 2

Таблиця порівняння PWA

Функції	PWA	Desktop	Web
Можливість використовувати на декількох платформах	+	–	+
Низька вартість у виробництві	+	–	–
Вимагає встановлення	–	+	–

Продовження табл. 2

Вимагає обновлення	–	+	–
Push-сповіщення	+	+	–
“Easy sharing”	+	–	+
Низьке споживання даних	+	–	+
Офлайн використання	+	–	–
Швидший UI	+	–	–

Тепер розглянемо плюси використання PWA:

- Імітує desktop-додатки – PWA можна використовувати так само, як і ваші desktop-додатки, як з точки зору взаємодії, так і навігації.
- Швидка передача даних – передача даних у PWA завжди швидка, завдяки процесу оновлення даних через service workers.
- Безпека – PWA додаток використовує HTTPS протокол, тому інформація не може бути відображена або змінена.
- Google індексація – оскільки це web-додаток з URL-адресами, схожими на звичайних веб-сайтах, їх можна легко індексувати пошуковими системами. Крім того, методи SEO також можна застосувати до PWA.
- Немає процесу встановлення – PWA можна безпосередньо встановити одним натисканням на мобільні та desktop пристрої.
- Незалежність від підключення – ці додатки також доступні в автономному режимі та в низькоякісних мережах. API під назвою “service workers” використовує дані кешовані під час останньої взаємодії з інтернетом, щоб зробити їх доступними також і в режимі офлайн. Таким чином дані доступні при обмеженій або нульовій підключеності до інтернету.
- Push-сповіщення – зазвичай вони пов’язані з desktop додатками,

але вони також є невід’ємною частиною PWA.

- Універсальність – прогресивні принципи, які використовуються для створення PWA, дозволяють йому працювати у всіх браузерах та будь-яких розмірах екрана: будь-то настільний, мобільний, планшетний або будь який майбутній пристрій.

Але у PWA також є і мінуси:

- Витрачає акумулятор – зважаючи на те, що вони написані складними кодами, телефони повинні докладати більше зусиль для інтерпретації коду. Ось чому PWA споживають більше акумулятора ніж звичайні програми.
- Неможливо отримати доступ до різних функцій пристрою – PWA не мають доступу до деяких функцій на вашому пристрої, що робить його менш привабливим ніж нативні додатки. PWA не мають доступу до NFC пристрою, а також Bluetooth пристрою розширеного керування камерою та багато іншого. Все це робить додаток робить менш ідеальним для користувачів.

3.2. Вибір мови програмування для backend частини

Наш додаток буде поділятися на дві частини – це backend та frontend частина. Для кожної із частин потрібно виконати дослідження та запланувати, які технології та мови програмування є популярними та доречними для розробки. Backend частина відповідає на запити користувача із клієнтської частини, при цьому роблячи запити до бази даних, або працюючи, як проксі, роблячи запити до інших сервісів, які не відносяться до нашого додатку.

Будемо розглядати найпопулярніші мови програмування та їх фреймворки, якщо такі наявні для backend частини. Найбільш популярні наразі є Python, Java, Node.js, Go lang, PHP.

3.2.1. Python

Python, безперечно очолює список. Багато хто вважає, що це найкращий варіант для вивчення, як перша мова програмування. Python – це швидка, проста у використанні та у розгортанні мова програмування, яка широко використовується для розробки масштабованих веб-додатків. YouTube, Instagram, Pinterest, SurveyMonkey – це все Python. Розглянемо плюси та мінуси даної мови.

Плюси:

- легке створення та використання класів та об’єктів завдяки ООП характеристиці;
- широка підтримка бібліотек;
- фокус на змозі читання коду;
- має можливість масштабувати навіть найскладніші програми;
- ідеально підходить для побудови прототипів та швидкого тестування ідей;
- з відкритим кодом та з постійно зростаючою аудиторією користувачів;
- забезпечує підтримку безліч платформ і систем;
- дуже простий у навчанні та використанні.

Мінуси:

- не підходить для мобільних обчислень;
- повільніше в силу інтерпретованої мови програмування;
- рівень доступу до бази даних дещо незрілий;
- потік виконання не дуже хороший через GIL.

3.2.2. Java

Java – ще один популярний вибір у великих організаціях, і він залишається таким протягом десятиліть. Мова широко використовується для побудови корпоративних веб-додатків. Як відомо, Java надзвичайно

стабільна, тому багато великих підприємств прийняли її. Також її широко використовується в розробці додатків для Android. Тепер розглянемо мову більш детальніше.

Плюси:

- велика кількість бібліотек з відкритим кодом;
- автоматичне розподілення пам’яті та збирання сміття;
- дотримується парадигм ООП;
- має систему розподілу стеків;
- висока ступінь незалежності платформи завдяки функції JVM;
- ідеально підходить для розподілених обчислень;
- пропонує безліч API для виконання різних завдань, таких як підключення до бази даних, підключення до мережі та аналіз XML;
- підтримує багатопотоковість.

Мінуси:

- відсутність шаблонів обмежує створення якісних структур даних;
- дороге управління пам’яттю.

3.2.3. Node.js

JavaScript широко використовується для розробки інтерактивних додатків для frontend. Наприклад, при натисканні на кнопку, яка відкриває спливаюче вікно, логіка реалізується через JavaScript.

У наші дні багато організацій, зокрема стартапи, використовують NodeJS [30], що є середовищем виконання JavaScript. Node.js дозволяє розробникам використовувати JavaScript для сценаріїв на стороні сервера – дозволяє запускати скрипти на стороні сервера та для створення динамічного контенту на веб-сторінці перед відправкою сторінки до веб-браузер користувача. Отже, тепер у JS ви можете використовувати

єдину мову програмування для скриптів на стороні сервера та клієнта. В цій мові також є плюси та мінуси.

Плюси:

- дуже універсальний;
- регулярні оновлення за допомогою специфікації ECMA;
- багато додатків, таких як Greasemonkey, для розширення функціональності;
- спрощена реалізація;
- багато ресурсів та величезна підтримка людей;
- використовується для побудови великої різноманітності додатків;
- добре працює з іншими мовами програмування.

Мінуси:

- дозволяє лише одиночне успадкування;
- в різних браузерах деякі функції можуть працювати по різному.

3.2.4. PHP

Як і Python, PHP – ще одна мова програмування, розроблена одним розробником як побічний проєкт протягом 90-х. Інженер програмного забезпечення Расмус Лердорф спочатку створив PHP як інтерфейс бінарних файлів, написаних на C для створення динамічних веб-додатків. Пізніше до продукту PHP було додано більше функціональних можливостей, і він органічно перетворився на повноцінну мову програмування.

В даний час PHP – це динамічна мова програмування загального призначення, в основному використовується для розробки веб-додатків на сервері.

З підйомом розробки веб-додатків на базі клієнта, PHP втрачає свою привабливість і популярність. Всупереч поширеній думці, PHP не скоро помре, хоча його популярність поступово зменшиться. Розглянемо всі плюси та мінуси даної мови програмування.

Плюси:

- безліч потужних фреймфорків;
- легко розпочати розробку web-додатків;
- першокласне дебагування за допомогою Xdebug;
- багато інструментів автоматизації для тестування та розгортання програм;
- підтримує ООП та парадигми функціонального програмування;

3.3. Вибір технологій для frontend частини

Так як великого вибору мови програмування для frontend невеликий, тому що ми будемо використовувати single page application та progressive web application, тому наш вибір – це JavaScript. Але хоч і вибір мови у нас не викликає проблем, але ми маємо вибір в frontend фреймворці. Найпопулярніші на даний момент фреймворки: React, Vue, Angular, Ember, Backbone. Порівняємо всі ці варіанти, проаналізувавши їх переваги та недоліки виберемо найкращий варіант.

Таблиця 3

Таблиця порівняння фреймворків

Назва	Тип	Популярність	Складність у вивчені
React	Бібліотека	5/5	5/5
Angular	Фреймворк	3/5	5/5
Ember	Фреймворк	1/5	5/5
Vue	Бібліотека	3/5	3/5
Backbone	Фреймворк	1/5	3/5

Не кожен “фреймворк” на основі JavaScript, які ми вже згадували – це не зовсім фреймворк. Але ми можемо згрупувати в пакет декілька бібліотек

і зробити так, щоб ці не зовсім фреймворки відчували себе справжніми. Ми говоримо про React і Vue, які є обома бібліотеками JS. Backbone, який також лише частково реалізує архітектуру Model-View-Controller (MVC). Але для наших цілей ми будемо подалі називати всі ці бібліотеки також фреймворками. Розглянемо найпопулярніші три фреймворки: React, Vue, Angular.

3.3.1. React

React [28] – це JavaScript-бібліотека з відкритим вихідним кодом для розробки інтерфейсів користувача.

В ReactJS присутній односторонній потік даних, що забезпечує стабільний код. ReactJS дозволяє здійснювати пряму роботу з компонентами та використовує прив'язку даних зверху до низу, щоб гарантувати, що зміни дочірніх структур не впливають на їх батьків. Це робить код стабільним.

Повторне використання компонентів React значно економить час. Інша перевага, яку Facebook представив з React – це можливість використовувати компоненти де завгодно в будь-який час, ще один значущий ефект економії часу.

Розглядаючи роботу дизайнерів. Вони постійно використовують одні й ті ж самі елементи. Якщо цього не зробити, їм доведеться малювати, наприклад, корпоративні логотипи знову і знову. Це досить очевидно: повторне використання – це ефективне проєктування.

Широкий набір інструментів React і Redux. І React, і Redux оснащені гідним набором відповідних інструментів, які полегшують життя розробника. Наприклад, розширення React Developer Tools для Chrome і подібне для Firefox дозволяють досліджувати ієрархії компонентів у віртуальному DOM та редагувати стани та властивості. Також Redux DevTools Profiler дозволяє проглядати всі дії, які були виконані завдяки Redux бібліотеці.

3.3.2. Vue

Vue [29] – це прогресивний фреймворк для створення інтерфейсів користувача. Vue є переваги які роблять його доволі хорошим для розробки фреймворком.

Крихітний розмір. Завантажений zip з фреймворком важить 18 Кб. Через невелику вагу, фреймворк не тільки швидко завантажується та встановлюється, але й позитивно впливають на ваш SEO та UX.

Реактивне двостороннє прив'язування даних. Ще одна перевага в маніпуляціях з DOM – це двостороння прив'язка даних, успадкована Vue від Angular. Двостороння прив'язка даних – це зв'язок між оновленнями даних моделі та відображенням їх. Зв'язані компоненти містять дані, які можна час від часу оновлювати. За допомогою двостороннього прив'язки даних простіше оновлювати пов’язані компоненти.

Однофайлові компоненти та читабельність. Кожен частина вашої майбутньої програми у Vue є компонентом. Компоненти представляють інкапсульовані елементи вашого інтерфейсу. У Vue.js в файлі міститься, як і HTML так і CSS з JavaScript, не поділяючи їх на окремі файли.

Розбиття коду програми насправді є архітектурним підходом, який називається Component Based Architecture (CBA), і він також використовується в Angular та React. Такий архітектурний підхід має багато переваг:

Багаторазове використання компонентів. Інкапсульовані компоненти – це в основному шматки коду, які можна повторно використовувати як шаблони для подібних системних елементів.

Читання коду. Оскільки всі компоненти зберігаються в окремих файлах (і кожен компонент – це лише один файл), код легше читати та розуміти, що полегшує його підтримку та виправлення.

3.3.3. Angular

AngularJS – JavaScript-фреймворк з відкритим вихідним кодом. Призначений для розробки односторінкових додатків. Його мета – розширення браузерних додатків на основі MVC-шаблону, а також спрощення тестування і розробки. Популярність Angular набув через деякі його плюси, та через те, що його написали Google.

Двостороння прив'язка даних. AngularJS був побудований з архітектурою Model-View-Controller. Двостороння прив'язка даних дозволила інженерам скоротити час розробки, оскільки не потрібно було писати додатковий код для постійної синхронізації View та Model компонент.

Ін'єкційна бібліотек. Бібліотеки визначають, як різні фрагменти коду, які взаємодіють між собою та як зміни одного компонента впливають на інші. Зазвичай бібліотеки безпосередньо визначаються в самих компонентах. Так що кожна зміна залежності також вимагає зміни компонентів. За допомогою AngularJS ви можете використовувати інжектори, які визначають залежності як зовнішні елементи, що відокремлюють компоненти від їх залежностей. Ін'єкційна залежність зробила компоненти більш багаторазовими, легшими в управлінні та тестуванні.

Але Angular ми відкинемо для розробки застосунку, тому що в ньому присутні мінуси, які перекривають всі плюси, та через це фреймворк уступає іншим.

Основний мінус в цьому фреймроці – це для міграції з застарілих систем з AngularJS на Angular нових версій потрібен час. Як ми вже говорили, між AngularJS та Angular існує монументальна різниця. Тому ми відкинемо Angular.

3.4. Загальна структура додатку

Вище ми розглянули всі популярні технології та мови для розробки і як ми вище зазначили додаток умовно можна поділити на дві частини: клієнтська частина (frontend) та серверна частина (backend). Кожна з цих частин відповідає за свою логіку в додатку, але це дві не розривні компоненти системи, які тісно пов’язані та спілкуються між собою.

Якщо брати backend частину, то вона буде виконувати наступні функції в нашій системі:

- реалізація за допомогою ORM-бібліотек спілкування із базою даних для запису, зчитування, видалення та оновлення інформації;
- оброблення запитів від клієнтської частини до серверної;
- збір даних від користувачів платформи, з роллю пацієнта;
- аналіз зібраних даних від користувачів;
- генерування висновків на основі даних від користувачів.

Тепер розглянемо frontend частину, в ній будуть реалізовані наступні функції:

- реалізація інтерфейсу користувача, який дозволяє взаємодіяти користувачу з системою;
- форма для збору даних від користувачів для відправлення на сервер;
- показ проаналізованих та зібраних даних опитувальників;
- демонстрування інтерфейсів для моніторингу за психічним станом пацієнтів.

Також детальніше розглянемо структурну схему системи, яка будується з багатьох елементів, сервісів, які пов’язані між собою. Можна розділити систему на декілька типів:

- Основний модуль серверної системи. Цей модуль пов’язую між собою всі інші модулі серверної системи, отримує запити від клієнтської сторони та перенаправляє на інші модулі.

- Клієнтський застосунок – відповідає за показ на стороні клієнта всіх отриманих даних з серверної частини, формує та відправляє всі клієнтські запити до сервера.
- Модуль збору пройдених опитувальників. Даний модуль відповідає за збір даних від користувачів системи в створених опитувальниках, та збереження їх в базу даних для подальшого аналізу даних та генерування висновків.
- Модуль генерування висновків на основі зібраних даних. Модуль відповідає за генерування висновків по кожному створеному опитувальнику для кожного користувача та зберігання в базу даних.
- Модуль аналізу та моніторингу зібраних даних. Модуль відповідає за створення графіків, таблиць та інших даних на основі зібраної інформації, для показу на клієнтській частині.
- Модуль бази даних. Даний модуль підключений до всіх інших та збирає, зчитує, оновлює та видаляє дані, які приходять від клієнтської частини.
- Модуль тестування. Цей модуль відповідає за тестування серверної частини на роботу без помилок, та генерування, відправлення даних без помилок.

Трохи детальніше про модуль бази даних, в модулі реалізовано спілкування, backend частини та її модулів або клієнтської частини, з базою даних в які ми будемо зберігати всю потрібну нам інформацію. Для спрощення спілкування бази даних з модулем ми будемо використовувати ORM технологію. ORM – технологія програмування, яка дозволяє здійснювати запит та обробляти дані з бази даних, використовуючи об'єктно-орієнтовану парадигму. Це фактично створює “віртуальну базу даних об'єктів”, яку можна використовувати з мовою програмування.

Серед плюсів в ORM виділяють:

- наявність явного опису схеми БД, представлене в термінах будь-якої мови програмування, яке знаходиться і редагується в одному місці;
- можливість оперувати елементами мови програмування, тобто класами, об'єктами, атрибутами, методами, а не елементами реляційної моделі даних;
- можливість автоматичного створення SQL-запитів, яка позбавляє від необхідності використання мови для опису структури БД (Data Definition Language) і мови маніпулювання даними (Data Manipulation Language) при проєктуванні БД і зміни її схеми відповідно;
- не потрібно створювати нові SQL-запити при перенесенні на іншу систему управління базами даних, оскільки за це відповідає низькорівневий драйвер ORM;
- ORM позбавляє від необхідності роботи з SQL і опрацювання значної кількості програмного коду, який часто одноманітний і схильний до помилок;
- код, що генерується ORM гіпотетично перевірений і оптимізований, отже не потрібно турбуватися про його тестуванні;
- розвинені реалізації ORM підтримують успадкування та композиції на таблицях;
- ORM дає можливість ізолювати код програми від подробиць зберігання даних.

Ми будемо використовувати одно із самих популярних ORM для нашого PHP фреймворку Symfony – doctrine. Як базу даних ми будемо використовувати реляційну СКБД PostgreSQL. Один з плюсів цієї бази даних – це те що ми можемо згенерувати схему, де буде зображена вся її структура у вигляді entity-relation діаграми (Додаток 1). Діаграма описує

таблиці нашої бази даних, поля кожної з таблиць, зв’язки між нашими таблицями.

В нашій створеній базі даних присутні наступні таблиці: User, Therapist, Patient, Trajectory, Session, Questionnaire, Answer, Question, Report, Graph, Norm, Calculation, які наведені на рис. 3.1.

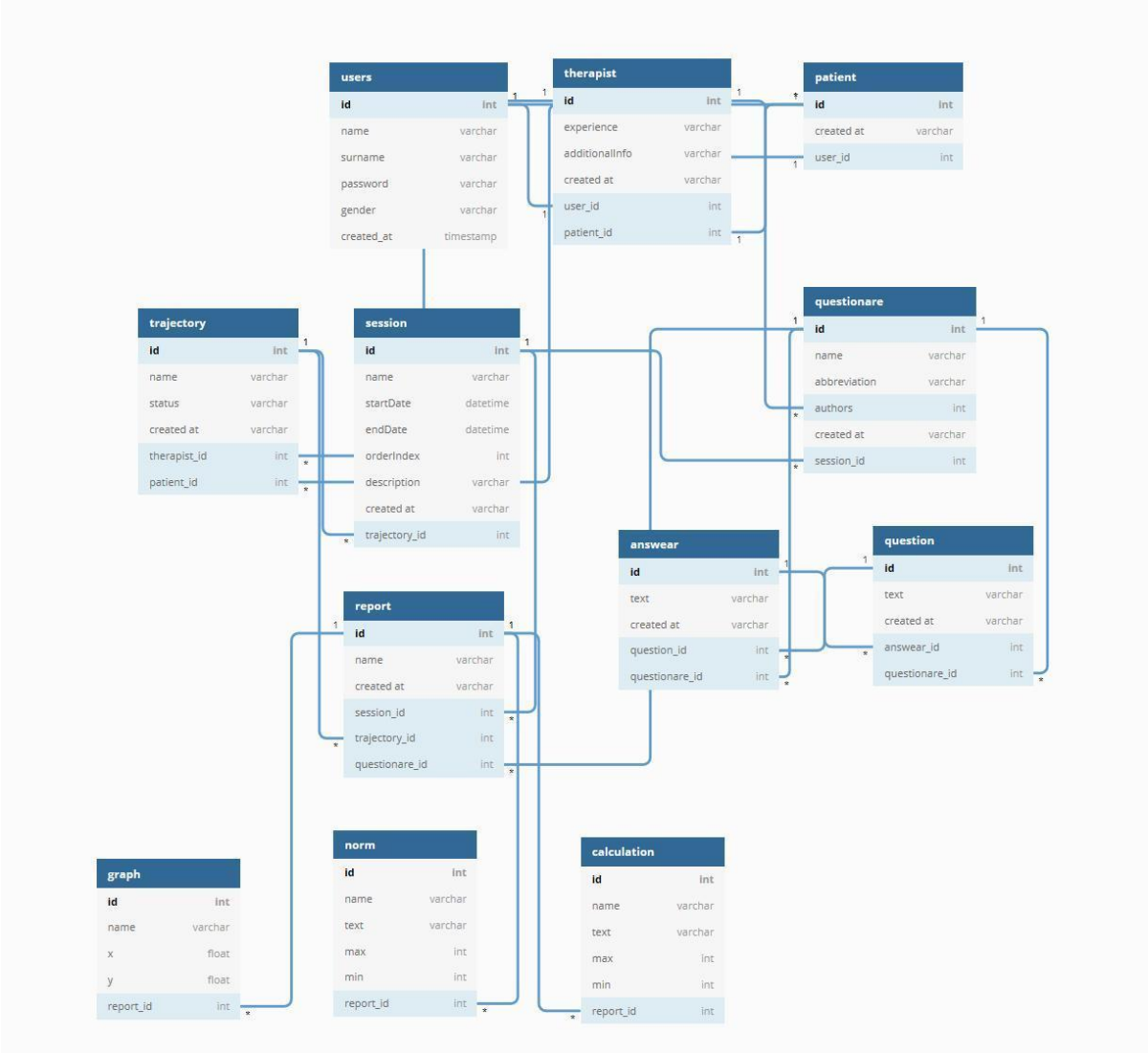


Рис. 3.1. Схема бази даних

Таблиця User зберігає загальні поля для таблиці Patient та Therapist:

- ім’я користувача;
- прізвище користувача;
- логін користувача у системі;

- email користувача;
- пароль профілю;
- стать користувача.

Таблиця Therapist зберігає дані про терапевта, який слідкує за психічним станом пацієнта:

- професійний досвід терапевта, який слідкує за психічним станом пацієнта;
- додаткова інформація, яка може знадобитися пацієнту;
- організація, в якій працює терапевт.

Таблиця Patient зберігає дані про пацієнта, який знаходиться під спостереженням терапевта:

- траєкторії пацієнта в якій зберігаються сесії;
- додаткова інформація, яка може знадобитися терапевту;
- члени родини.

Таблиця Trajectory зберігає дані про лікування пацієнта:

- ім’я траєкторії;
- статус траєкторії на даний момент, може бути закінчена або активна;
- додаткова інформація про траєкторію.

Таблиця Session, в сесіях зберігаються анкети для опитування:

- в сесії присутнє поле ім’я;
- дата початку сесії;
- дата закінчення сесії;
- порядковий номер в траєкторії;
- опис даної сесії.

Таблиця Questionnaire в якій зберігаються питання та відповіді всіх опитувальників:

- ім’я опитувальника;
- абревіатура опитувальника;

- автори – це терапевти системи, які створили даний опитувальник.

Таблиця Answer – таблиця відповідей, яка зв’язана з опитувальниками та питаннями:

- текст відповіді;

Таблиця Questions – таблиця питань, яка прив’язана до таблиць Questionnaire та Answer:

- текст запитання;
- тип запитання.

Таблиця Report, в якій зберігаються згенеровані висновки по всім доступним даним, яка прив’язана до таблиць Norm, Graph, Calculation:

- ім’я висновку;
- прив’язані до нього опитувальники;
- прив’язані до нього опитувальники;
- прив’язані до нього сесії;
- прив’язані до нього траєкторії.

Таблиця Graph, яка зберігає в собі всі графіки в системі:

- ім’я графіку;
- x координата;
- y координата;
- тип графіку.

3.5. Модуль створення опитувальників

В розроблюваному програмному забезпеченні передбачається модуль для створення опитувальників для збору даних пацієнтів та подальшого аналізу. Робота з модулем поділяється на дві частини, як і на стороні клієнта так і на стороні сервера.

Починається з клієнтської частини на якій терапевт за допомогою користувацького інтерфейсу може створити опитувальник для пацієнтів. В

клієнтському інтерфейсі є сторінка з формою на якій користувач може додавати поля для збору даних (Додаток 1). Також на цій формі присутні поля для вибору виду висновку, який буде згенерований після проходження пацієнтом опитувальника. Після того як терапевт зазначив назву опитувальника, та згенерував та заповнив всі питання та види відповідей, всі дані відправляються на сервер.

Після надходження даних до сервера, він їх обробляє та зберігає в базу даних. При надходженні даних, він проводить валідацію всіх надісланих питань, та відповідей, прив’язує опитувальник до певного каркаса висновку в базі даних.

Після цього даний опитувальник можна прив’язати як і до певної сесії в траєкторії користувача, так користувач зможе побачити його, та пройти, відповівши на всі згенеровані питання, після чого, як і терапевт, так і пацієнт, отримають згенерований висновок та зможуть почати відстежувати динаміку розвитку пацієнта.

3.6. Модуль генерування висновків

В розроблюваному додатку також присутнє генерування висновків на основі всіх зібраних даних від користувачів. Тому, що в модулі створення опитувальників у нас модуль поділяється на дві частини на клієнтську та серверну. Також цей модуль використовує модуль бази даних.

Звіти на клієнтській частині генеруються через інтерфейс терапевта, як і в модулі створення опитувальника, та використовуються, як і на інтерфейсі терапевта та пацієнта.

Такі звіти містять в собі дані та зображуються через текст, графіки та діаграми:

- загальні дані про опитувальник, сесію або траєкторію, дату та ім’я;
- пункт відгуків, де знаходиться відгук від терапевта, та загальна градація стану пацієнта;

- розрахунок пунктів та рівень загрози з зображенням через таблицю;
- розрахунок пунктів та рівня загрози з зображенням через графік з прямими або кривими;
- передбачення ризику розвитку, або виявленню захворювань.

Для створення графіків ми будемо використовувати бібліотеку Rechart, тому що вона добре працює з нашим фреймворком, який ми обрали. Rechart (написаний на базі D3.js) – дуже проста та модульна бібліотека. Всі компоненти в бібліотеці призначені для багаторазового використання, тому це спрощує створення та налаштування діаграм.

3.7. Модуль аналізу даних та висновків

У психології аналіз можна поділити на два види. Аналізують за допомогою чітких алгоритмів або евристиці. Евристика – це розумові здібності, які дозволяють людям швидко приймати рішення та розв’язувати проблеми. Цей спосіб можна віднести більше до штучного інтелекту, якщо розглядати, як аналіз за допомогою технологій. Евристика використовується в наступних ситуаціях:

- коли людина стикається із занадто великою кількістю інформації;
- коли час для прийняття рішення обмежений;
- коли рішення, яке потрібно прийняти, неважливо;
- коли є дуже мало інформації, яку ми будемо використовувати для прийняття рішення.

Метод алгоритмів – це формула розв’язання проблеми, яка надає покрокові інструкції, що використовуються для досягнення бажаного результату. Цей метод який дуже добре підходить для аналізу та моніторингу психічного стану пацієнта в нашій системі. Метод дає найкращу точність в аналізі. За допомогою методу алгоритмів підвищується точність і мінімізуються потенційні помилки.

Даний аналіз більш схожий на психологічне консультування, чим на психокорекцію або психотерапії. Основні відмінності:

- ширша сфера застосування в порівнянні з клінічною практикою, спрямованість до проблем психічно здорових людей;
- орієнтація на ширше використання даних, отриманих в емпіричних дослідженнях, організованих по експериментальному плану та використання результатів методів математичної статистики для аналізу;
- орієнтація на велику активність і самостійність в процесі роботи з клієнтом, пробудження внутрішніх психологічних ресурсів людини;
- робота проводиться з ситуаційними проблемами, які можна вирішити на рівні свідомості, на відміну від націленості на глибинний аналіз проблем і роботу з несвідомим в психотерапії;
- велика діалогічність та суб'єкт-суб'єктність спілкування психолога, консультанта та клієнта;
- допустимість в психологічному консультуванні ширшого спектра різноманітних професійних моделей діяльності психолога-консультанта, ніж в психотерапії.

4. АНАЛІЗ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАСТОСУНКУ

4.1. Особливості реалізації програмного застосунку

Front end частина веб застосунку використовує PWA [26] технологію. PWA – це тип програмного забезпечення, що побудований за допомогою загальних веб-технологій, включаючи HTML [22], CSS та JavaScript. Він призначений для роботи на будь-якій платформі, яка використовує браузери. Функціональність включає роботу в автономному режимі, push-сповіщення та доступ до апаратного забезпечення пристрою, що дозволяє створити враження для користувача, подібні до native-програм на настільних та мобільних пристроях. За всі основні функції у PWA відповідають service-workers, більш детально, як працюють service-workers можна побачити на рис. 4.1.

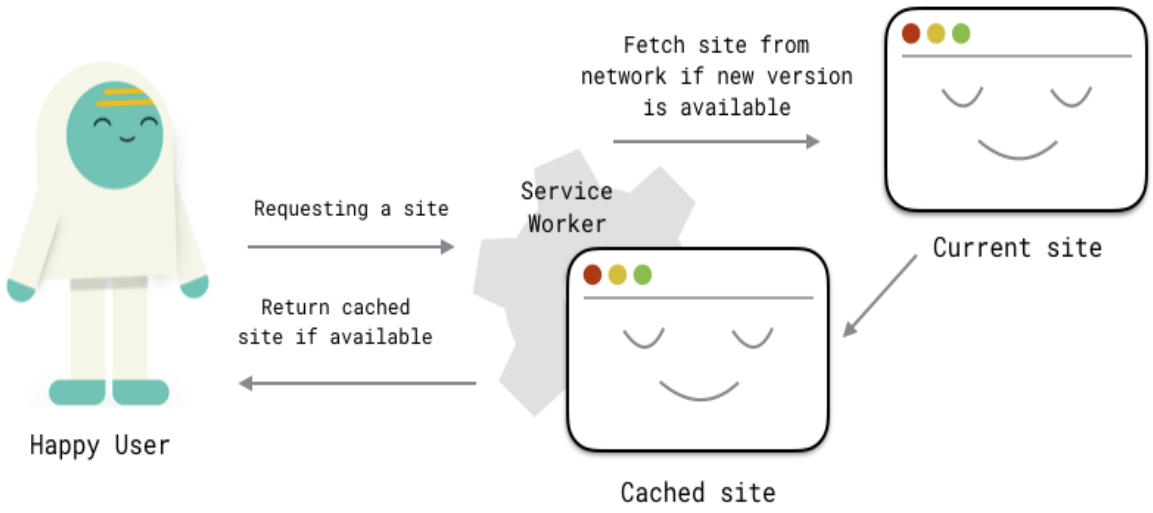


Рис. 4.1. Ілюстрація роботи service-worker

Прогресивні веб-програми – є майбутнім веб-розробки, що дозволяє розробникам веб-додатків створювати досвід користувача, який мається у звичайних веб-додатків та функціональність нативних програм. Як і будь-яка нова технологія, вони мають свої фундаментальні принципи дизайну, але їх переваги роблять їх гідними інвестиціями в використанні.

4.2. Дизайн та інтерфейс користувача

Реалізація інтерфейсу та дизайну та наповнення сторінок веб застосунку було використано наступні технології CSS3, HTML5, JavaScript та фреймворк React. Через те що клієнтська частина була виконана на React з технологією PWA, тому усі сторінки підтримують всі сучасні браузери, мають адаптивний дизайн, що підлаштовується під будь яку роздільність екрану. Через використання React ми також використали технологію SPA, через це у нас клієнтська та серверна частини лежать на окремих серверах. Також у нас на клієнтській частині використовуються бібліотеки для пришвидшення завантаження та відображення елементів для користувача по типу Webpack та Rollup. Сама клієнтська частина поділена на три частини на UI, Data, App. Rollup використовується для мінімізації та збірки UI та Data компонентів. Webpack використовується для збірки всіх частин до одної та зменшення всього додатку.

У веб-застосунку ми маємо наступні сторінки:

- сторінка входу;
- сторінка реєстрації;
- сторінка створення опитувальник;
- сторінка створення висновку;
- сторінка проходження опитувальника;
- сторінка пацієнта;
- сторінка аналізу та моніторингу;
- сторінка пройденого опитувальника;
- сторінка висновку;
- сторінка всіх висновків;
- сторінка всіх опитувальників.

Майже на всіх сторінка пере використовуються одні ті самі компоненти з UI, але на всіх присутній компонент навігаційної частини (рис. 4.2). Користуючись цим компонентом користувач має змогу

завжди повернутися на головну сторінку, також може вийти з системи або налаштувати свій профіль.

Рис. 4.2. Навігаційна частина сайту

При вході на веб застосунок користувач попадає на сторінку входу. Де він бачить два поля для логіну та паролю (рис. 4.3). Також на сторінці користувач може перейти до реєстрації нового користувача.

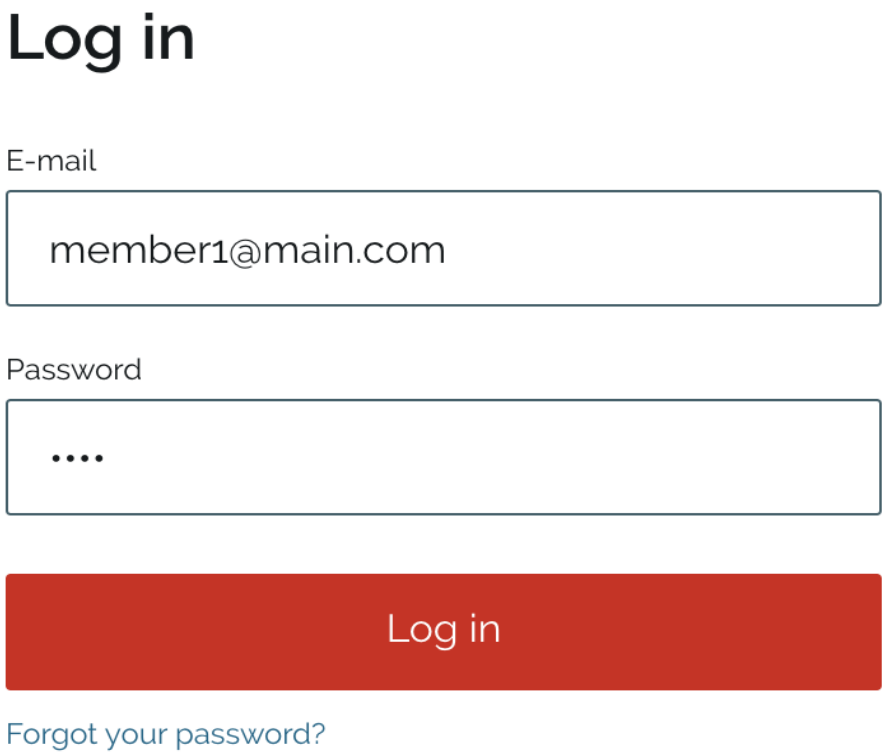
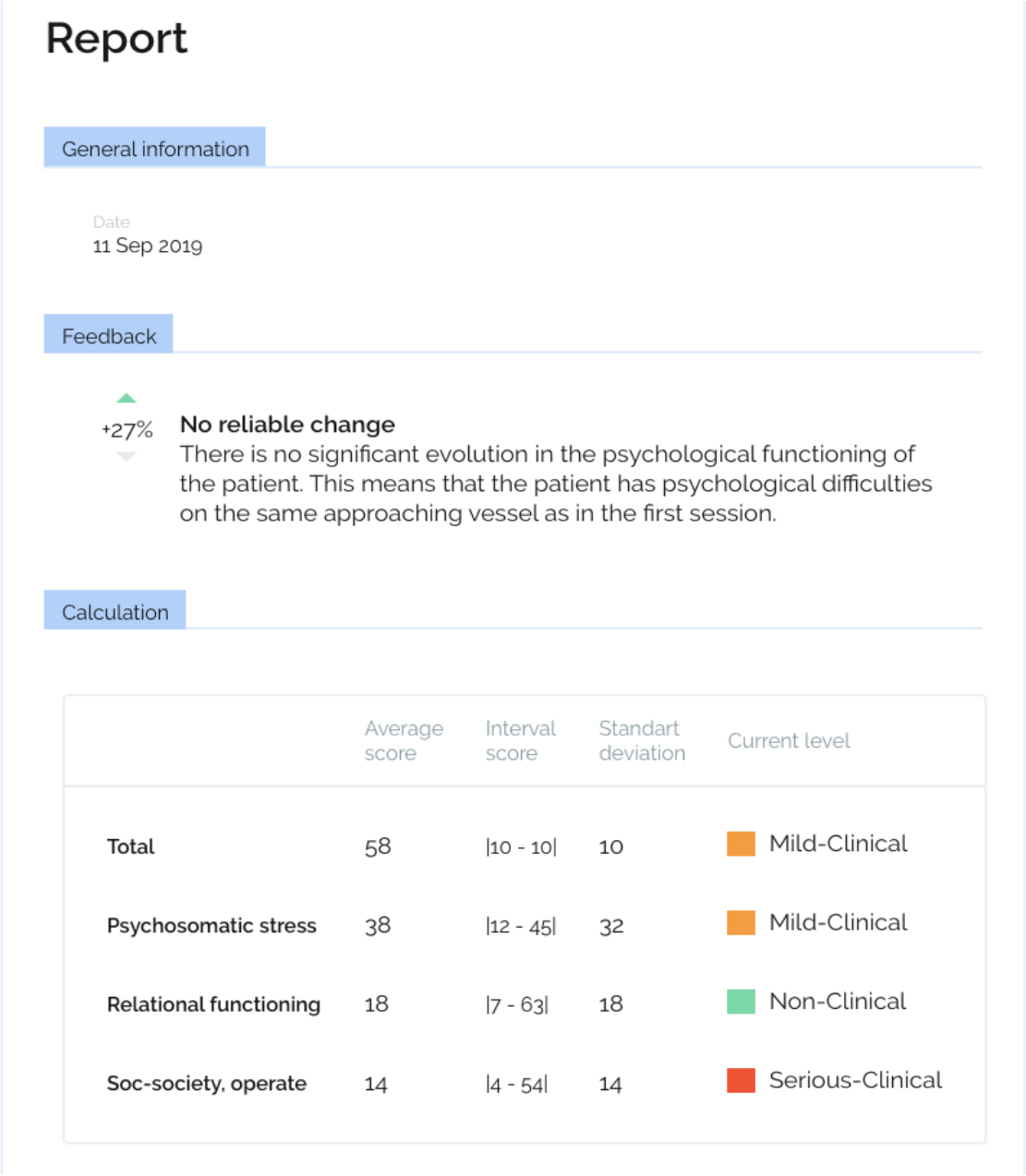


Рис. 4.3. Сторінка входу для користувача

На наступних рисунках (рис. 4.4, 4.5) відображено сторінку з висновками, згенерованими після проходження опитувальника пацієнтом.

Де як і пацієнт так і терапевт може бачити розвиток пацієнта. Терапевт також може продивлятися динаміку та робити висновки.



Report

General information

Date
11 Sep 2019

Feedback

+27%

No reliable change
There is no significant evolution in the psychological functioning of the patient. This means that the patient has psychological difficulties on the same approaching vessel as in the first session.

Calculation

	Average score	Interval score	Standart deviation	Current level
Total	58	|10 - 10|	10	Mild-Clinical
Psychosomatic stress	38	|12 - 45|	32	Mild-Clinical
Relational functioning	18	|7 - 63|	18	Non-Clinical
Soc-society, operate	14	|4 - 54|	14	Serious-Clinical

Рис. 4.4. Верхня частина згенерованого висновку

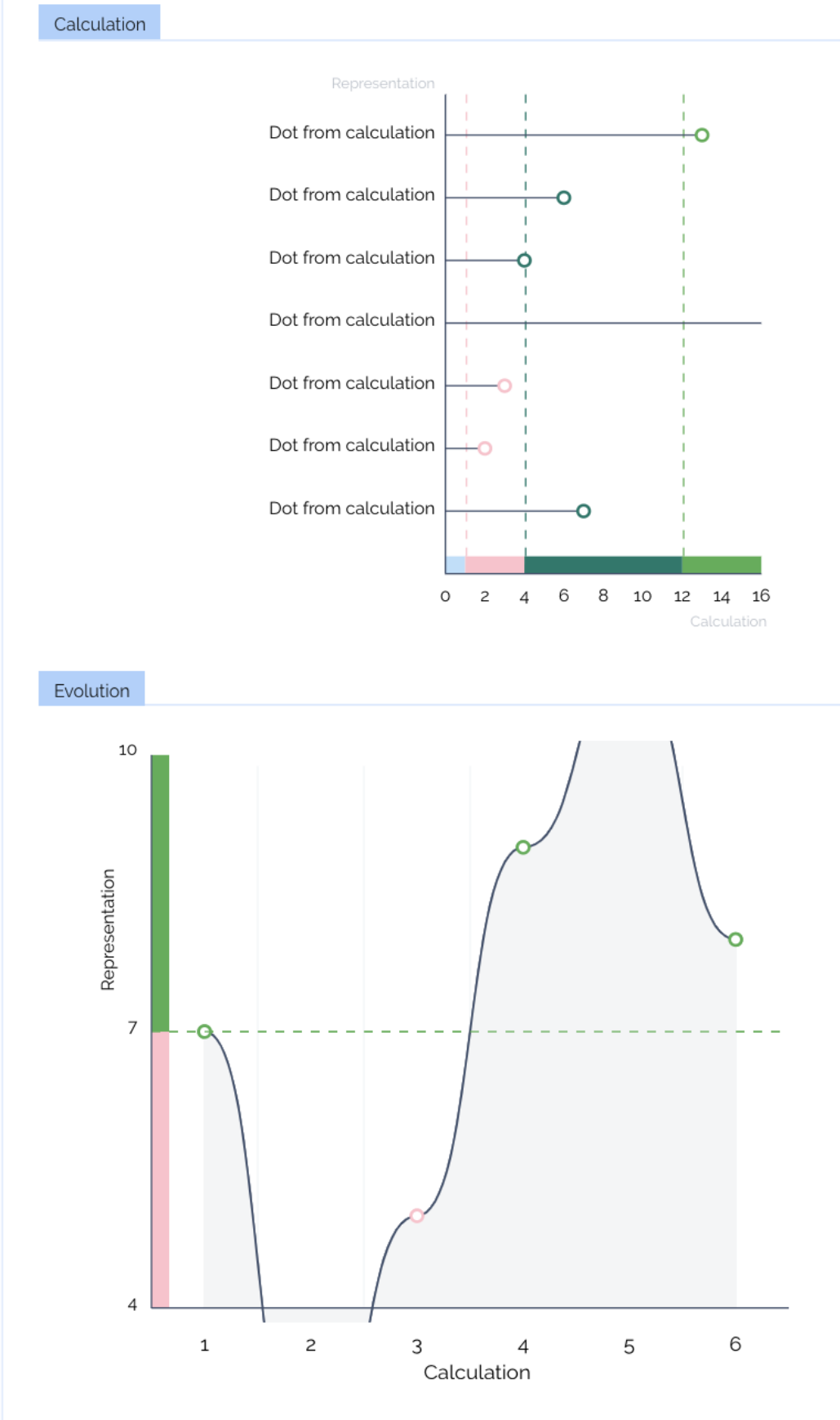


Рис. 4.5. Нижня частина згенерованого висновку

Далі зображено на рисунку 4.6 та 4.7 сторінку для проходження опитувальника, який ми можемо прив’язати до конкретного пацієнта та слідкувати за динамікою розвитку. На основі цього опитувальника генерується конкретний висновок.

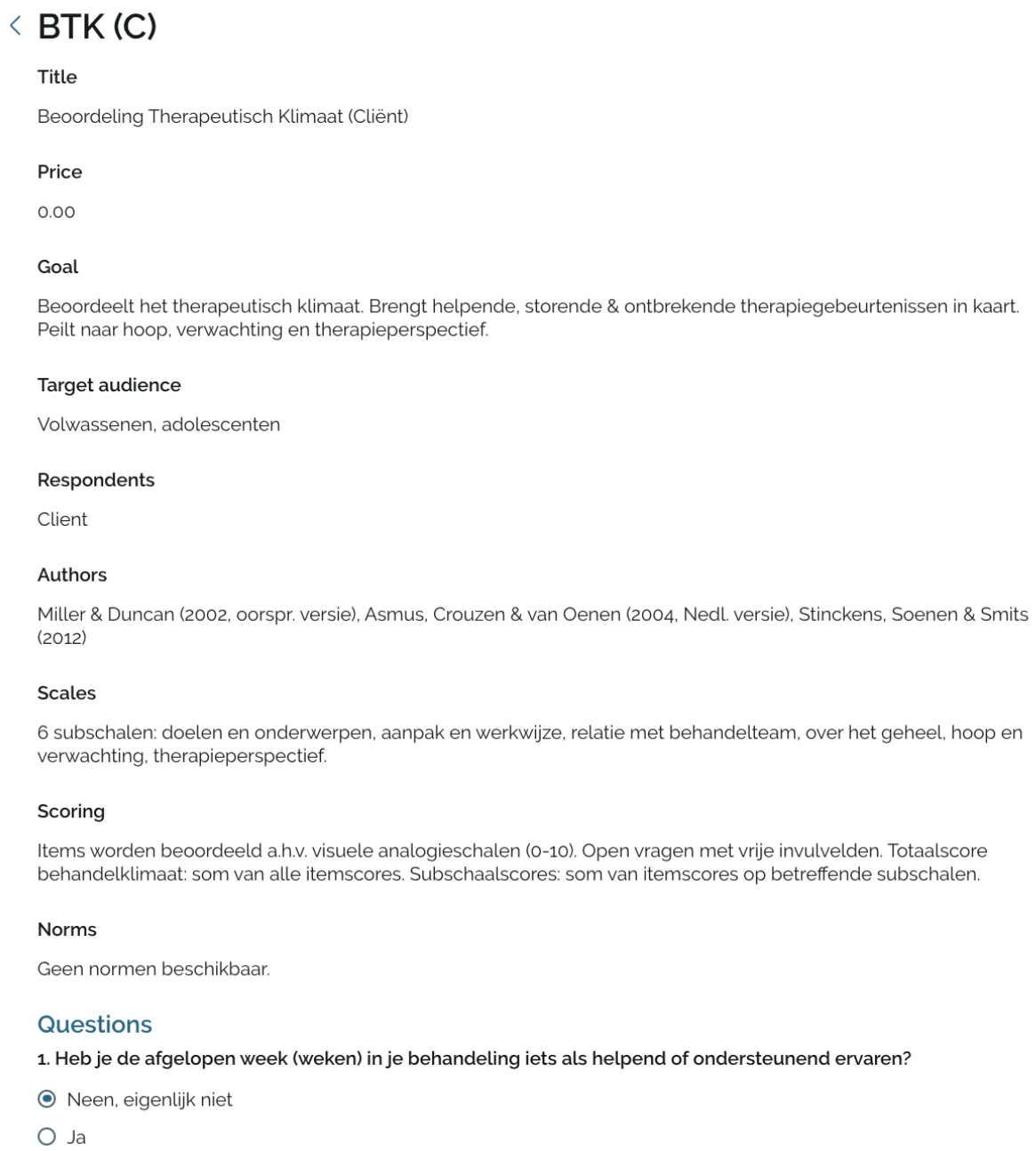


Рис. 4.6. Початок згенерованого опитувальника

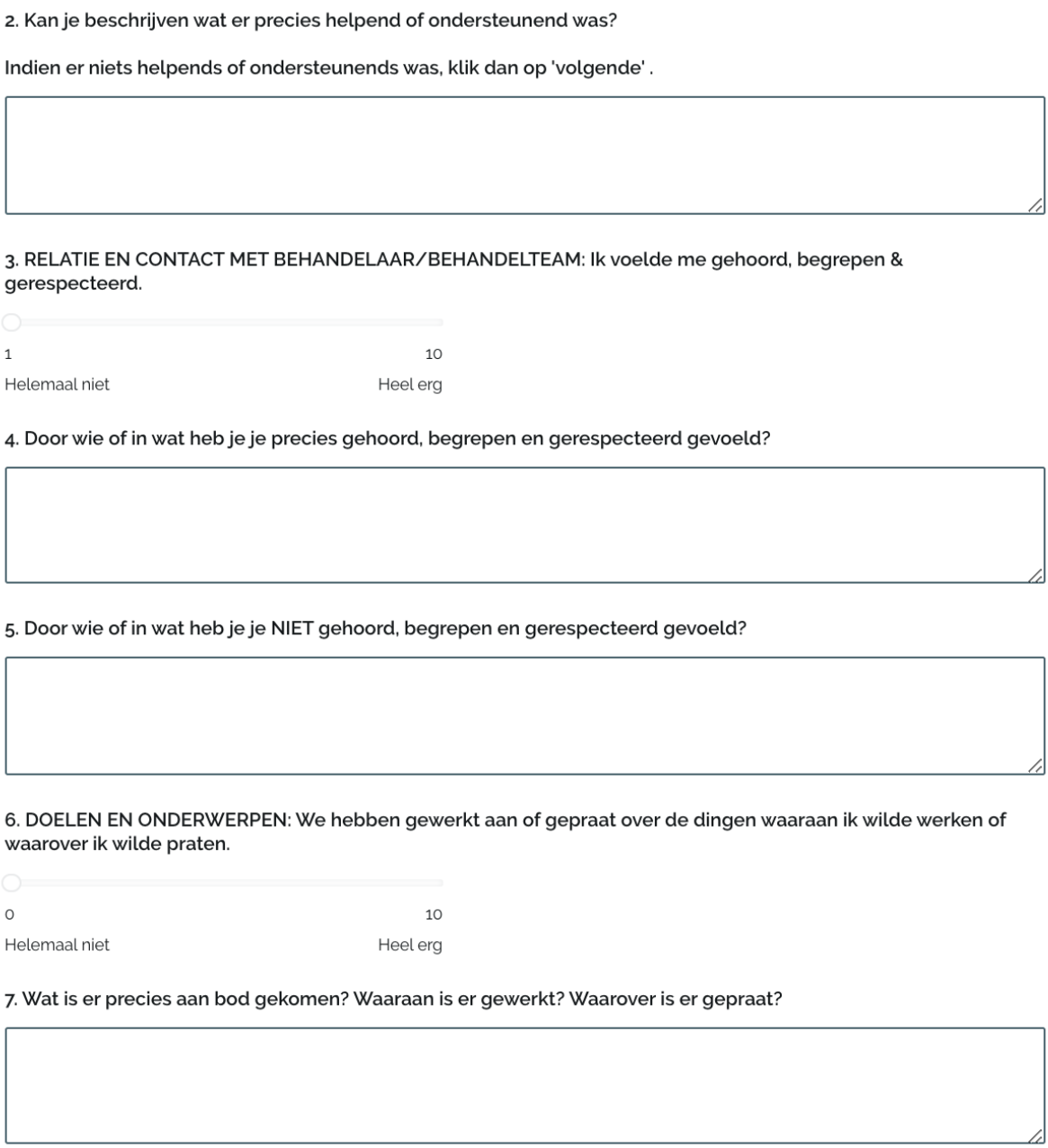


Рис. 4.7. Продовження опитувальника

4.3. Тестування на серверній та клієнтській частині

Тестування системи дуже важливе для даної системи, для валідації опитувальників, генерування висновків, та аналізу пацієнтів. Тестування здійснюється в системі, як і на клієнтській частині для перевірки інтерфейсів так і на серверній для тестування функцій.

Тестування на стороні сервера відбувається за допомогою бібліотеки PHPUnit. PHPUnit – це тестовий фреймворк для мови

програмування PHP. Це екземпляр архітектури xUnit для unit-тестування фреймворків, які виникли з SUnit та стали популярними у JUnit.

Тестування на стороні клієнта відбувається за допомогою платформи Cypress.io (рис. 4.8). Cypress.io – це End-to-end тестування, технологія, яка тестує весь програмний продукт від початку до кінця, щоб переконатися, що flow додатку ведеться так, як очікувалося. Технологія визначає системні залежності продукту та забезпечує всі інтегровані частини, що працюють разом, як очікувалося.

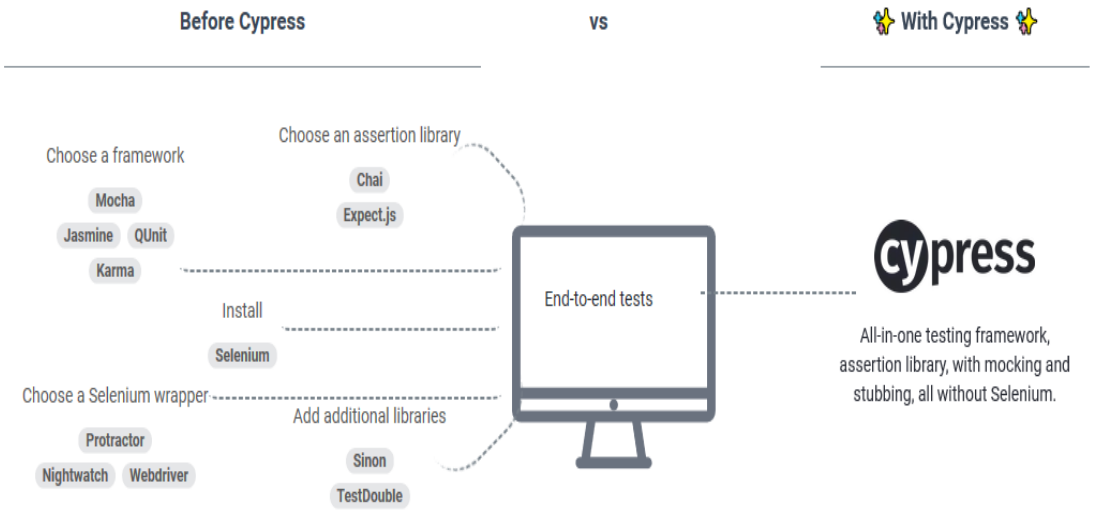


Рис. 4.8. Ілюстрація роботи технології cypress тестування

Чому був обраний cypress:

- cypress працює на мережевому рівні й здатний управляти трафіком додатки;
- легко вбудовується в проєкт;
- має окремий браузер Google Chrome;
- об'єднує інші види тестування в одній платформі;
- здатний звернутися до кожного об'єкту веб-сторінки в DOM.

Тому, що тестування frontend частини часто упускають, а якщо і тестують то тестують за допомогою бібліотек типу Mocha, Chai. Cypress був обраний, тому що тестування зачіпає, як і клієнтську частина, так і серверну та має додаткові плюси:

- Більшість інструментів тестування на основі Selenium, тому всі вони мають однакові проблеми. Щоб Cypress відрізнявся, було побудовано нову архітектуру з самого початку. Тоді як Selenium виконує віддалені команди через мережу, Cypress запускається в тому ж циклі, що і ваша програма.
- Cypress не є загальною системою автоматизації, а також не є базовим тестуванням для ваших сервісів зворотного зв'язку. Там уже є чудові інструменти, які це роблять. Швидше ми спеціалізуємось на одному – створити чудовий досвід, коли ви пишете тести для своїх веб-додатків.
- Cypress тестує все, що працює у веб-браузері. Вся архітектура навколо Cypress побудована для того, щоб особливо добре працювати з сучасними фреймворки JavaScript.
- Хоча ви можете компілювати в JavaScript з будь-якої іншої мови, зрештою тестовий код виконується всередині самого браузера. Немає жодних прив’язок до мови або драйверів – лише до JavaScript.
- Для написання end-to-end тестів потрібно багато різних інструментів для спільної роботи. З Cypress ви отримуєте всі ці інструменти в одному пакеті. Для налаштування тестового набору не потрібно встановлювати 10 окремих інструментів і бібліотек.
- Cypress – це найкраще рішення, коли ви використовуєте його під час створення програми або вже QA тестувальників.
- Cypress був побудований так, що тестування та розробка можуть відбуватися одночасно. Ви можете розроблювати швидше, якщо

будете покривати свій додаток тестами, оскільки: ви можете бачити всю картину програми; зміни відображаються в режимі реального часу. Кінцевий результат полягає в тому, що ви розробили більше, ваш код стане кращим, і він буде повністю перевірений.

ВИСНОВКИ

Метою даного дипломного проєкту було розроблення застосунку для моніторингу та аналізу психічного стану пацієнта.

Аналіз існуючих застосунків, виконаний в даному дипломному проєкті, показав важливість та доцільність в розробці даного застосунку. Для створення системи були використані відповідні технології, а саме: JavaScript для створення веб-клієнту, для відображення даних, які були передані з сервера та для відображення інтерфейсу користувачів. Для реалізації серверної частини було використано мову програмування PHP та для бази даних було використано PostgreSQL. Даний набір технологій забезпечує стабільну роботу додатку.

Розроблений програмний додаток:

- Дозволяє створювати, редагувати, проходити та продивлятися опитувальники для різних користувачів.
- Генерує різні висновки для різних користувачів в залежності від використаного шаблону.
- Генерує різні графіки для моніторингу та аналізу пацієнтів.
- Зрозумілий інтрефейс та дизайн, який працює на будь яких браузерах.

Застосунок був реалізований в повній мірі, були реалізовані всі нефункціональні та функціональні вимоги, було протестована, як серверна так і клієнтська частина.

Система буде використовуватись психологами та психоаналітиками для аналізу пацієнтів та їх психічного стану та для подальшого відстежування динаміки їх розвитку.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Аванесов, В. С. Тесты в социологическом исследовании [Текст] / В. С. Аванесов. – М. : 1982. – 199 с.
2. Анастази, А. Психологическое тестирование [Текст] / А. Анастази. – М. : 1982. – 318 с.
3. Аптон, Г. Анализ таблиц сопряженности [Текст] / Г. Аптон. – М. : 1982. – 143 с.
4. Белнап, Н. Логика вопросов и ответов [Текст] / Н. Белнап, Т. Стил – М. : 1981. – 288 с.
5. Битинас, Б. П. Многомерный анализ в педагогике и педагогической психологии [Текст] / Б. П. Белнап. – Вильнюс, 1971. – 347 с.
6. Гайда, В. К. Психологическое тестирование [Текст] / В. К. Гайда, В. П. Захаров. – Л. : 1982. – 100 с.
7. Губерман, Ш. А. Применение алгоритмов распознавания образов в психодиагностике [Текст] / Ш. А. Губерман, Л. Т. Ямпольский // Вопросы психологии. – 1985. – № 3. – С.118-125.
8. Дюран, Б. Кластерный анализ [Текст] / Б. Дюран., П. Одел. – М. : 1977. – 128 с.
9. Дэвис, Дж. Статистика и анализ геологических данных [Текст] / Дж. Дєвис. – М. : 1977. – 572 с.
10. Ильин, В. И. Сравнительный многомерный анализ личностных особенностей больных с разными формами гинекологической патологии [Текст] / В. И Ильин, В. И. Похилько. // Журнал невропатологии и психиатрии им. С. С. Корсакова – 1983. – №. 12, С. 1836-1840.
11. Кондратьева, А. С. Зависимость тревожности от познавательного стиля у лиц с нормальным и повышенным артериальным давлением [Текст] /

А. С. Кондратьева. – К. : Проблемы медицинской психологии. – М. : 1980.
12. Коффман, А. Введение в прикладную комбинаторику [Текст] / А. Коффман. – М. : 1975. – 474 с.
13. Лазарсфельд, П. Ф. Латентно-структурный анализ и теория тестов [Текст] / П.Ф. Лазарсфельд. – К: Математические методы в социальных науках. – М. : 1973.
14. Миркин, Б. Г. Анализ качественных признаков и структур [Текст] / Б. Г. Миркин. – М. : 1980. – 319 с.
15. Окунь, Я. Факторный анализ [Текст] / Я. Окунь. – М. : 1974. – 199 с.
16. Орлов, Ю. М. Проблема измерения потребностей [Текст] / Ю. М. Орлов. – К. : Прогнозирование социальных потребностей молодежи. – М.: 1978. – 208 с.
17. Davis, F. B. Item-analysis data [Text] / F. B. Davis. – Cambridge : 1949.
18. Edwards, A. L. The measurement of personality traits by scales and inventories [Text] / A. L. Edwards. – N. Y. : Holt, 1970. – 308 p.
19. Garret. Test construction [Text]. – In: Statistics in psychology and education. – N. Y. : 1960.
20. Guilford, J. P. The phi-coefficient and chi-square as indices of item validity [Text] / J. P. Guilford. – Psychometrika, 1941. – 11-19 p.
21. Kline, P. Psychometrics and psychology [Text] / P. Kline. – N. Y. : Acad. press, 1979. – IX, 381 p.
22. HTML [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTML.
23. Шмелев, А. Г. Психометрика многомерного теста [Текст] / А. Г. Шмелев, В. И. Похилько. – К : Практикум по психодиагностике. Дифференциальная психометрика. – М. : 1984, С. 120-135.
24. Шмелев, А. Г. Выявление субъективных группировок личностных черт методом клайк-анализа [Текст] / А. Г. Шмелев. — Новые исследования в психологии. – 1981 – № 1, С. 63-68.

25. Шмелев, А. Г. Введение в экспериментальную психосемантику [Текст] / А. Г. Шмелев. – М. : 1983. – 158 с.
26. Шмелев, А. Г. Надежность тестов [Текст] / А. Г. Шмелев. – К. : Практикум по психодиагностике. Дифференциальная психометрика. – М. – 1984. – С. 52-69.
27. Sofia Jonsson. Progressive Web Applications Under the Hood (PWA) [Електронний ресурс] – Режим доступу до ресурсу: https://dev.to/sofiajonsson/progressive-web-applications-under-the-hood-pwa-pfl.
28. React (javascript framework) [Електронний ресурс] – Режим доступу до ресурсу: https://ru.reactjs.org/docs/getting-started.html.
29. Vue (javascript framework) [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/v2/guide/.
30. Introduction to Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.dev/.
31. Annabel Ness Evans. Methods in Psychological Research [Електронний ресурс] / Annabel Ness Evans, Bryan J. Rooney. – 2013. – Режим доступу до ресурсу: https://books.google.com.ua/books?id=6tpDBAAAQBAJ&dq=testing+is+the+most+powerful+thing+in+psychology&hl=ru&source=gbs_navlinks_s.
32. Kyle Simpson. You don’t know JS [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/azat-io/you-dont-know-js-ru.

ДОДАТКИ

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ТА АНАЛІЗУ ПСИХІЧНОГО СТАНУ ПАЦІЄНТА

Програма та методика тестування

ДП.045440-04-51

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Анастасія ГРЕЧКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитро ПОНОМАРЧУК

2019

ЗМІСТ

1. ОБ'ЄКТ ВИПРОБУВАНЬ

Програмне забезпечення для моніторингу та аналізу психічного стану пацієнта, який являє собою web-сайт, створений на платформі React та PHP з використанням технології PWA.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) функціональна працездатність елементів сторінок web-ресурсу;
2) наявність доступу до бази дистанційних курсів в системі EDU;
3) відповідність форматів та протоколів передачі даних з системою EDU;
4) забезпечення належного рівня безпеки даних;
5) зручність роботи з web-сайтом;
6) відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування на стороні сервера відбувається за допомогою бібліотеки PHPUnit. PHPUnit – це тестовий фреймворк для мови програмування PHP. Це екземпляр архітектури xUnit для unit-тестування фреймворків, які виникли з SUnit та стали популярними у JUnit.

Тестування на стороні клієнта відбувається за допомогою платформи Cypress.io (рис. 4.8). Cypress.io – це End-to-end тестування, технологія, яка тестує весь програмний продукт від початку до кінця, щоб переконатися, що flow додатку ведеться так, як очікувалося. Технологія визначає системні залежності продукту та забезпечує всі інтегровані частини, що працюють разом, як очікувалося.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію SpecFlow.
Працездатність web-ресурсу перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування web-ресурсу в різних web-браузерах;
5) тестування при максимальному навантаженні;
6) тестування стабільності роботи при різних умовах;
7) тестування зручності використання;
8) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ Іван ДИЧКА

"___" _____ 2020 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ТА АНАЛІЗУ ПСИХІЧНОГО СТАНУ ПАЦІЄНТА

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Анастасія ГРЕЧКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитро ПОНОМАРЧУК

2020

ЗМІСТ

1. Опис структури web-ресурсу

Реалізація інтерфейсу та дизайну та наповнення сторінок веб застосунку було використано наступні технології CSS3, HTML5, JavaScript та фреймворк React. Через те що клієнтська частина була виконана на React з технологією PWA, тому усі сторінки підтримують всі сучасні браузери, мають адаптивний дизайн, що підлаштовується під будь яку роздільність екрану. Через використання React ми також використали технологію SPA, через це у нас клієнтська та серверна частини лежать на окремих серверах. Також у нас на клієнтській частині використовуються бібліотеки для пришвидшення завантаження та відображення елементів для користувача по типу Webpack та Rollup. Сама клієнтська частина поділена на три частини на UI, Data, App. Rollup використовується для мінімізації та збірки UI та Data компонентів. Webpack використовується для збірки всіх частин до одної та зменшення всього додатку.

У веб-застосунку ми маємо наступні сторінки:

- сторінка входу;
- сторінка реєстрації;
- сторінка створення опитувальник;
- сторінка створення висновку;
- сторінка проходження опитувальника;
- сторінка пацієнта;
- сторінка аналізу та моніторингу;
- сторінка пройденого опитувальника;
- сторінка висновку;
- сторінка всіх висновків;
- сторінка всіх опитувальників.

2. Опис структури web-ресурсу

Майже на всіх сторінка пере використовуються одні ті самі компоненти з UI, але на всіх присутній компонент навігаційної частини (рис. 2.1). Користуючись цим компонентом користувач має змогу завжди повернутися на головну сторінку, також може вийти з системи або налаштувати свій профіль.

Рис. 2.1. Навігаційна частина сайту

3. Процедура авторизації користувача

При вході на веб застосунок користувач попадає на сторінку входу. Де він бачить два поля для логіну та паролю (рис. 3.1). Також на сторінці користувач може перейти до реєстрації нового користувача.

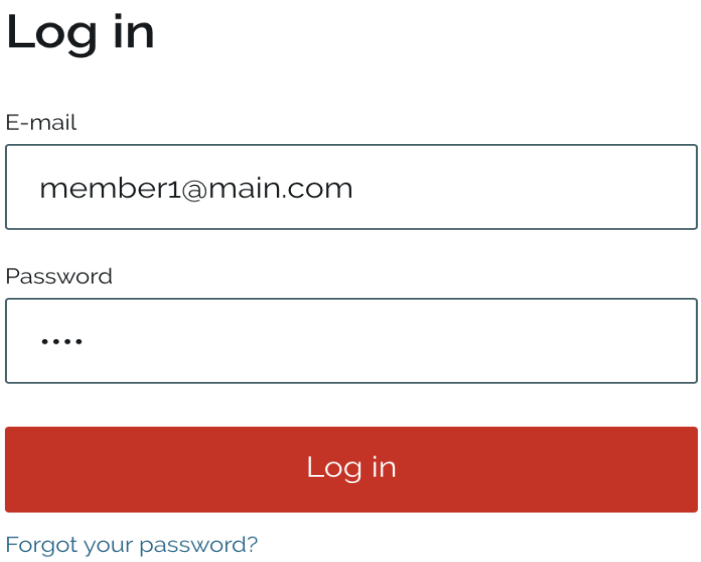


Рис. 3.1. Сторінка входу для користувача

Після вводу користувачем даних для входу, спочатку форма валідується на стороні клієнта, а потім відправляється на сторону серверу

для подальшого глибшого валідування та видання користувачу токена для подальших операцій.

4. Процедура генерування висновків

На наступних рисунках (рис. 4.1, 4.2) відображено сторінку з висновками, згенерованими після проходження опитувальника пацієнтом. Де як і пацієнт так і терапевт може бачити розвиток пацієнта. Терапевт також може продивлятися динаміку та робити висновки.

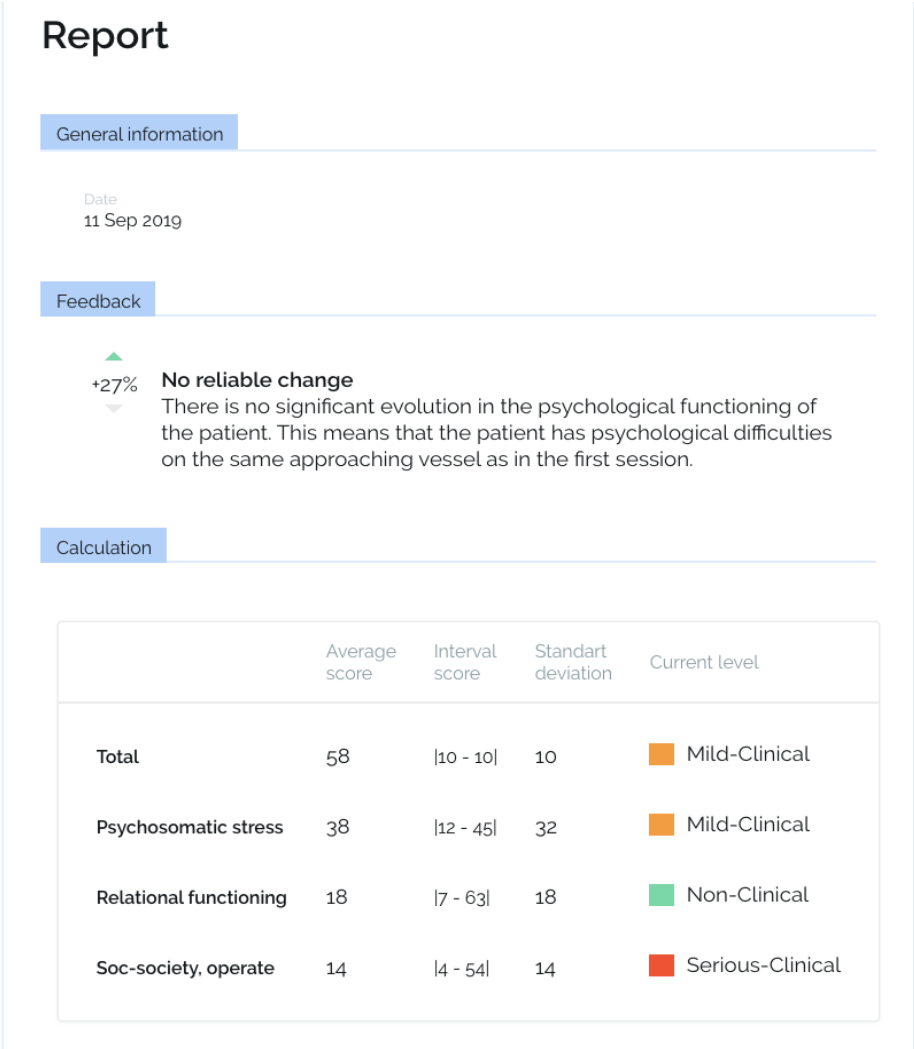


Рис. 4.1. Верхня частина згенерованого висновку

Для генерування висновків терапевту спочатку потрібно створити опитувальник, потім терапевт має прив’язати опитувальник

до конкретного користувача. Після чого користувач може пройти опитувальник, та висновок буде згенерований.

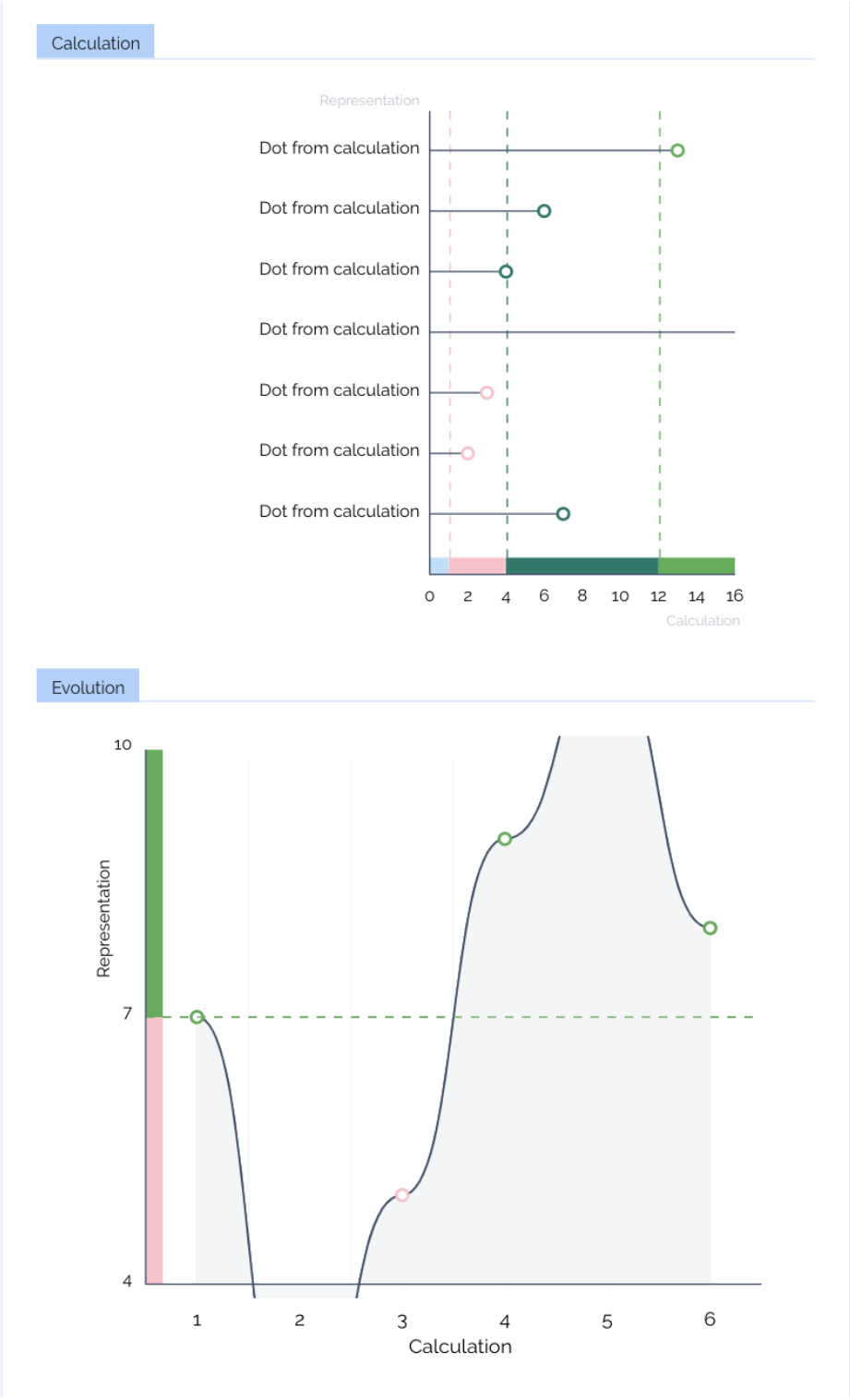


Рис. 4.2. Нижня частина згенерованого висновку

5. Процедура генерування висновків

Далі зображено на рисунках 4.6 та 4.7 сторінку для проходження опитувальника, який ми можемо прив’язати до конкретного пацієнта та слідкувати за динамікою розвитку. На основі цього опитувальника генерується конкретний висновок.

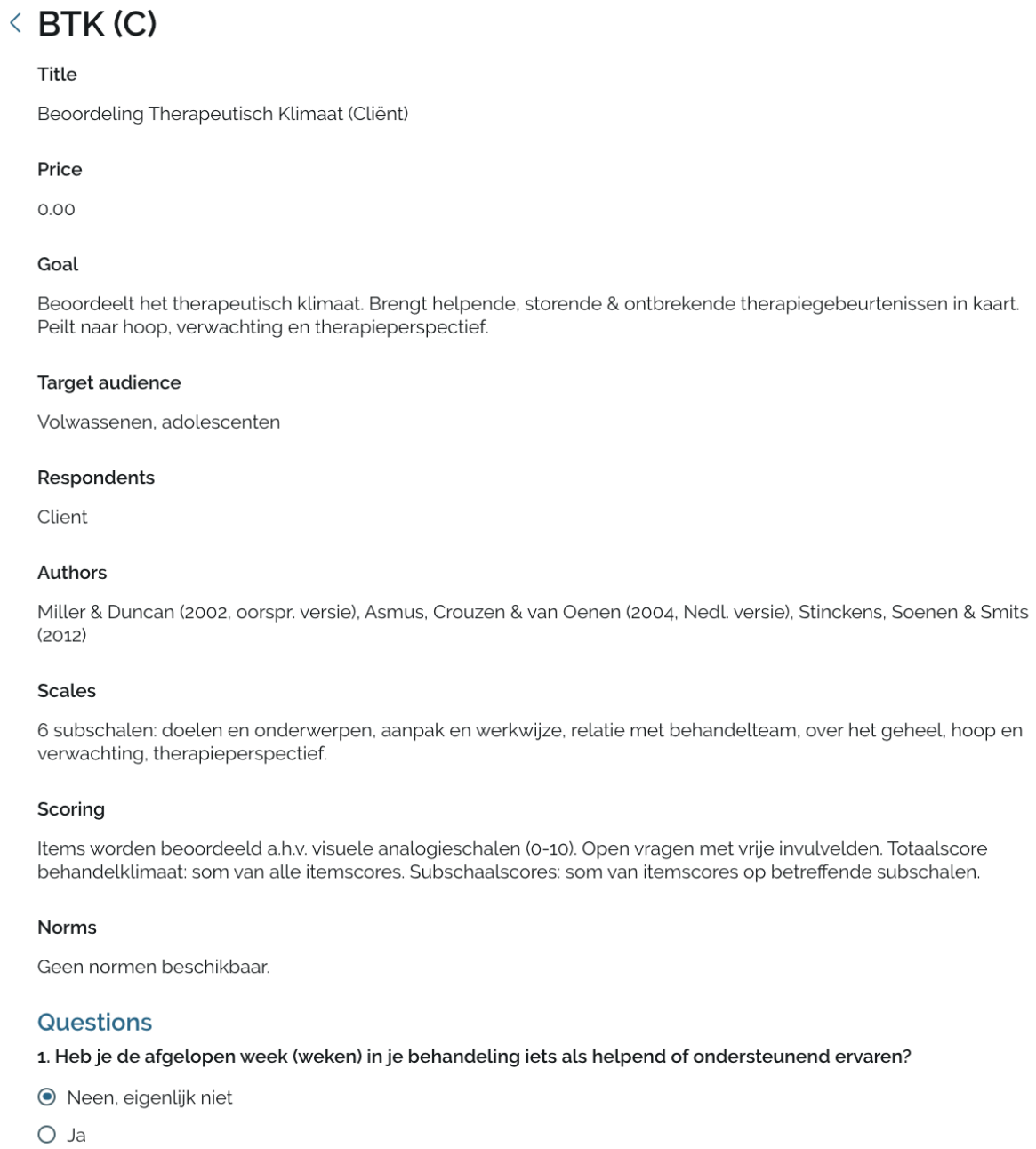


Рис. 4.6. Початок згенерованого опитувальника

2. Kan je beschrijven wat er precies helpend of ondersteunend was?

Indien er niets helpends of ondersteunends was, klik dan op 'volgende' .

3. RELATIE EN CONTACT MET BEHANDELAAR/BEHANDELTEAM: Ik voelde me gehoord, begrepen & gerespecteerd.

1 10

Helemaal niet Heel erg

4. Door wie of in wat heb je je precies gehoord, begrepen en gerespecteerd gevoeld?

5. Door wie of in wat heb je je NIET gehoord, begrepen en gerespecteerd gevoeld?

6. DOELEN EN ONDERWERPEN: We hebben gewerkt aan of gepraat over de dingen waaraan ik wilde werken of waarover ik wilde praten.

0 10

Helemaal niet Heel erg

7. Wat is er precies aan bod gekomen? Waaraan is er gewerkt? Waarover is er gepraat?

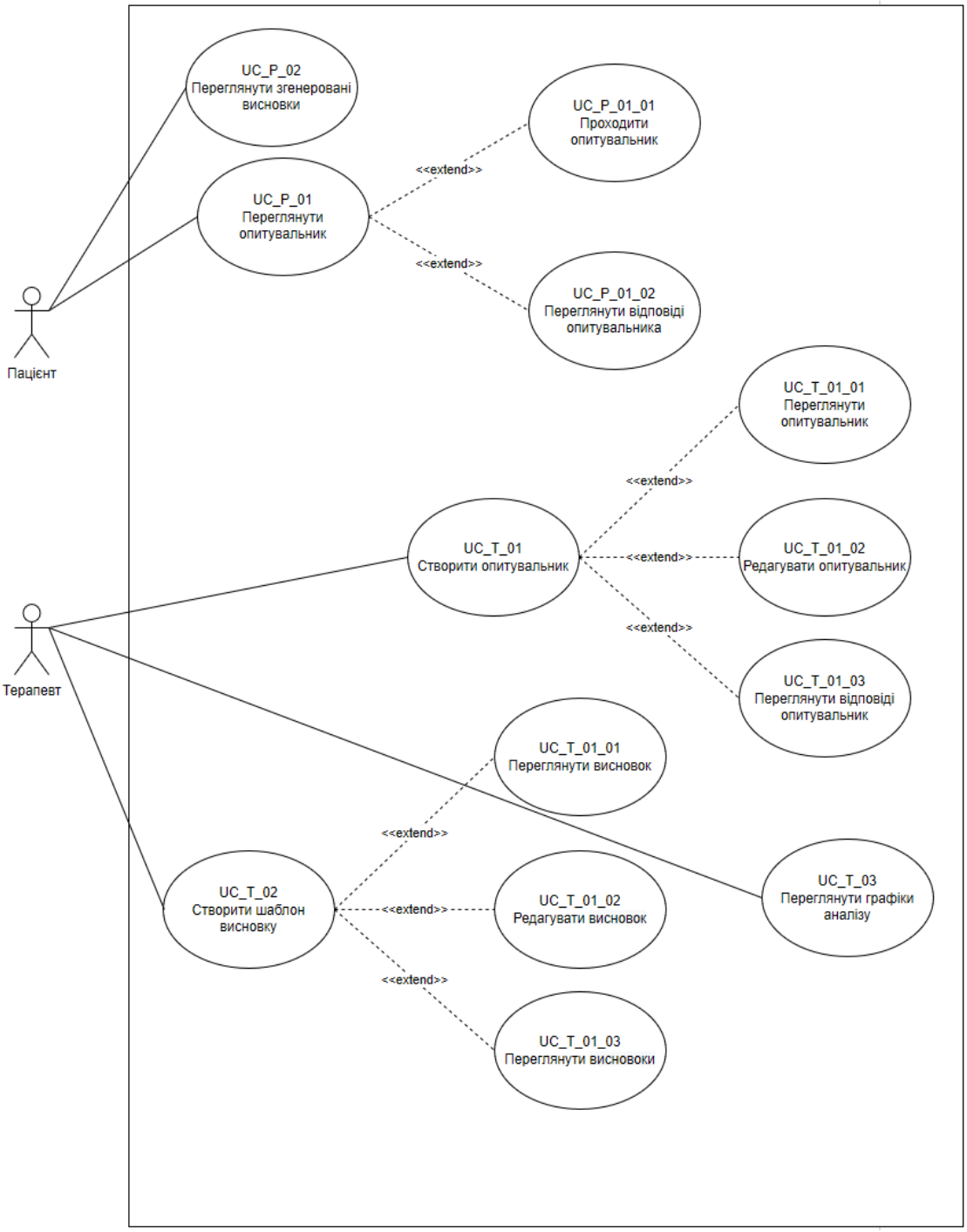
Рис. 4.7. Продовження опитувальника

Додаток 1

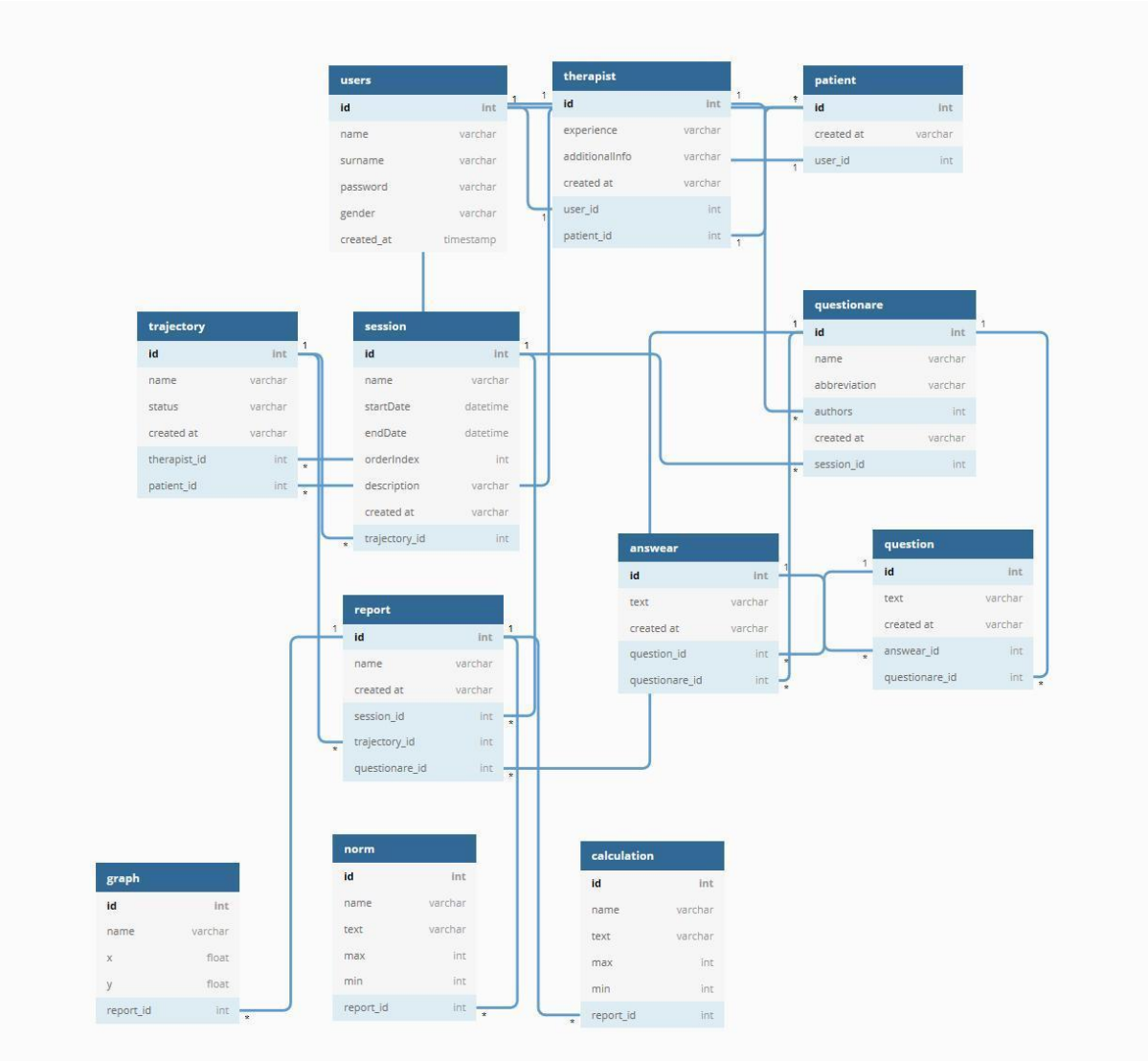
Копії графічних матеріалів

Пономарчук Д.Ю., групу КП-62

Пономарчук Д.Ю., групу КП-62



ДП. 045440-06-99 Програмне забезпечення для моніторингу та аналізу психічного стану пацієнта. Функціональність програмних засобів. UML діаграма прецедентів



ДП. 045440-07-99
Програмне забезпечення для моніторингу та аналізу психічного стану пацієнта. Структура бази даних. ER діаграма

Додаток 2

Лістинг модуля опитувальника

```
   /**Class QuestionnaireController
   * @Rest\RouteResource("questionnaires")
   */
   class   QuestionnaireController   extends   AbstractFOSRestController
implements ClassResourceInterface
   {
   use ProcessFormTrait;
   private $dispatcher;
   public function __construct(EventDispatcherInterface $dispatcher)
   {
   $this->dispatcher = $dispatcher;
   }
   /**
   * Get all questionnaires
   * List of questionnaires.
   *
   * @SWG\Response(
   * response=200,
   * description="Returns the list of questionnaires",
   * @SWG\Schema(
   * type="array",
   * @Model(type=Questionnaire::class)
   * )
   * )
   * @SWG\Tag(name="Questionnaires")
   * @SWG\Parameter(
   * name="id",
   * in="path",
   * type="string",
   * description="Questionnaire id",
   * required=true
   * )
   * @Rest\Route("/questionnaires")
   * @PaginationParams()
   *             @Security("is_granted('ROLE_SUPER_ADMIN')             or
is_granted('ROLE_THERAPIST')")
   *     @Rest\View(serializerGroups={"basic",     "id",     "questions",
"answerOptions", "files"})
   */
   public function cgetAction(?int $limit, int $offset, array $sort, array
$filters): View
   {
   $repo = $this->getDoctrine()->getRepository(Questionnaire::class);
   $filters['active']                =                !$this->getUser()-
>hasRole(User::ROLE_SUPER_ADMIN);
   $qb = $repo->findAllQuery($sort, $filters);
   return $this->view($repo->paginate($qb, $limit, $offset, $sort));
   }
   /**
   * Get single questionnaire
   * @param Questionnaire $questionnaire
   * @return Questionnaire
   * @Rest\Get("/questionnaires/{questionnaire}")
   *
   * Single questionnaire details.
   *
   * @SWG\Response(
```

```
   * response=200,
   * description="Returns a single questionnaire",
   * @SWG\Schema(
   * type="array",
   * @Model(type=Questionnaire::class)
   * )
   * )
   * @SWG\Tag(name="Questionnaires")
   * @SWG\Parameter(
   * name="id",
   * in="path",
   * type="string",
   * description="Questionnaire id",
   * required=true
   * )
   *             @Security("is_granted('ROLE_SUPER_ADMIN')             or
(is_granted('ROLE_THERAPIST') and questionnaire.isActive() === true)")
   *     @Rest\View(serializerGroups={"basic",     "id",     "questions",
"answerOptions", "files" })
   * @return Questionnaire
   */
   public      function      getAction(Questionnaire      $questionnaire,
QuestionnaireEditRule $rule): View
   {
   $questionnaire->setEditable(!$rule->contentOnlyEdit($questionnaire));
   return $this->view($questionnaire);
   }
   /**
   * @SWG\Response(
   * response=204,
   * description="Deletes a questionnaire",
   * )
   * @SWG\Tag(name="Questionnaires")
   * @SWG\Parameter(
   * name="questionnaire",
   * in="path",
   * type="string",
   * description="Questionnaire id",
   * required=true
   * )
   ****************************************
   * @param Questionnaire $questionnaire
   * @param DeletionService $service
   * @Security("is_granted('ROLE_SUPER_ADMIN')")
   * @return View
   * @Rest\Delete("/questionnaires/{questionnaire}")
   */
   public         function         deleteQuestionnaireAction(Questionnaire
$questionnaire, DeletionService $service): View
   {
   $service->delete($questionnaire);
   return $this->view(null, Response::HTTP_NO_CONTENT);
   }
   /**
   * @SWG\Response(
   * response=200,
   * description="Edits a questionnaire",
```

```
   * @SWG\Schema(
   * type="array",
   * @Model(type=Questionnaire::class)
   * )
   * )
   * @SWG\Tag(name="Questionnaires")
   * @SWG\Parameter(
   * name="questionnaire",
   * in="path",
   * type="string",
   * description="Questionnaire id",
   * required=true
   * )
   ****************************************
   * @param Request $request
   * @param Questionnaire $questionnaire
   * @param QuestionnaireEditRule $rule
   * @Security("is_granted('ROLE_SUPER_ADMIN')")
   * @return View
   */
   //*    @Rest\View(serializerGroups={"basic",    "id",    "questions",
"answerOptions" })
   public    function    patchAction(Request    $request,    Questionnaire
$questionnaire, QuestionnaireEditRule $rule): View
   {
   $allowAddAndRemove = !$rule->contentOnlyEdit($questionnaire);
   $form  =  $this->createForm(QuestionnaireType::class,  $questionnaire,
['attr' => ['allowAddAndRemove' => $allowAddAndRemove]]);
   $this->processForm($request, $form);
   return $this->view($form->getData(), Response::HTTP_OK);
   }
   /**
   * Get single questionnaire
   * @return Questionnaire
   * Single questionnaire details.
   * @SWG\Response(
   * response=200,
   * description="Returns a single questionnaire",
   * @SWG\Schema(
   * type="array",
   * @Model(type=Questionnaire::class)
   * )
   * )
   * @SWG\Tag(name="Questionnaires")
   * @SWG\Parameter(
   * name="id",
   * in="path",
   * type="string",
   * description="Questionnaire id",
   * required=true
   * )
   *
   *
@Rest\Get("/therapists/{therapist}/trajectories/{trajectory}/sessions/{s
ession}/questionnaire-container/{container}/questionnaire")
```

```
   * @Security("is_granted('ROLE_SUPER_ADMIN') or (therapist === user and
therapist.getTrajectories().contains(trajectory)                    and
trajectory.getSessions().contains(session))")
   */
   public function getSessionQuestionnaireAction(Therapist $therapist,
Trajectory   $trajectory,   Session   $session,   QuestionnaireContainer
$container): Questionnaire
   {
   return $container->getQuestionnaire();
   }
   /**
   * Get single questionnaire
   * @return array
   * Single questionnaire details.
   *
   * @SWG\Response(
   * response=200,
   * description="Returns a single questionnaire",
   * @SWG\Schema(
   * type="array",
   * @Model(type=Questionnaire::class)
   * )
   * )
   * @SWG\Tag(name="Questionnaires")
   * @SWG\Parameter(
   * name="id",
   * in="path",
   * type="string",
   * description="Questionnaire id",
   * required=true
   * )
   * @PaginationParams()
   *
@Rest\Get("/therapists/{therapist}/trajectories/{trajectory}/sessions/{s
ession}/questionnaires")
   *
   * @Security("is_granted('ROLE_SUPER_ADMIN') or (therapist === user and
therapist.getTrajectories().contains(trajectory)                    and
trajectory.getSessions().contains(session))")
   */
   public function getSessionQuestionnairesAction(
   Therapist $therapist,
   Trajectory $trajectory,
   Session $session,
   QuestionnaireRepository $repo,
   $limit, $offset, $sort): array
   {
   $qb = $repo->findBySession($session->getId());
   return $repo->paginate($qb, $limit, $offset, $sort);
   }
   /**
   * @SWG\Response(
   * response=200,
   * description="Creates questionnaire container",
   * @SWG\Schema(
   * type="array",
   * @Model(type=QuestionnaireContainer::class)
```

```
   * )
   * )
   * @SWG\Tag(name="Questionnaires")
   ****************************************
   * @param Request $request
   * @param Therapist $therapist
   * @param Trajectory $trajectory
   * @param Session $session
   * @return View
   *
@Rest\Post("/therapists/{therapist}/trajectories/{trajectory}/sessions/{
session}/questionnaire-container")
   */
   public function postQuestionnaireContainerAction(Request $request,
Therapist $therapist, Trajectory $trajectory, Session $session)
   {
   $form = $this->createForm(QuestionnaireContainerType::class, (new
QuestionnaireContainer())->setSession($session));
   $this->processForm($request, $form);
   return $this->view();
   }
   /**
   * @param Request $request
   * @param Therapist $therapist
   * @param Trajectory $trajectory
   * @param Session $session
   * @return View
   *
@Rest\Post("/therapists/{therapist}/trajectories/{trajectory}/sessions/{
session}/question-container")
   */
   public function postQuestionContainerAction(Request $request,
Therapist $therapist, Trajectory $trajectory, Session $session)
   {
   $form = $this->createForm(QuestionContainerType::class);
   $this->processForm($request, $form);
   return $this->view();
   }
   /**
   * @param Request $request
   * @param Therapist $therapist
   * @param Trajectory $trajectory
   * @param Session $session
   * @return View
   *
@Rest\Patch("/therapists/{therapist}/trajectories/{trajectory}/sessions/
{session}/questioncontainers/{questionContainer}")
   */
   public function patchAnswerQuestionAction(Request $request, Therapist
$therapist, Trajectory $trajectory, Session $session, QuestionContainer
$questionContainer)
   {
   $form = $this->createForm(QuestionContainerType::class,
$questionContainer);
   $this->processForm($request, $form);
   return $this->view();
   }
```

```
   /**
   * @param Request $request
   * @param QuestionContainer $questionContainer
   * @param QuestionnaireHelperService $helper
   * @param EntityManagerInterface $entityManager
   * @return View
   * @throws \Doctrine\ORM\NonUniqueResultException
   * @Rest\Patch("/answers/question/{questionContainer}")
   * @Rest\View(serializerGroups={"basic"})
   */
   //*             @Security("is_granted('ROLE_PATIENT')             and
questionContainer.getQuestionnaireContainer().getRespondent()==user",
message="this questionnaire is supposed to be answered by another user")
   public function patchAnswerAction(Request $request, QuestionContainer
$questionContainer,          QuestionnaireHelperService          $helper,
EntityManagerInterface $entityManager): View
   {
   $helper->checkQuestionOrder($questionContainer);
   $data = [];
   if ($helper->back($request)) {
   $event = new QuestionReturnEvent();
   $event->setData($questionContainer);
   $this->dispatcher->dispatch($event, $event->getName());
   $nextQuestionIndex = $questionContainer->getQuestionnaireContainer()-
>getPointer();
   $previousContainer                  =                 $entityManager-
>getRepository(QuestionContainer::class)-
>findByOrderIndex($questionContainer, $nextQuestionIndex);
   $answerOptions = $previousContainer->getAnswers()->getValues();
   $previousContainer->reset();
   $data = ['nextQuestionIndex' => $nextQuestionIndex, 'answers' =>
$answerOptions];
   } elseif ($helper->pass($request)) {
   if (true === $questionContainer->getQuestion()->isRequired()) {
   throw new BadRequestHttpException('you are trying to skip question that
is required');
   }
   $event = new QuestionAnsweredEvent();
   $event->setData($questionContainer);
   $this->dispatcher->dispatch($event, $event->getName());
   $nextQuestionIndex                      =                     $helper-
>getNextQuestionIndex($questionContainer);
   $data = ['nextQuestionIndex' => $nextQuestionIndex];
   } else {
   $form        =        $this->createForm(QuestionContainerType::class,
$questionContainer, ['attr' => ['method' => Request::METHOD_PATCH]]);
   $this->processForm($request, $form, new QuestionAnsweredEvent());
   $nextQuestionIndex = $questionContainer->getQuestionnaireContainer()-
>getPointer();
   $data = ['nextQuestionIndex' => $nextQuestionIndex];
   }
   $entityManager->flush();
   return $this->view($data, Response::HTTP_OK);
   }
   /**
   * @param Request $request
   * @param QuestionContainer $questionContainer
```

```
   * @param QuestionnaireHelperService $helper
   * @param EventDispatcherInterface $dispatcher
   * @return View
   * @throws \Doctrine\ORM\NonUniqueResultException
   * @Rest\Post("/answers/question/{questionContainer}")
   */
   public function postAnswerQuestionAction(Request $request,
QuestionContainer $questionContainer, QuestionnaireHelperService $helper,
EventDispatcherInterface $dispatcher): View
   {
   $helper->checkQuestionOrder($questionContainer);
   $form = $this->createForm(QuestionContainerType::class,
$questionContainer);
   $this->processForm($request, $form, new QuestionAnsweredEvent());
   $nextQuestionIndex = $questionContainer->getQuestionnaireContainer()-
>getPointer();
   return $this->view(['nextQuestionIndex' => $nextQuestionIndex],
Response::HTTP_OK);
   }
   /**
   * @param QuestionnaireContainer $questionnaireContainer
   * @param EntityManagerInterface $em
   * @return View
   * @throws \Exception
   * @Rest\Post("/answers/question/{questionnaireContainer}/reset")
   */
   public function postResetAnswersAction(QuestionnaireContainer
$questionnaireContainer, EntityManagerInterface $em): View
   {
   foreach ($questionnaireContainer->getQuestionContainers() as
$questionContainer) {
   foreach ($questionContainer->getAnswerOptions() as $answer) {
   $questionContainer->removeAnswerOption($answer);
   }
   }
   $questionnaireContainer->getSession()-
>setStatus(Session::STATUSES[Session::SESSION_STATUS_SENT]);
   $questionnaireContainer-
>setStatus(QuestionnaireContainer::STATUSES[QuestionnaireContainer::SESS
ION_STATUS_SENT]);
   $questionnaireContainer->setPointer(null);
   $questionnaireContainer->setPath([]);
   $em->flush();
   return $this->view(null, Response::HTTP_OK);
   }
   /**
   * @SWG\Response(
   * response=200,
   * description="Creates a questionnaire",
   * @SWG\Schema(
   * type="array",
   * @Model(type=Questionnaire::class)
   * )
   * )
   * @SWG\Tag(name="Questionnaires")
   ****************************************
   * @param Request $request
```

```
    * @return View
    * @Rest\Post("/questionnaires")
    * @Security("is_granted('ROLE_SUPER_ADMIN')")
    */
    public function postQuestionnaireAction(Request $request): View
    {
    $form = $this->createForm(QuestionnaireType::class);
    $this->processForm($request, $form);
    return $this->view($form->getData(), Response::HTTP_CREATED);
    }
    /**
    * @SWG\Response(
    * response=200,
    * description="Clones a questionnaire",
    * @SWG\Schema(
    * type="array",
    * @Model(type=Questionnaire::class)
    * )
    * )
    * @SWG\Tag(name="Questionnaires")
    * @SWG\Parameter(
    * name="questionnaire",
    * in="path",
    * type="string",
    * description="Questionnaire id",
    * required=true
    * )
    ****************************************
    * @param Questionnaire $questionnaire
    * @param EntityCopierService $service
    * @return View
    * @throws \Exception
    * @Security("is_granted('ROLE_SUPER_ADMIN')")
    * @Rest\Post("/questionnaires/clone/{questionnaire}")
    */
    public     function     cloneAction(Questionnaire     $questionnaire,
EntityCopierService $service): View
    {
    $copy = $service->copy($questionnaire);
    if ($copy instanceof Questionnaire) {
    foreach ($copy->getReports() as $report) {
    $copy->removeReport($report);
    }
    $copy->setName('Copy of '.$copy->getName());
    $this->getDoctrine()->getManager()->persist($copy);
    $this->getDoctrine()->getManager()->flush();
    return $this->view($copy, Response::HTTP_CREATED);
    }
    throw new \Exception('unexpected entity copying result');
    }
    /**
    * @SWG\Response(
    * response=200,
    * description="Download file attached to questionnaire",
    * @SWG\Schema(
    * type="array",
    * @Model(type=Questionnaire::class)
```

```
    * )
    * )
    * @SWG\Tag(name="Questionnaires")
    * @SWG\Parameter(
    * name="questionnaire",
    * in="path",
    * type="string",
    * description="Questionnaire id",
    * required=true
    * )
    * @SWG\Parameter(
    * name="file",
    * in="path",
    * type="string",
    * description="QuestionnaireFile id",
    * required=true
    * )
    ****************************************
    * @param QuestionnaireFile $file
    * @return BinaryFileResponse
    * @Security("is_granted('IS_AUTHENTICATED_FULLY')")
    * @Rest\Get("/questionnaires/{questionnaire}/files/{file}")
    */
    public function getQuestionnaireFileAction(Questionnaire
$questionnaire, QuestionnaireFile $file): Response
    {
    return new BinaryFileResponse($file->getPath().'/'.$file->getName(),
Response::HTTP_OK);
    }
    /**
    * @SWG\Response(
    * response=200,
    * description="Attaches files to questionnaire",
    * @SWG\Schema(
    * type="array",
    * @Model(type=Questionnaire::class)
    * )
    * )
    * @SWG\Tag(name="Questionnaires")
    * @SWG\Parameter(
    * name="questionnaire",
    * in="path",
    * type="string",
    * description="Questionnaire id",
    * required=true
    * )
    ****************************************
    * @param Request $request
    * @param Questionnaire $questionnaire
    * @param QuestionnaireFileCreatorService $creatorService
    * @Security("is_granted('ROLE_SUPER_ADMIN')")
    * @Rest\Post("/questionnaires/{questionnaire}/files")
    * @return View
    */
    public function postFilesAction(Request $request, Questionnaire
$questionnaire, QuestionnaireFileCreatorService $creatorService): View
    {
```

```
   $form = $this->createForm(QuestionnaireType::class);
   $form->handleRequest($request);
   if ($form->isSubmitted() && $form->isValid()) {
   if  (!empty($form->getData()->getDocUploadFiles())  ||  !empty($form-
>getData()->getPdfUploadFiles())) {
   if (!empty($collection = $form->getData()->getDocUploadFiles())) {
   $creatorService->process($collection, $questionnaire);
   }
   if (!empty($collection = $form->getData()->getPdfUploadFiles())) {
   $creatorService->process($collection, $questionnaire);
   }
   $creatorService->flush();
   }
   return $this->view($questionnaire, Response::HTTP_OK);
   }
   return $this->view($form->getErrors(), Response::HTTP_BAD_REQUEST);
   }
   /**
   * @SWG\Response(
   * response=200,
   * description="Deletes files attached to questionnaire",
   * @SWG\Schema(
   * type="array",
   * @Model(type=Questionnaire::class)
   * )
   * )
   * @SWG\Tag(name="Questionnaires")
   * @SWG\Parameter(
   * name="questionnaire",
   * in="path",
   * type="string",
   * description="Questionnaire id",
   * required=true
   * )
   * @SWG\Parameter(
   * name="file",
   * in="path",
   * type="string",
   * description="QuestionnaireFile id",
   * required=true
   * )
   * @param EntityManagerInterface $em
   * @param QuestionnaireFile $file
   * @param Questionnaire $questionnaire
   * @Security("is_granted('ROLE_SUPER_ADMIN')")
   * @return View
   */
   public    function    deleteFileAction(Questionnaire    $questionnaire,
QuestionnaireFile $file, EntityManagerInterface $em): View
   {
   $em->remove($file);
   $em->flush();
   return $this->view(null, Response::HTTP_NO_CONTENT);
   }
   /**
   * @param QuestionnaireContainer $container
   * @return View
```

```
   * @Rest\Get("/answers/{container}")
   *      @Rest\View(serializerGroups={"basic",      "questionContainers",
"containerAnswerOptions",          "questionnaire",          "questions",
"containerQuestion", "answerOptions", "reportItems"})
   */
   public  function  getQuestionnaireAnswersAction(QuestionnaireContainer
$container): View
   {
   return $this->view($container, Response::HTTP_OK);
   }
   /**
   * @param Organisation $organisation
   * @return View
   * @Rest\Get("/usages/{organisation}")
   */
   public     function     calculateOrganisationUsagesAction(Organisation
$organisation, UsagesService $service): View
   {
   return           $this->view($service->calculateAmount($organisation),
Response::HTTP_OK);
   }
   }
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ТА АНАЛІЗУ ПСИХІЧНОГО СТАНУ ПАЦІЄНТА

Виконав: Пономарчук Дмитро Юрійович

Керівник: ст. викл. Гречко Анастасія Валеріївна

Київ – 2020

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: розробити програмне забезпечення для моніторингу та аналізу психічного стану пацієнта.

Завдання:

1. Проаналізувати існуючі рішення-сервіси для створення та проходження опитувальників.
2. Розробити алгоритми для створення, проходження, аналізу та моніторингу опитувальників.
3. Розробити користувацький інтрефейс.
4. Протестувати алгоритми та інтерфейс.

2

АКТУАЛЬНІСТЬ

- Повна автоматизація процесу опитування.
- Автоматизація процесів для моніторингу та аналізу психологічного стану.
- Виключення помилок перевірки опитувальників через людський фактор.
- Економія часу через автоматизацію перевірок.

АРХІТЕКТУРА СИСТЕМИ

- Клієнт-серверна архітектура.
- База даних.
- Модуль збору даних для подальшого аналізу.
- Модуль генерування, проходження опитувальників.
- Модуль аналізу та моніторингу зібраних даних.

АНАЛІЗ НАЯВНИХ РІШЕНЬ

- Аналоги:
  - Questbase;
  - Psychology today;
  - Flexi Quiz.
  - Typeform

- Недоліки:
  - Недостатній функціонал;
  - Не професійна націленість;
  - Автоматизація тільки збору даних.

5

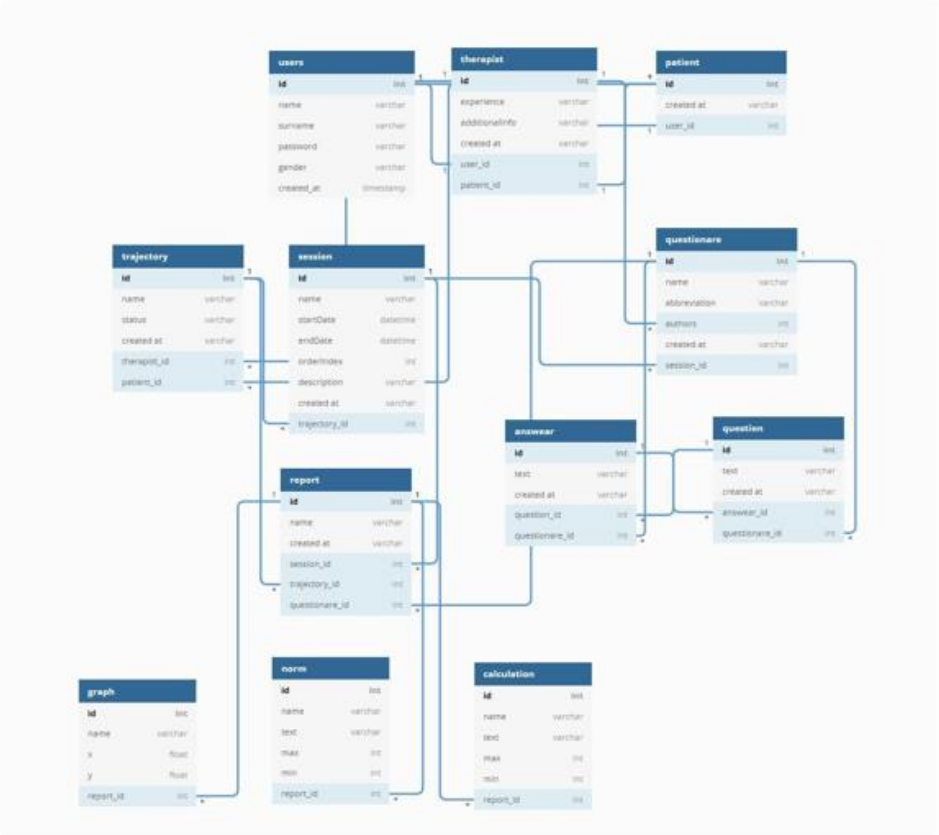
ОБРАНІ ЗАСОБИ РОЗРОБЛЕННЯ

ОБРАНІ ЗАСОБИ РОЗРОБЛЕННЯ

7

ОБРАНІ ЗАСОБИ РОЗРОБЛЕННЯ

РОЗРОБЛЕНІ ЗАСОБИ. СТРУКТУРА БАЗИ ДАНИХ



9

РОЗРОБЛЕНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

BTK (C)

Title

Beoordeling Therapeutisch Klimaat (Cliënt)

Price

0.00

Goal

Beoordeelt het therapeutisch klimaat. Brengt helpende, storende & ontbrekende therapiegebeurtenissen in kaart. Peilt naar hoop, verwachting en therapieperspectief.

Target audience

Volwassenen, adolescenten

Respondents

Client

Authors

Miller & Duncan (2002, oorspr. versie), Asmus, Crouzen & van Oenen (2004, Nedl. versie), Stinckens, Soenen & Smits (2012)

Scales

6 subschalen: doelen en onderwerpen, aanpak en werkwijze, relatie met behandelteam, over het geheel, hoop en verwachting, therapieperspectief.

Scoring

Items worden beoordeeld a.h.v. visuele analogieschalen (0-10). Open vragen met vrije invulvelden. Totaalscore behandelklimaat: som van alle itemscores. Subschaalscores: som van itemscores op betreffende subschalen.

Norms

Geen normen beschikbaar.

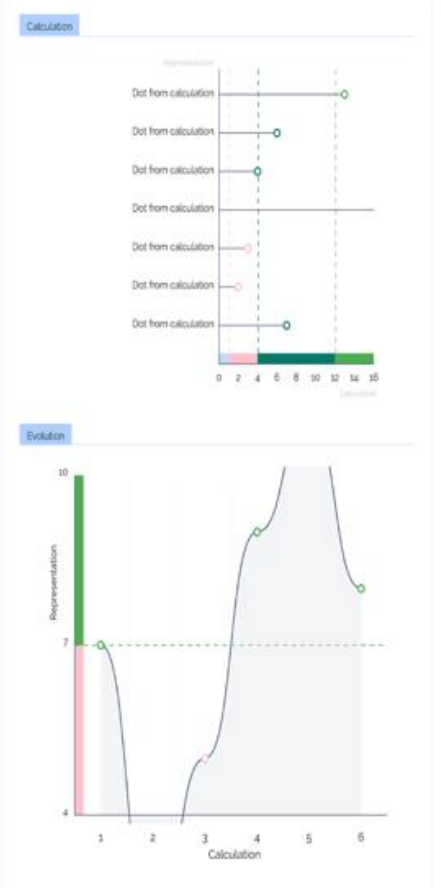
Questions

1. Heb je de afgelopen week (weken) in je behandeling iets als helpend of ondersteunend ervaren?

- (•) Neen, eigenlijk niet
- ( ) Ja

10

РОЗРОБЛЕНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ



ВИСНОВКИ

1. Був проведений аналіз існуючих аналогів, виділено їх переваги та недоліки.
2. Було використано клієнт-серверну архітектуру для реалізації застосунку
3. Були використані сучасні технології для розробки, як і клієнтської так і серверної частини.
4. Були розроблені алгоритми для збору та аналізу даних.
5. Був розроблений веб-додаток для клієнтів.
6. Було проведено тестування всіх модулів системи.

Дякую за увагу!