Міністерство освіти і науки України
Національний університет "Одеська політехніка"
Навчально-науковий інститут комп'ютерних систем
Кафедра інженерії програмного забезпечення

Комлева Олена Олександрівна,
студентка групи АС-171

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Програмна система для аналізу та візуалізації результатів опитувань

Спеціальність:
121 – Інженерія програмного забезпечення

Освітня програма:
Інженерія програмного забезпечення

Керівник:
Зіноватна Світлана Леонідівна,
канд. техн. наук, доцент

Одеса – 2022

ЗМІСТ

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ 4
АНОТАЦІЯ 6
ВСТУП 7
1 КРИТИЧНИЙ АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ 10
1.1 Опис предметної області 10
1.2 Аналіз існуючих програмних аналогів 12
1.3 Висновки до розділу 17
2 РОЗРОБКА ПРОГРАМИ АНАЛІЗУ ТА ВІЗУАЛІЗАЦІЇ РЕЗУЛЬТАТІВ ОПИТУВАНЬ 18
2.1 Створення опитувань 18
2.2 Систематизація видів графічного представлення інформація 19
2.3 Аналіз і класифікація засобів візуалізації текстової інформації 20
2.4 Висновки до розділу 24
3 СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ 25
3.1 Функціональні вимоги до програми 25
3.2 Нефункціональні вимоги до розроблюваної програмної системи 39
3.3 Висновки до розділу 47
4 ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ 48
4.1 Проектування архітектури програми 48
4.2 Основні діаграми роботи програмної системи 50
4.3 Розробка структури бази даних 55
4.4 Проектування структури класів 61
4.5 Висновки до розділу 65
5 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ОПИТУВАНЬ 66
5.1 Набір інструментальних засобів розробки 66
5.2 Реалізація інтерфейсу користувача 69
5.3 Висновки до розділу 74

6 ТЕСТУВАННЯ РОБОТИ ПРОГРАМИ АНАЛІЗУ ТА ВІЗУАЛІЗАЦІЇ РЕЗУЛЬТАТІВ ОПИТУВАНЬ 75
6.1. Сценарії функціонального тестування 75
6.2 Експериментальна перевірка зручності роботи 77
6.3 Висновки до розділу 78
ВИСНОВКИ 79
СПИСОК ЛІТЕРАТУРИ 80
ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ 82

Міністерство освіти і науки України
Національний університет "Одеська політехніка"
Навчально-науковий інститут комп'ютерних систем
Кафедра інженерії програмного забезпечення

Рівень вищої освіти: другий (магістерський)
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Комлева Н. О.
«___» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Комлевої Олени Олександрівни, група АС-171

Тема роботи: Програмна система для аналізу та візуалізації результатів опитувань
Керівник роботи: Зіноватна Світлана Леонідівна, канд. техн. наук, доцент
затверджені наказом ректора від «20» жовтня 2022 р. № 399-в.

1. Зміст роботи: критичний аналіз предметної області та програмних аналогів; створення опитувань, систематизація видів графічного представлення інформації, побудова засобів візуалізації текстової інформації; специфікація функціональних та нефункціональних вимог; проектування архітектури системи, UML діаграми роботи програмної системи, проектування структури бази даних, проектування програмних класів; вивчення та видів інструментальних засобів розробки, програмна реалізація інтерфейсу користувача; функціональне тестування роботи програмної системи з використанням сценаріїв тестування, визначення списку помилок системи, проведення експерименту щодо зручності користування системою та обробка результатів експерименту.
2. Перелік ілюстративного матеріалу: згідно зі слайдами презентації.

3. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

4. Дата видачі завдання: «01» вересня 2022р.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	Аналіз існуючих рішень	01.09.2022 – 14.09.2022	вик.
2	Розробка системи аналізу та візуалізації результатів опитувань	15.09.2022 – 01.10.2022	вик.
3	Специфікація вимог до програми	02.10.2022 – 13.10.2022	вик.
4	Проектування програмної системи аналізу та візуалізації результатів опитувань	14.10.2022 – 28.10.2022	вик.
5	Програмна реалізація системи	29.10.2022 – 15.11.2022	вик.
6	Тестування системи та проведення експериментів	16.11.2022 – 25.11.2022	вик.
7	Оформлення пояснювальної записки та графічного матеріалу	26.11.2022– 05.12.2022	вик.

Здобувач вищої освіти _____ О.О. Комлева

Керівник роботи _____ С. Л. Зіноватна

АНОТАЦІЯ

Метою роботи є підвищенні зручності аналізу та візуалізації результатів опитувань шляхом розробки та використання програмної системи з сучасним інтерфейсом та розвинутими можливостями.

Методи розробки базуються на мові програмування Java для реалізації серверної частини, засобам JavaScript з бібліотекою Redux, HTML та CSS для клієнтської частини, середовищі розробки IntelliJIDEA, реляційній базі даних MySQL, фреймворку Bootstrapта системі контролю версій Git.

Як результат роботи розроблено програмну систему, що дозволяє проводити опитування та візуалізувати результати відповідно до обраного користувачем виду представлення.

Ключові слова: аналіз опитувань, візуалізація опитувань, сервер, Java, JavaScript, Redux, IntelliJIDEA, клієнт, HTML, CSS, MySQL, фреймворк, Git.

ABSTRACT

The purpose of the work is to increase the convenience of analysis and visualization of survey results by developing and using a software system with a modern interface and advanced capabilities.

The development methods are based on the Java programming language for the implementation of the server part, JavaScript tools with the Redux library, HTML and CSS for the client part, the IntelliJIDEA development environment, the MySQL relational database, the Bootstrap framework and the Git version control system.

As a result of the work, a software system was developed that allows you to conduct surveys and visualize the results according to the type of presentation chosen by the user.

Keywords: survey analysis, survey visualization, server, Java, JavaScript, Redux, IntelliJIDEA, client, HTML, CSS, MySQL, framework, Git.

ВСТУП

Опитування є дієвим засобом для збору інформації та розуміння важливих тенденцій та думок респондентів стосовно будь-якого процесу та сфери застосування. Важливим поняттям при роботі з опитуваннями є їх життєвий цикл.

Життєвий цикл опитування охоплює процеси з моменту прийняття рішення про використання методів опитування до ретельного аналізу даних опитування та розповсюдження результатів. Життєвий цикл опитування включає визначення елементів даних, які необхідно зібрати для вирішення гіпотез дослідження, формулювання запитань для виявлення елементів даних, перегляд і перевірку питань опитування, впровадження опитування (наприклад, створення форми анкети для розсилки, кодування опитування онлайн), перевірка та підтвердження інструменту опитування, проведення опитування, керування та аналіз даних опитування та забезпечення того, щоб необхідні сторони мали дані та результати опитування.

Ефективна робота системи опитування неможлива без виконаного належним чином архітектурного проектування. Система з надійною та логічною архітектурою має перевагу в досягненні своїх цілей, оскільки існує структура, яка дозволяє елементам системи функціонувати гармонійно. Подібним чином проект опитування з надійною та логічною архітектурою має перевагу в досягненні своїх дослідницьких цілей.

Архітектура проекту опитування включає такі міркування, як спосіб залучення групи потенційних респондентів до участі та як ці учасники нададуть необхідні дані. Наприклад, для деякого дослідження може бути визначено, що ідеальний спосіб залучити респондентів до участі – це як надіслати електронний лист, так і надати посилання на опитування. Вміст цього електронного листа та текст, пов’язаний із посиланнями на опитування, повинні спонукати потенційного респондента натиснути посилання та почати перегляд опитування. Інакше може бути неважливим, наскільки вражаючим є опитування, якщо потенційний респондент ніколи не побачить опитування.

Кожна анкета опитування повинна мати архітектурний дизайн, незалежно від того, кастомізований цей дизайн чи ні. Хороший архітектурний дизайн може покращити організацію та проведення опитування. Оскільки респонденти є зацікавленими сторонами в проведенні опитування, якісно організована форма опитування, яка протікає гладко, може дати більший відсоток відповідей, ніж та, яка складніша для респондентів. Важливо згрупувати пов'язані питання разом і надати пояснювальну інформацію таким чином, щоб вона була доступною, коли респонденти відповідають на кожне запитання. У тематичному дослідженні набір основних термінів використовується протягом усього опитування, і цілісність дослідження залежить від спільного тлумачення респондентами цих основних термінів. Через це основні терміни визначаються на початку опитування та посилаються на них кожного разу, коли вони використовуються в опитуванні.

Існують організації, що займаються набором респондентів. Часто окремі дослідники, які використовують методи опитування, не мають доступу до достатньо великих груп потенційних кваліфікованих респондентів, щоб зібрати достатньо велику вибірку даних. У цьому випадку може знадобитися звернутися за допомогою до ключових осіб або організацій, щоб охопити більше потенційних респондентів. Якщо необхідно, щоб респонденти володіли спеціальними знаннями, необхідними для відповіді на запитання опитування, можна звернутись до відповідних професійних товариств.

При цьому дослідник повинен чітко сформулювати потенційні переваги при зверненні за допомогою до професійних товариств, ретельно вибрати назву та складати опис дослідження так, щоб було зрозуміло, чому допомога в такому дослідженні буде корисною.

Результати опитувань, крім їх корисності, повинні бути наочними та зрозумілими кінцевому користувачу, будь то респондент чи дослідник. Якщо результати по одному респонденту можуть бути представлені в текстовому вигляді, то узагальнені результати по групі респондентів повинні бути представлені також і в графічному вигляді. Це підвищує зручність аналізу тенденцій та думок груп респондентів.

У рамках роботи розроблена програмна система, що дозволяє організовувати опитування, обробляти їх, аналізувати показники та візуалізувати результати відповідно до обраного користувачем виду представлення.

Метою роботи є підвищенні зручності аналізу та візуалізації результатів опитувань шляхом розробки та використання програмної системи з сучасним інтерфейсом та розвинутими можливостями.

У першому розділі розглянуто проведення опитування як спосіб отримання інформації. Визначено процеси, що стосуються розробки проекту системи опитування. Проведено огляд та аналіз найбільш розповсюджених аналогів..

У другому розділі розглянуто типи тестів, з якими повинна працювати розроблювана система. Проведено систематизацію видів графічного представлення інформації.

Третій розділ містить функціональні та нефункціональні вимоги до розроблюваного програмного продукту.

У четвертому розділі виконано проектування архітектуру програмної системи відповідно до шаблону проектування MVC. Створено діаграму станів та переходів, діаграми діяльності та діаграму класів. Спроектовано структуру бази даних.

У п’ятому розділі проаналізовано інструменти розробки – мови та бібліотеки для реалізації серверної та клієнтської частин. Наведено приклади інтерфейсу програмної системи для роботи в режимах Manager та Interviewee.

У шостому розділі проведено функціональне тестування програмної системи на базі тестових сценаріїв. Проведено експерименти з визначенням зручності аналізу та візуалізації результатів опитувань.

1 КРИТИЧНИЙ АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Опис предметної області

Проведення опитування як способу отримання інформації відноситься до сфери емпіричних досліджень. Крім опитувань, серед емпіричних методів збору даних можна використовувати інтерв'ю та експерименти.

Опитування мають явні переваги для певних типів збору даних. Вони можуть отримати дані від значної кількості респондентів, які географічно розосереджені, за розумний час та ціну.

В останні роки зростає кількість досліджень, в тому числі, системної інженерії, які використовують опитування для збору даних. Приклади таких досліджень включають питання для розуміння впливу застосування найкращих практик на результативність проекту, стосуються побудови бізнес-кейсів, оцінки важливості різних економічних інструментів, визначення ролі домену продукту та домену проекту, вибору елементів інженерії, які важливі для модернізації застарілих систем тощо.

Якщо опитування стосується розробки проекту, то у контексті опитувальних досліджень обговорюються наступні процеси:

- аналіз вимог зацікавлених сторін,
- проектування архітектури,
- планування проекту,
- управління ризиками та можливостями,
- впровадження проекту,
- управління супутньою інформацією.

Зацікавленими сторонами в опитуванні є будь-яка особа, яка бере участь у певній діяльності або на неї впливає. Аналіз вимог зацікавлених сторін передбачає розуміння потреб зацікавлених сторін так, щоб система могла надавати необхідні послуги/інформацію кінцевим користувачам та іншим зацікавленим сторонам. Цей набір зацікавлених сторін та їхні потреби слід

враховувати з самого початку проекту дослідження. Приклади аналізу багатьох систем опитувань включають кінцевого користувача результатів дослідження, інших осіб, які в кінцевому підсумку використовуватимуть результати дослідження або на яких вплине застосування дослідження та базу потенційних кваліфікованих респондентів опитування.

Дослідник (менеджер) – це особа, якій потрібна відповідь на запитання дослідження. Також це може бути хтось, хто спонсорує дослідження. Проект опитування задовольняє дослідника, якщо він надає дані, достатні для вирішення питань дослідження.

Розглянемо інших осіб, які зацікавлені в результатах та/або на які вони впливають. Особи, які приймають рішення, можуть зрештою використовувати результати опитування для встановлення проектних, організаційних, економічних рішень.

Якщо результати опитування недійсні або неправильно застосовані, можуть виникнути негативні наслідки.Через це дослідники повинні дуже уважно підходити до того, як вони представляють результати аналізу опитування. Необхідно чітко вказати обмеження дослідження, а загрози валідності мають бути чітко пояснені.

Тепер визначимо вимоги до респондентів. Характеристики бази респондентів також впливають на проект опитування. Рівень мотивації, інтереси, навички та кваліфікація бази респондентів можуть бути основними факторами, які впливають на прийняття рішень у проекті опитування. Наприклад, рівень мотивації набору потенційних кваліфікованих респондентів може накладати обмеження на тривалість опитування. Люди, які цікавляться конкретною професійною темою, можуть бути більш терпимими до довгих анкет, ніж представники широкої громадськості, коли оцінюють споживчий продукт чи послугу.

Для будь-якого опитування респонденти повинні бути знайомі з певною термінологією та поняттями. Під час планування опитування та формулювання

запитань слід враховувати використання зрозумілої термінології та понять, з якими знайома певна спільнота респондентів.

Використання такого підходу дозволить ефективно отримати велику кількість інформації. Може знадобитися менше пояснювальної інформації, і багато інформації можна отримати за допомогою кількох запитань, спрямованих на поняття, знайомі базі респондентів.

База кваліфікованих респондентів також може накладати обмеження. Наприклад, організації, які наймають певних потенційних респондентів, можуть не дозволити оприлюднення певних елементів даних, які збираються опитуванням, через потенційні ризики для організації. Крім того, якщо більшість потенційних респондентів належать до певного сегменту, наприклад, великі корпорації, то можуть існувати обмеження на використання інформаційних технологій, які можуть вплинути на онлайн-опитування. Це може в свою чергу вплинути на рівень відповідей і валідність опитування.

1.2 Аналіз існуючих програмних аналогів

На даний час існує досить велика кількість програм, які дозволяють проводити опитування та візуалізувати результати. Розглянемо найбільш розповсюджені аналоги.

Програма SurveyGizmo є інструментом для проведення тестів (рис. 1.1). «Безкоштовна версія – це 25 запитань, за допомогою яких можна дізнатися будь-яку інформацію, від того, в якій сфері працюють респонденти до наявності у них вдома певних видів техніки. Купівля підписки відкриває двері у світ необмеженої кількості питань, опитувань і респондентів.»[1]

Нажаль, програма не передбачає узагальнення чи статистичної обробки результатів тестування, а також вибір типу графіку для візуалізації отриманих результатів.

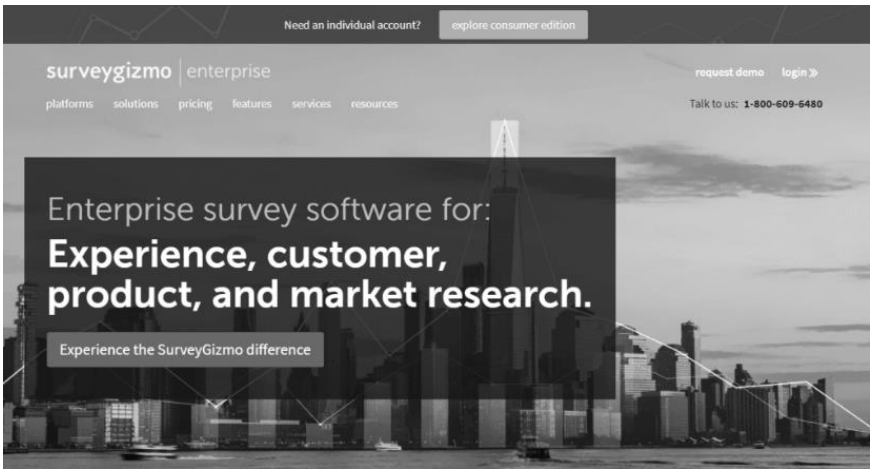

Рисунок 1.1 – Програма SurveyGizmo

На рис. 1.2 показаний приклад екранної форми для OnlineTestPad.

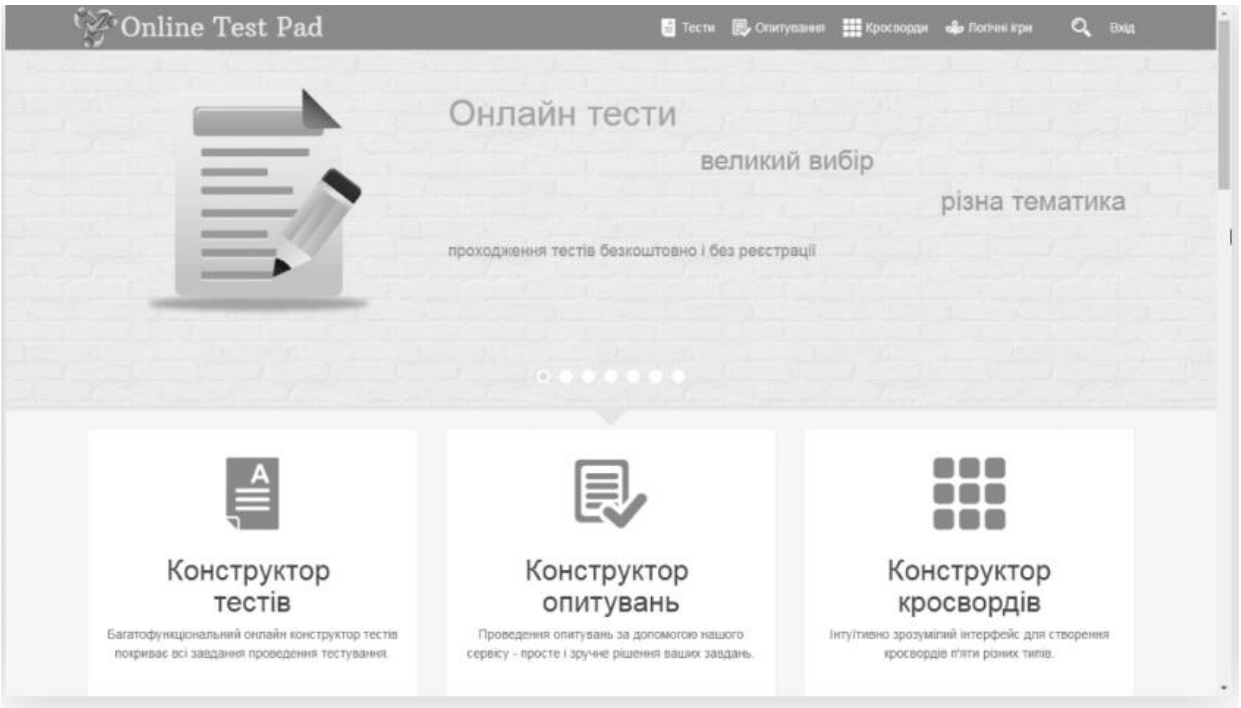

Рисунок 1.2 – Початкове вікно платформи OnlineTestPad

Багатофункціональна платформа OnlineTestPad«дозволяє створювати не тільки опитування, але і тести, кросворди. Сервіс безкоштовний та доступний українською мовою. Вибрати можна з 10 типів питань. Завжди є можливість додати універсальну відповідь “Інше”. Є можливість створювати графічні форми з фото і відео, обмежувати проходження за IP-адресою. Статистика збирається у

вигляді графіків і таблиць. Головний мінус, на думку самих розробників – відсутність можливостей для брендування.» [2] Але, крім того, недоліком можна вважати досить складний інтерфейс, тобто користувачеві потрібен час для того, щоб навчитись працювати з платформою.

Розглянемо наступний програмний продукт для тестування та представлення результатів – сервіс MySurveylab (рис. 1.3). «Сервіс надає пробний період 14 днів. У тестовий період є необмежена кількість анкет без обмежень щодо кількості питань. Обмеження щодо респондентів – 1000 осіб. Шлях збору відповідей – сайт, розсилка, посилання, фреймер. Присутній доступ до редагування питань і аналітики.» [3] Отже, обмеження кількості респондентів та короткий пробний період є перешкодою у широкому застосуванні продукту.

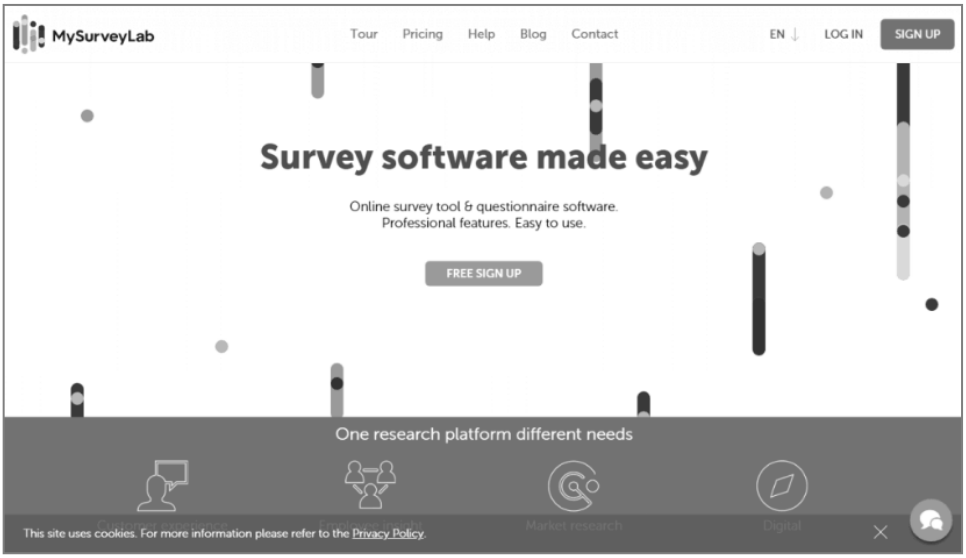

Рисунок 1.3 – Приклад інтерфейсу сервісу MySurveylab

Сервіс Surveynuts дозволяє складати тести і проводити тестування, та також потребує оплати. «Є безкоштовний період, в рамках якого можна використовувати готову логіку побудови опитування. Але сервіс, за відгуками багатьох користувачів, відштовхує своїм юзабіліті. Також ніякої допомоги при упорядкованому зборі статистики відповідей не надається.» [4] На рис. 1.4 наведено початкове вікно сервісу Surveynuts.

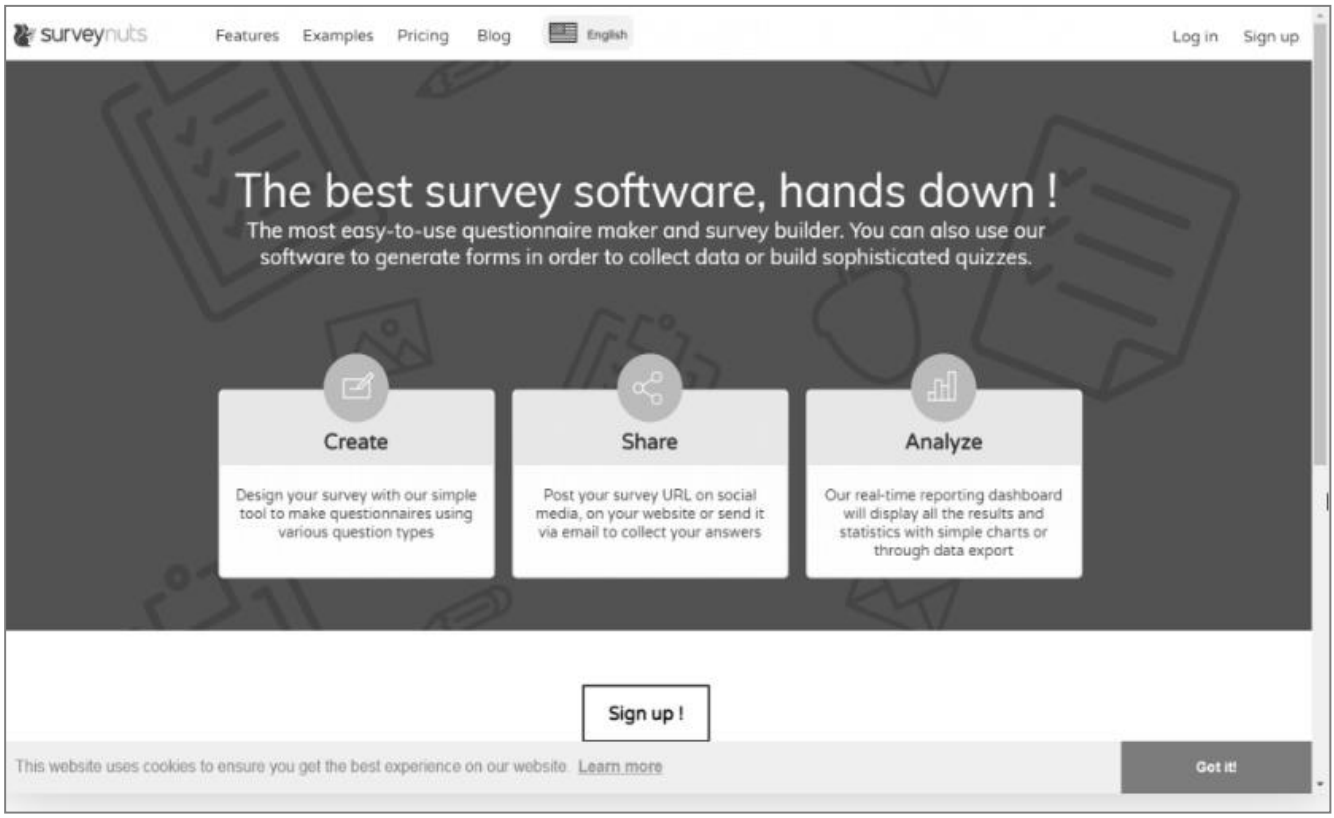

Рисунок 1.4 – Зовнішній вигляд вікна сервісу Surveynuts

Останній аналог, який розглянемо, це сервіс SurveyMonkey. Вигляд початкового вікна показаний на рис. 1.5.

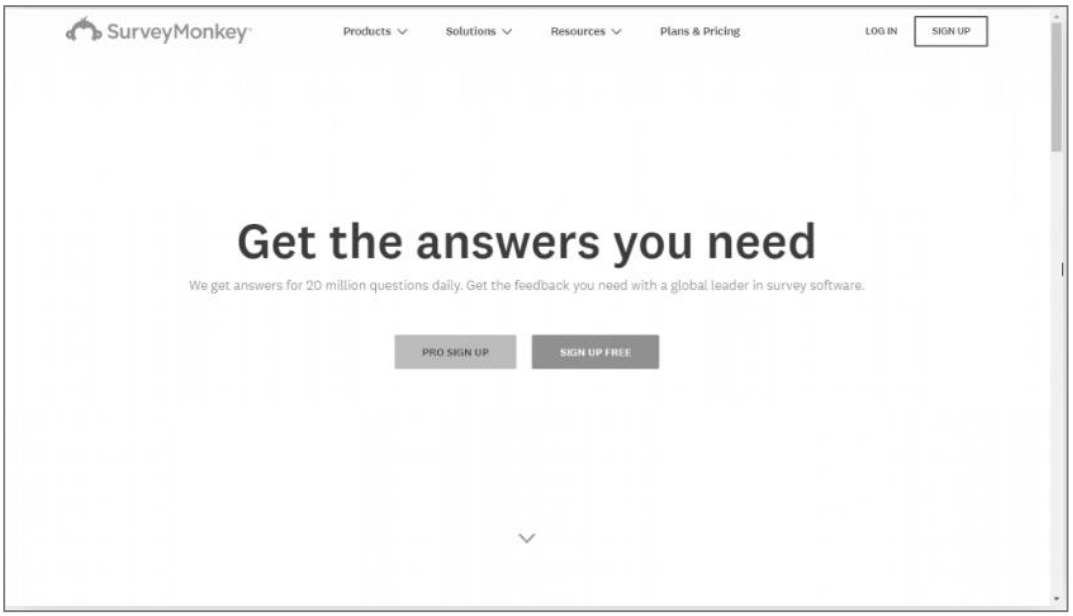

Рисунок 1.5 – Приклад інтерфейсу сервісу SurveyMonkey

«Сервіс, який дозволяє проводити понад 10 видів опитувань і A/B тестування. Є безкоштовна версія, російськомовний інтерфейс. Місце використання – сайти, соціальні мережі, поштова розсилка. Використовуючи базовий тариф, можна задати до 10 питань, отримавши при цьому до 100 відповідей.» [5]

Зазначимо, що A/B тестування є дуже простим та ефективним способом дослідження характеристик інтерфейсу: розташування та розміру елементів сайту, колірної схеми та ін. На рис. 1.6 наведен приклад використання A/B тестування характеристики інтерфейсу сайта.

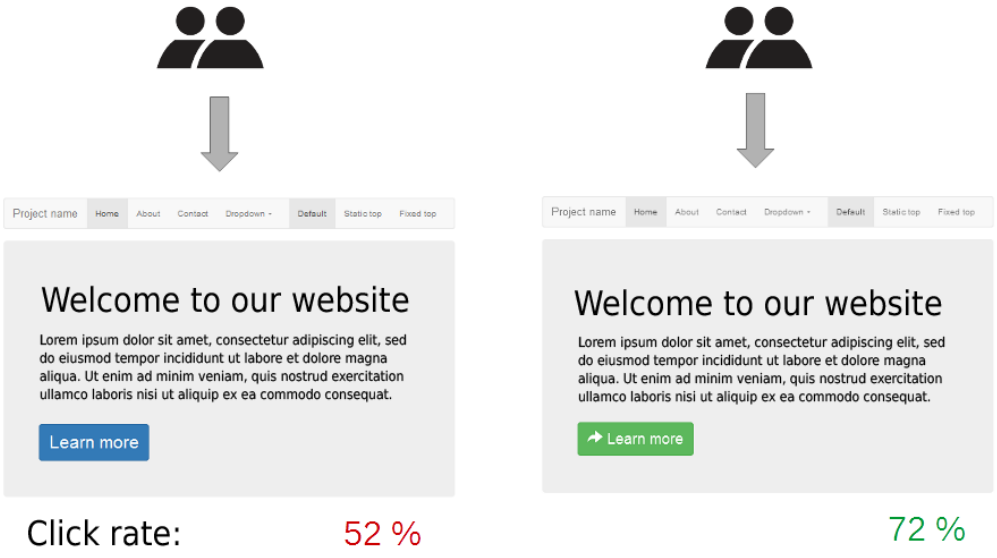

Рисунок 1.6 – Приклад A/B тестування колірної схеми сайту

Результати аналізу характеристик програмних аналогів було зведено у табл. 1.1. Також проведений загальний аналіз вимог до характеристик розроблюваної програмної системи, результати також приведені у табл. 1.1.

Як можна побачити, у будь-якому програмному аналогу існують суттєві недоліки, отже, можна зробити висновок про доцільність розробки власної програмної системи для аналізу та візуалізації результатів опитувань.

Таблиця 1.1 – Порівняльна характеристика програмних продуктів

Властивості продукту	Назва продукту					
	SurveyGizmo	OnlineTestPad	MySurveylab	Surveynuts	SurveyMonkey	Власна програмна система
Зрозумілий інтерфейс	+	–	+	+	–	+
Відсутність обмежень на кількість питань	–	+	–	–	–	+
Можливість статистичного аналізу результатів опитувань	+	–	+	–	+	+
Можливість візуалізації результатів аналізу опитувань	+	–	+	–	–	+
Вибір типу графіку	+	–	–	–	–	+
Різні типи питань	+	+	–	+	+	+

1.3 Висновки до розділу

У першому розділі розглянуто проведення опитування як спосіб отримання інформації. Визначено процеси, що стосуються розробки проекту системи опитування. Проведено огляд та аналіз найбільш розповсюджених аналогів. Зроблено висновок про доцільність розробки власної програмної системи для аналізу та візуалізації результатів опитувань.

2 РОЗРОБКА ПРОГРАМИ АНАЛІЗУ ТА ВІЗУАЛІЗАЦІЇ РЕЗУЛЬТАТІВ ОПИТУВАНЬ

2.1 Створення опитувань

Розглянемо типи тестів, з якими працюватиме система.

– Info block – набір текстових блоків, які користувач повинен розташувати у правильному порядку. Це відбувається шляхом вибору блоку, для якого потрібно змінити порядок, та натискання на стрілки «вверх» та «вниз»;

– Single choice – набір відповідей, серед яких можна обрати як правильну тільки одну;

– Multiple choice – набір відповідей, серед яких можна обрати декілька як правильні;

– Single text line – короткий фрагмент тексту;

– Multiple lines – розширений фрагмент тексту.

При створенні опитувань передбачається наступне:

– кожне опитування повинно мати унікальну назву;

– мінімальна кількість тестів в опитуванні – 1, максимальна – без обмежень;

– кожен тест має певний тип: Info block, Single choice, Multiple choice, Single text line чи Multiple lines;

– мінімальна кількість питань у тестах типів Single choice та Multiple choice – 2, максимальна – без обмежень;

– мінімальна кількість символів для тесту типу Single text line – 1, максимальна – 80;

– мінімальна кількість символів для тесту типу Multiple lines – 1, максимальна – 5000;

– мінімальна кількість символів у кожному блоці для тесту типу Info block – 1, максимальна – 100;

– мінімальна кількість блоків для тесту типу Info block – 1, максимальна – без обмежень.

2.2 Систематизація видів графічного представлення інформація

Візуалізація даних, що є результатами опитувань, відіграє важливу роль. Використовуються різні способи їх відображення – таблиці, графіки, звіти та багато інших [6].

Проблема відображення великої кількості даних вирішується з використанням діаграм (дашбордів, графіків), що являють собою користувальницькі інтерфейси візуалізації.

Аналітичні дані можуть бути представлені різними віджетами від таблиці та діаграми до стрілочних індикаторів. Деякі інструменти дозволяють програмісту самому реалізувати необхідну візуалізацію до анімацій, відео або довільної інфографіки. Самі аналітичні дані являють собою згруповані та агреговані вихідні дані. Є можливість застосувати фільтри та сортування на різних рівнях, відсікати дані за типовими значеннями, створювати обчислювані поля практично будь-якої складності. Приклад дашбордів з аналітикою показаний на рис. 2.1.

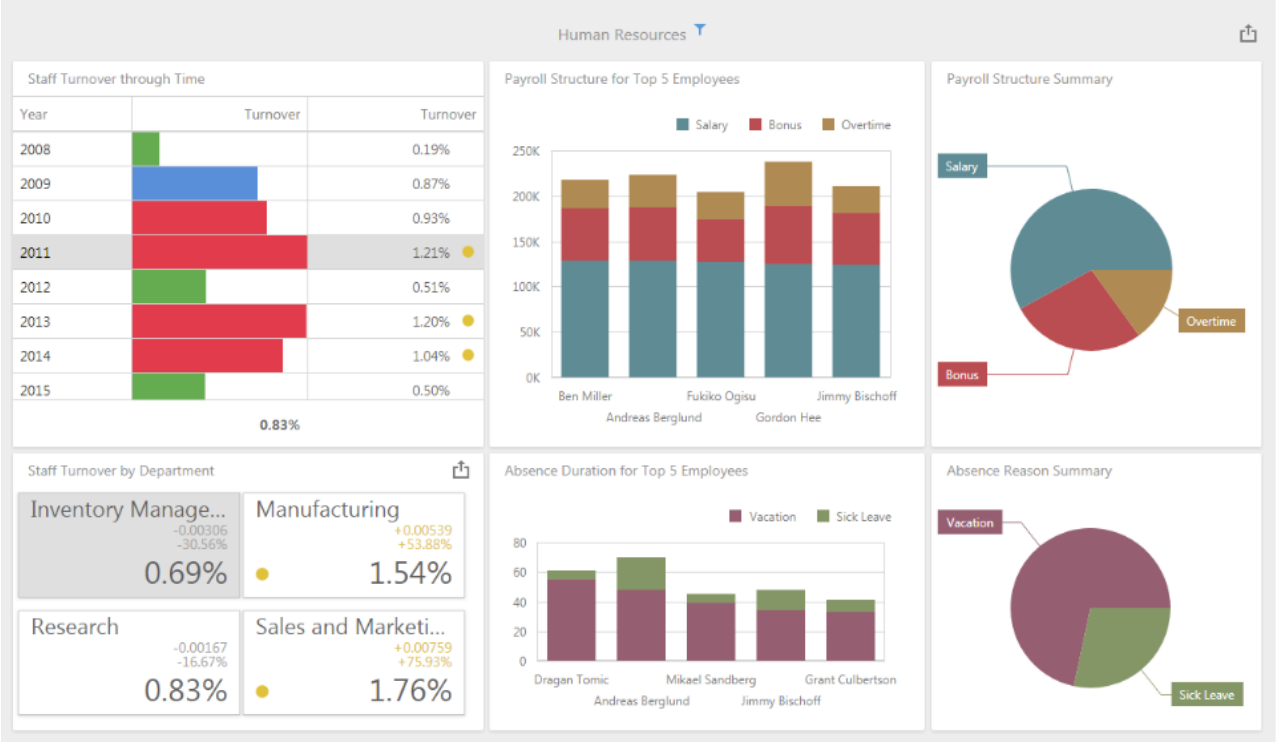

Рисунок 2.1 – Приклади аналітичних дашбордів

Серед найбільш відомих та інформативних дашбордів слід навести такі:

1. Картодіаграми.
2. Теплові карти.
3. Графіки.
4. Гістограми.
5. Кругові (секторні) діаграми.
6. Радіальні (сітчасті) діаграми.

2.3 Аналіз і класифікація засобів візуалізації текстової інформації

Опитування може включати в себе відкриту відповідь, тобто текстове поле, яке містить думки респондента стосовно певного питання чи процеса.

При обробці результатів опитувань застосовуються найпопулярніші методи виділення ознак, які використовуються для представлення текстових елементів, а саме:

1. Методи сумки слів (bag of words), які виділяють ознаки тексту шляхом підрахунку входжень термінів у тексті.

2. Розпізнавання сутностей, спрямоване на виділення власних імен сутностей, таких як імена осіб, організацій, місць або країн.

3. Методи реферування, що скорочують текст і подають лише найактуальнішу інформацію.

4. Розбір структури документа, який витягує структурну інформацію з тексту, таку як заголовки, імена авторів і дати публікації.

5. Аналіз настроїв і афектів, який використовується для визначення та кількісного визначення емоційних аспектів тексту.

Опитування класифікує підходи до візуалізації тексту на дві категорії:

1. Підходи до термінологічної тенденції базуються на частоті терміну в тексті. У таких методах вибір ознак використовується для зменшення кількості вимірів.

2. Підходи до семантичного простору сприяють семантичним методам вилучення особливостей тексту(ів). У більшості випадків вектори ознак, що представляють текст, є багатовимірними, тому для відображення цих функцій у 2D або 3D просторі використовуються більш вдосконалені алгоритми зменшення розмірності.

Техніки візуалізації та аналізу тексту пов’язані спільними завданнями аналізу. Загальноприйнятою э наступна класифікація:

- друкарська візуалізація;
- візуалізація діаграми;
- візуалізація графіка;
- візуалізація шкали часу;
- просторова візуалізація;
- багатовимірна візуалізація;
- топологічна візуалізація;
- радіальна візуалізація;
- 3d візуалізація.

Колір на графічному представленні використовується для відображення великої різноманітності ознак, наприклад, класифікації, подібності чи важливості.

Розмір шрифту також використовується для передачі характеристик тексту, наприклад, частоти або значення слів.

Зв’язки допомагають проілюструвати зв’язок між текстовими сутностями, наприклад, показати наступні слова, щоб відстежити варіації між різними редакціями тексту або передати структуру речення.

Теплові карти зазвичай використовуються для відображення текстових шаблонів, таких як подібності. На рис. 2.2 наведен приклад теплової карти, тобто таблиці, що має комірки, які містять значення відповідно до смислового навантаження рядку та стовпця, та колір, який показує ступінь «інтенсивності» значення, яке міститься у комірці. Використання кольору дозволяє загально усвідомити вміст матриці, не вдаючись до подробиць та не досліджуючи значення

у кожній комірці. На узагальненому рис. 2.2 відсутні підписи рядків та стовпців, але для конкретних теплових карт вони повинні бути присутніми.

Рисунок 2.2 – Приклад теплової карти

Хмари тегів кодують частоту появи слів у тексті за допомогою змінного розміру шрифту.

Карти відображають геопросторову інформацію, що міститься в тексті.

Шкала часу використовується для візуалізації тексту, який передає часову інформацію. Така техніка може використовувати метадані тексту та підтримувати часовий аналіз використання слова з часом.

Графи зазвичай використовують вузли та ребра для візуалізації певних структурних особливостей текстового корпусу.

Розглянемо тепер статистичні показники, які використовуються при аналізі текстової інформації, та застосуємо їх для аналізу відкритих відповідей (текстових полів) опитувань.

Статистичний показник TF-IDF (TF – term frequency, IDF – inverse document frequency) використовується для «оцінки важливості слів у контексті документа, що є частиною колекції документів чи корпусу. Вага (значимість) слова пропорційна кількості вживань цього слова у документі, і обернено пропорційна частоті вживання слова у інших документах колекції.

Показник TF-IDF використовується в задачах аналізу текстів та інформаційного пошуку. Його можна застосовувати як один з критеріїв релевантності документа до пошукового запиту, а також при розрахунку міри спорідненості документів при кластеризації.

Найпростішу функцію ранжування можна визначити як суму TF-IDF кожного терміну в запиті. Більшість просунутих функцій ранжування ґрунтуються на цій простій моделі.»

Показник TF (term frequency – частота слова) є відношенням числа входжень певного слова текста до загальної кількості слів у тексті. Таким чином, оцінюється важливість слова t_i в межах обраного текста.TF обчислюється за формулою (2.1):

$$TF = \frac{n_i}{\sum_k n_k}, \tag{2.1}$$

де n_i є кількістю входжень означеного слова в текст, а в знаменнику знаходиться загальна кількість слів у тексті.

IDF (inverse document frequency – обернена частота документа) – інверсія частоти, з якою слово зустрічається в текстах. Використання показника IDF дозволяє зменшує вагу слів, які часто зустрічаються у багатьох текстах.IDF обчислюється за формулою (2.2):

$$IDF = \log \frac{|D|}{|(d_i \supset t_i)|}, \tag{2.2}$$

де |D| –загальна кількість текстів;

$|d_i \supset t_i|$–кількість текстів, у яких зустрічається слово t_i.

У сучасній комп'ютерній лінгвістиці важливим статистичним інструментом є біграми, або n-грами. Біграма – це два слова (токена), які в тексті є сусідніми.

Частотний розподіл кожної біграми в рядку зазвичай використовується для простого статистичного аналізу тексту в багатьох застосуваннях, включаючи обчислювальну лінгвістику, криптографію, розпізнавання мовлення і т.д.

Біграми допомагають визначити умовні ймовірність токена з урахуванням попереднього токена, коли застосовується співвідношення умовної ймовірності. Відображення кольорів комірок вказує на ваги відповідної біграми.

2.4 Висновки до розділу

У даному розділі розглянуто типи тестів, з якими повинна працювати розроблювана система.

Проведено систематизацію видів графічного представлення інформації, наведено приклади аналітичних дашбордів, визначено типи дашбордів.

Проведено аналіз технік візуалізації та статистичних показників, які використовуються для аналізу текстових складових опитувань.

3 СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

3.1 Функціональні вимоги до програми

З програмною системою для аналізу та візуалізації (скорочено – САВ) результатів опитування може працювати User, від якого успадковуються ще 2 категорії користувачів – Manager (менеджер) та Interviewee (той, хто проходить опитування). Manager відповідає за створення опитувань та перегляд узагальнених результатів. Interviewee обирає опитування та проходить його, а потім може переглядати власний результат.

Функціональні вимоги включають наступні варіанти використання:

1. Створення опитування
2. Імпорт опитування
3. Створення тесту
4. Конструювання тесту
5. Імпорт тесту
6. Редагування опитування
7. Редагування тесту
8. Видалення тесту
9. Видалення опитування
10. Перегляд узагальнених результатів
11. Вибір типу візуалізації
12. Авторизація у системі
13. Реєстрація у системі
14. Проходження опитування
15. Вибір опитування
16. Перегляд результатів

На рис. 3.1 представлена діаграма варіантів використання системи для аналізу та візуалізації результатів опитування.

Нижче наведені покрокові сценарії роботи варіантів використання.

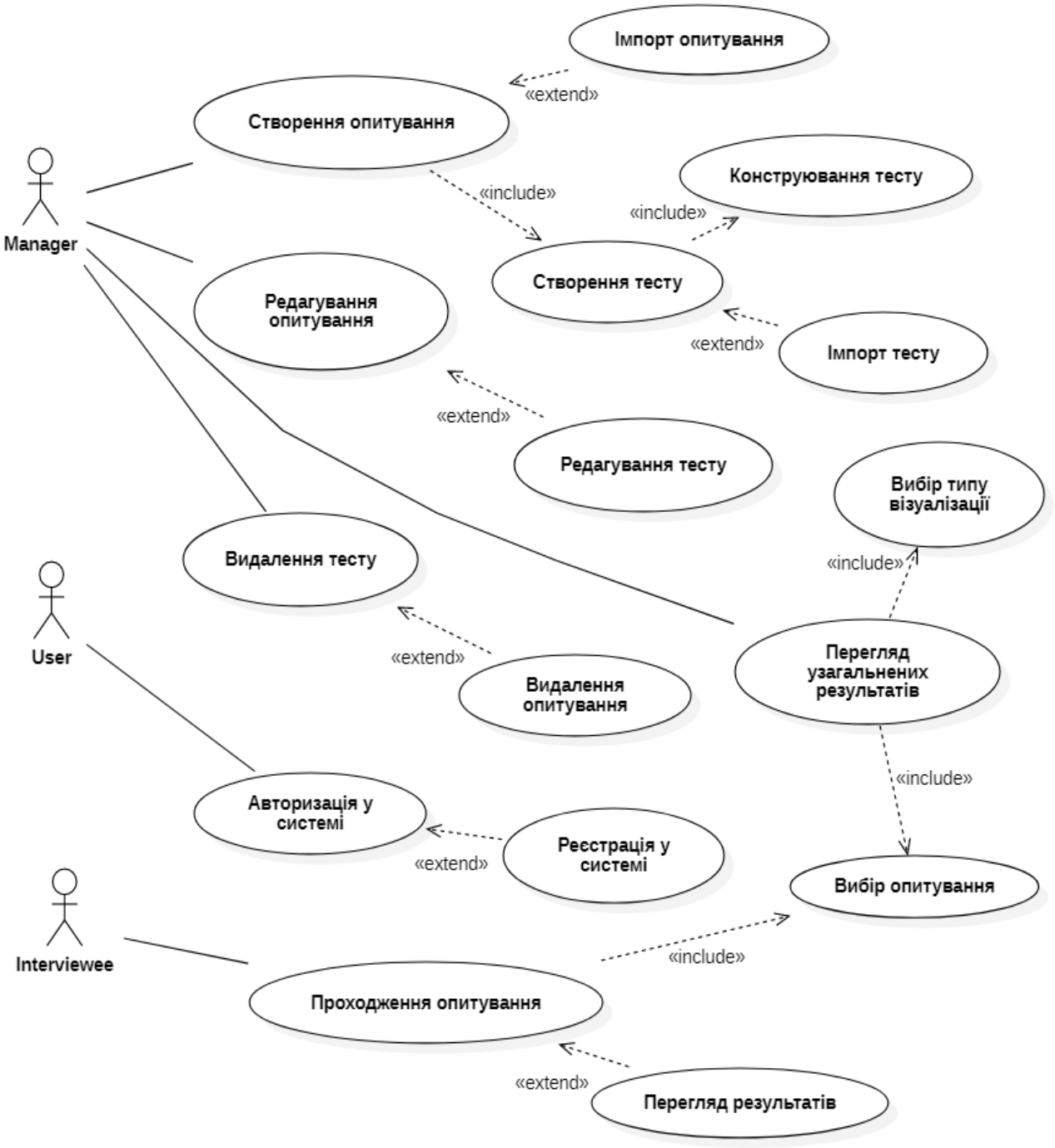

Рисунок 3.1 – Діаграма варіантів використання

Варіант використання «Створення опитування»

Основний виконавець: Користувач, зареєстрований як Manager.

Передумови: Manager бажає створити нове опитування.

Післяумови: Нове опитування створено та збережено у системі.

Основний успішний сценарій.

1. Manager обирає команду для створення нового опитування.
2. CAB надає поле для ввода назви опитування.
3. Manager вводить назву та натискає ОК.
4. CAB пропонує створення опитування власноруч чи імпорту всіх тестів з іншого опитування.
5. Якщо Manager обирає імпортувати опитування цілком, то запускається варіант використання «Імпорт опитування». Кінець сценарію.
6. Якщо Manager обирає створювати опитування власноруч, то CAB пропонує створити нову сторінку (Page). Перша сторінка створюється автоматично без участі Manager.
7. Якщо Manager хоче створити новий тест, він натискає знак «+». CAB запускає варіант використання «Створення тесту».
8. Для завершення роботи з новим опитуванням Manager обирає відповідну команду.
9. CAB зберігає опитування у БД.

Альтернативні сценарії.

3а) Manager не ввів назву для опитування.

1) CAB видає повідомлення про порожнє поле назви.

3б) Manager ввів назву, яка вже є у БД.

1) CAB видає повідомлення про помилку назви.

5а) Немає зв’язку з імпортуємим опитуванням.

1) CAB видає повідомлення про помилку. Невдале завершення прецеденту.

8а) Manager не завершує роботу з опитуванням.

1) CAB чекає на завершення.

9а) Немає зв’язку з БД, неможливо зберегти опитування.

1) CAB видає повідомлення про помилку БД. Невдале завершення прецеденту.

Варіант використання «Імпорт опитування»

Основний виконавець: Користувач, зареєстрований як Manager.

Передумови: Manager бажає створити нове опитування на базі іншого.

Післяумови: Нове опитування створено та збережено у системі.

Основний успішний сценарій.

1. Manager обирає пристрій чи хмару, де розташоване опитування, яке імпортується.
2. САВ надає форму для вибору певного опитування.
3. Manager обирає опитування та натискає ОК.
4. САВ завантажує вибране опитування та зберігає його під новою назвою (яку ввів Manager).
5. Завершення сценарію.

Альтернативні сценарії.

1а) Manager не обрав пристрій.

1) САВ чекає на вибір.

3а) Manager не обрав опитування.

1) САВ чекає на вибір.

4а) Обране опитування має невідповідний формат.

1) САВ видає повідомлення про помилку. Невдале завершення прецеденту.

Варіант використання «Створення тесту»

Основний виконавець: Користувач, зареєстрований як Manager.

Передумови: Manager бажає створити новий тест для поточного опитування.

Післяумови: Новий тест створений та збережений у БД.

Основний успішний сценарій.

1. САВ пропонує Manager створювати тест власноруч чи імпортувати його з іншого тесту.
2. Якщо Manager бажає створити поточний тест на базі іншого, САВ запускає варіант використання «Імпорт тесту».

3. САВ створює новий об’єкт для тесту та надає користувачу Manager форму для вибору типу тесту:
   - Info block;
   - Single choice;
   - Multiple choice;
   - Single text line;
   - Multiple lines.
4. Manager обирає тип тесту.
5. САВ викликає варіант використання «Конструювання тесту» для візуалізації форми тесту на екрані.
6. Manager вводить дані в поля форми відповідно до обраного типу тесту та натискає ОК.
7. САВ зберігає тест у БД.
8. Завершення сценарію.

Альтернативні сценарії.

4а) Manager не обрав тип тесту.

1) САВ чекає на вибір.

6а) Manager не заповнив дані в полях форми.

1) САВ чекає на вибір.

7а) Помилка звернення до БД.

1) САВ видає повідомлення про помилку. Невдале завершення прецеденту.

Варіант використання «Конструювання тесту»

Основний виконавець: Користувач, зареєстрований як Manager, САВ.

Передумови: Manager обрав тип тесту.

Післяумови: Графічна форма тесту відтворена на екрані.

Основний успішний сценарій.

1. САВ завантажує з БД опис графічних елементів для форми тесту.

2. САВ відтворює графічні елементи на екрані.
3. Manager переглядає графічну форму тесту.
4. Завершення сценарію.

Альтернативні сценарії.

1а) БД з описами графічних елементів недоступна.

1) САВ видає повідомлення про помилку. Невдале завершення прецеденту.

Варіант використання «Імпорт тесту»

Основний виконавець: Користувач, зареєстрований як Manager.

Передумови: Manager бажає створити новий тест на базі іншого.

Післяумови: Новий тест створено та збережено у системі.

Основний успішний сценарій.

1. Manager обирає пристрій чи хмару, де розташований тест, який імпортується.
2. САВ надає форму для вибору певного опитування.
3. Manager обирає опитування та натискає ОК.
4. САВ відображає тести з опитування.
5. Manager обирає потрібний тест та натискає ОК.
6. САВ завантажує вибраний тест та зберігає його в поточному опитуванні (в тому, який створює Manager).
7. Завершення сценарію.

Альтернативні сценарії.

1а) Manager не обрав пристрій.

1) САВ чекає на вибір.

3а) Manager не обрав опитування.

1) САВ чекає на вибір.

4а) Обране опитування має невідповідний формат.

1) САВ видає повідомлення про помилку. Невдале завершення прецеденту.

5а) Manager не обрав тест.

1) CAB чекає на вибір.

Варіант використання «Редагування опитування»

Основний виконавець: Користувач, зареєстрований як Manager.

Передумови: Manager бажає відредагувати опитування.

Післяумови: Опитування відредагувано та збережено у системі.

Основний успішний сценарій.

1. Manager обирає команду для редагування опитування.
2. CAB надає перелік опитувань, які є у базі.
3. Manager обирає опитування та натискає ОК.
4. CAB виводить на екран всі тести з опитування.
5. Якщо Manager бажає змінити назву опитування, він двічі клацає мишею, вводить нову назву та натискає ОК.
6. Якщо Manager бажає змінити порядок тестів в опитуванні, він обирає тест, який потрібно перемістити, та натискає стрілки «вверх» та «вниз».
7. Для завершення редагування опитування Manager обирає відповідну команду.
8. CAB зберігає оновлене опитування у БД.

Альтернативні сценарії.

2а) Немає доступу до бази.

1) CAB видає повідомлення про помилку доступу.

3а) Manager не обрав опитування.

1) CAB чекає на вибір.

5а) Manager не ввів нову назву опитування.

1) CAB залишає попередню назву.

5б) Manager ввів назву, яка співпадає з назвою іншого опитування.

1) CAB видає повідомлення про помилку назви.

7а) Manager не завершує роботу з опитуванням.

1) CAB чекає на завершення.

8а) Немає зв’язку з БД, неможливо зберегти опитування.

1) САВ видає повідомлення про помилку БД. Невдале завершення прецеденту.

Варіант використання «Редагування тесту»

Основний виконавець: Користувач, зареєстрований як Manager.

Передумови: Manager бажає відредагувати тест.

Післяумови: Тест відредагувано та збережено у системі.

Основний успішний сценарій.

1. Якщо Manager хоче редагувати питання в тесті, він двічі клацає мишею на текст питання, вводить новий текст та натискає ОК.
2. Якщо Manager бажає змінити порядок питань в тесті, він обирає питання, яке потрібно перемістити, та натискає стрілки «вверх» та «вниз».
3. Для завершення редагування тесту Manager обирає відповідну команду.
4. САВ зберігає оновлений тест у БД.

Альтернативні сценарії.

3а) Manager не завершує роботу з тестом.

1) САВ чекає на завершення.

4а) Немає зв’язку з БД, неможливо зберегти опитування.

1) САВ видає повідомлення про помилку БД. Невдале завершення прецеденту.

Варіант використання «Видалення тесту»

Основний виконавець: Користувач, зареєстрований як Manager.

Передумови: Manager бажає видалити тест з поточного опитування.

Післяумови: Опитування оновлено та збережено у системі.

Основний успішний сценарій.

1. САВ надає перелік тестів, які є у базі.
2. Manager обирає тест, який потрібно видалити.

3. Manager обирає відповідну команду, САВ питає чи впевнений Manager, що хоче видалити тест, та після підтвердження видаляє його.
4. САВ зберігає оновлене опитування у БД.

Альтернативні сценарії.

1а, 4а) Немає доступу до бази.

1) САВ видає повідомлення про помилку доступу.

2а) Manager не обрав тест.

1) САВ чекає на вибір.

3а) Manager не підтвердив видалення тесту.

1) Опитування залишається без змін.

Варіант використання «Видалення опитування»

Основний виконавець: Користувач, зареєстрований як Manager.

Передумови: Manager бажає видалити опитування з системи.

Післяумови: Опитування видалено.

Основний успішний сценарій.

1. САВ надає перелік опитувань, які є у базі.
2. Manager обирає опитування, яке потрібно видалити цілком чи частково.
3. Якщо з опитування потрібно видалити окремий тест, Manager обирає відповідну команду та САВ запускає варіант використання «Видалення тесту». Кінець сценарія.
4. Якщо Manager обрав опитування, яке потрібно видалити цілком, САВ питає чи впевнений Manager, що хоче видалити опитування, та після підтвердження видаляє його.
5. САВ зберігає набір опитувань у БД.

Альтернативні сценарії.

1а, 5а) Немає доступу до бази.

1) САВ видає повідомлення про помилку доступу.

4а) Manager не підтвердив видалення опитування.

1) Набір опитувань залишається без змін.

Варіант використання «Перегляд узагальнених результатів»

Основний виконавець: Користувач, зареєстрований як Manager.

Передумови: Manager бажає переглянути узагальнені результати опитування, які пройшли декілька респондентів.

Післяумови: Узагальнені результати переглянуті.

Основний успішний сценарій.

1. Manager робить запит на перелік опитувань.
2. CAB надає перелік опитувань, які є у базі.
3. Manager обирає потрібне опитування, для цього CAB запускає варіант використання «Вибір опитування».
4. CAB звертається до БД та завантажує дані щодо обраного Manager'ом опитування.
5. Manager робить запит на відображення переліку типів візуалізації.
6. CAB аналізує поточне опитування та типи полів, яке воно містить.
7. CAB надає Manager'у припустимі варіанти типів візуалізації для поточного (обраного) опитування.
8. Manager обирає тип візуалізації, для цього CAB запускає варіант використання «Вибір типу візуалізації».
9. CAB будує візуалізацію результатів опитування.
10. Manager переглядає результати.

Альтернативні сценарії.

2а) Немає доступу до бази.

1) CAB видає повідомлення про помилку доступу до БД.

3а) Manager не обрав опитування.

1) CAB очікує на вибір.

8а) Manager не обрав тип візуалізації.

1) CAB очікує на вибір.

Варіант використання «Вибір типу візуалізації»

Основний виконавець: Користувач, зареєстрований як Manager.

Передумови: Manager бажає переглянути узагальнені результати опитування відповідно до обраного типу візуалізації.

Післяумови: Manager обрав тип візуалізації для перегляду узагальнених результатів опитування.

Основний успішний сценарій.

1. CAB надає Manager'у варіанти типів візуалізації, які вона підтримує.
2. Manager обирає потрібний тип візуалізації.
3. CAB завантажує відповідні графічні елементи для подальшої візуалізації.

Альтернативні сценарії.

2а) Manager не обрав тип візуалізації.

1) CAB очікує на вибір.

Варіант використання – «Авторизація у системі»

Основний виконавець: User.

Передумови: Користувач попередньо зареєстрований (має логін/пароль) у системі.

Післяумови: Користувач авторизований як Manager чи Interviewee.

Основний успішний сценарій.

1. Користувач User заходить в розділ авторизації CAB.
2. Для авторизації Manager повинен ввести логін, пароль та номер телефону, Interviewee – тільки логін та пароль.
3. CAB надає поля для заповнення відповідно до обраного користувачем типу профіля.
4. Користувач заповнює дані у полях.
5. CAB перевіряє введені дані та знаходить профіль користувача.
6. CAB надає користувачу права доступу.
7. Завершення авторизації.

Альтернативні сценарії.

3а) Не заповнені або частково заповнені поля.

1) САВ видає повідомлення про порожні поля.

3б) Не знайдений профіль користувача.

1) САВ видає повідомлення, що авторизація не здійснена.
2) САВ пропонує виконати реєстрацію.

Варіант використання – «Реєстрація у системі»

Основний виконавець: User.

Передумови: Користувач попередньо незареєстрований у системі.

Післяумови: Користувач зареєстрований як Manager чи Interviewee.

Основний успішний сценарій.

1. Користувач User заходить в розділ реєстрації САВ.
2. Для реєстрації Manager повинен ввести логін, пароль та номер телефону, Interviewee – тільки логін та пароль.
3. САВ надає поля для заповнення відповідно до обраного користувачем типу профіля.
4. Користувач заповнює дані у полях.
5. САВ перевіряє введені дані та створює профіль користувача.
6. Завершення реєстрації.

Альтернативні сценарії.

4а) Не заповнені або частково заповнені поля.

1) САВ видає повідомлення про порожні поля.

4б) Введені дані дублюються з профілем іншого користувача.

1) САВ видає повідомлення про помилку реєстрації.
2) САВ пропонує повторно виконати реєстрацію.

Варіант використання– «Проходження опитування»

Основний виконавець: Користувач, зареєстрований як Interviewee.

Передумови: Interviewee має намір пройти опитування.

Післяумови: Опитування пройдено.

Основний успішний сценарій.

1. Interviewee робить запит на проходження опитування.
2. САВ запускає варіант використання «Вибір опитування».
3. Interviewee відкриває форму для проходження опитування.
4. САВ визначає кількість кроків N для обраного опитування.
5. Поточний крок P= 0.
6. P = P + 1. САВ відображає питання тесту на екрані.
7. Interviewee відповідає на тест відповідно до його типу.
8. САВ зберігає відповіді Interviewee на поточний тест.
9. Якщо залишились не пройдені запитання (P<N), перехід до п. 6.
10. Для перегляду власних результатів Interviewee запускає відповідну команду.
11. Система обчислює значення тестових критеріїв по кожному напрямку.
12. САВ запускає варіант використання «Перегляд результатів».
13. Завершення прецеденту.

Альтернативні сценарії.

1a) Interviewee відмовляється проходити опитування.

1) Завершення прецеденту.

7a) Interviewee відмовляється відповідати на тест.

1) САВ пропонує завершення опитування.

8a) Немає доступу до БД.

1) САВ видає помилку.
2) САВ виконує повторне з’єднання з БД.

Варіант використання – «Вибір опитування»

Основний виконавець: Користувач, зареєстрований як Interviewee.

Передумови: Interviewee має намір обрати опитування.

Післяумови: Опитування обрано.

Основний успішний сценарій.

1. САВ відкриває список опитувань.

2. Interviewee переглядає список.
3. При бажанні звернутись до останніх опитувань (кількість налаштовується, за умовчанням – 5) – Interviewee обирає меню «Last surveys».
4. При бажанні звернутись відсортувати опитування за назвою (по алфавіту) чи датою створення опитування Interviewee обирає меню «Sort surveys by» та вказує вид сортування.
5. Interviewee обирає опитування, двічі клацаючи мишею.
6. САВ завантажує матеріали обраного опитування.
7. Завершення прецеденту.

Альтернативні сценарії.

5а) Interviewee відмовляється обирати опитування.

1) САВ очікує на вибір.

Варіант використання – «Перегляд результатів»

Основний виконавець: Користувач, зареєстрований як Interviewee.

Передумови: Interviewee має намір переглянути результати опитування.

Післяумови: Результати опитування переглянуто.

Основний успішний сценарій.

1. Interviewee робить запит на перегляд результатів опитування.
2. САВ надає перелік тестів з опитування та результати проходження кожного тесту.
3. Interviewee переглядає список тестів. При бажанні зберегти результати на власному носії, обирає відповідну команду, вказує шлях та зберігає результати.
4. Завершення прецеденту.

Альтернативні сценарії.

2а) Немає зв’язку з БД, помилка завантаження результатів опитування.

1) САВ видає повідомлення.
2) Завершення прецеденту.

3.2 Нефункціональні вимоги до розроблюваної програмної системи

Розроблювана програмна система, крім функціональних, повинна відповідати вимогам до її нефункціональних характеристик [7, 8]. Розглянемо основні атрибути та відповідні сценарії з урахуванням джерела, стимулу, артефакту, середи, реакції та її міри.

Продуктивність

Сценарій – додавання нового опитування у власну БД.

Джерело – програмна система.

Стимул – додавання потрібних опитувань.

Артефакт – власна БД.

Середа – нормальна.

Реакція – оновлення БД.

Міра реакції – оновлення БД відбувається не довше ніж за 1 секунду.

Той же сценарій:

Середа – режим підвищеного навантаження.

Міра реакції – оновлення БД відбувається не довше ніж за 4 секунди.

Сценарій – пошук опитування за назвою.

Джерело – програмна система.

Стимул – запит на пошук.

Артефакт – БД.

Середа – нормальна.

Реакція – отримання опитування та його тестів.

Міра реакції – отримання даних відбувається не довше ніж за 3 секунди.

Той же сценарій:

Середа – режим підвищеного навантаження.

Міра реакції – отримання даних відбувається не довше ніж за 6 секунд.

Сценарій – відображення списку опитувань у БД.

Джерело – програмна система.

Стимул – запит на відображення опитувань.

Артефакт – список опитувань.

Середа – нормальна.

Реакція – відображення списку опитувань.

Міра реакції – відображення списку опитувань відбувається не більше ніж за 1 секунду.

Той же сценарій:

Середа – погіршений режим.

Міра реакції – відображення списку опитувань відбувається не більше ніж за 2 секунди.

Сценарій – відкриття опитування системі з відображенням тестів, які в нього входять.

Джерело – програмна система.

Стимул – запит на відображення опитування.

Артефакт – опитування та його тести.

Середа – нормальна (об’єм опитування не перевищує 10 Мб).

Реакція – відображення опитування та його тестів в упорядкованому вигляді.

Міра реакції – відображення опитування з можливістю скролінгу тестів відбувається не більше ніж за 2,5 секунди.

Той же сценарій:

Середа – режим підвищеного навантаження (об’єм опитування перевищує 10 Мб та/або кількість тестів перевищує 100).

Міра реакції – відображення опитування з можливістю скролінгу відбувається не більше ніж за 4 секунди.

Сценарій – авторизація користувача системи.

Джерело – зареєстрований користувач системи.

Стимул – запит користувача на авторизацію.

Артефакт – процес обробки запиту на авторизацію.

Середа – нормальна.

Реакція – виконання авторизації та отримання доступу до власної БД.

Міра реакції – не більше ніж 1 секунда.

Той же сценарій:

Середа – режим підвищеного навантаження (прайм-тайм).

Міра реакції – не більше ніж 3 секунди.

Сценарій – реєстрація нового користувача системи.

Джерело – незареєстрований користувач системи.

Стимул – запит нового користувача на реєстрацію.

Артефакт – процес обробки запиту на реєстрацію.

Середа – нормальна.

Реакція – виконання реєстрації та створення власного профілю.

Міра реакції – не більше ніж 2 секунда.

Той же сценарій:

Середа – режим підвищеного навантаження (прайм-тайм).

Міра реакції – не більше ніж 4 секунди.

Зручність користування

Сценарій: зареєстрований користувач бажає створити запит на пошук опитування.

Джерело – зареєстрований користувач.

Стимул – запит на пошук.

Артефакт – програмна система.

Середа – нормальна, відкритий доступ.

Реакція – опитування знайдено.

Міра реакції – пошук опитування у БД відбувається не більше ніж за 2 секунди.

Той же сценарій:

Середа – погіршений режим.

Міра реакції – пошук опитування у БД відбувається не більше ніж за 4 секунди.

Сценарій: зареєстрований користувач Manager бажає додати тест в опитування.

Джерело – зареєстрований користувач Manager.

Стимул – запит на додавання тесту в опитування.

Артефакт – програмна система.

Середа – нормальна.

Реакція – оновлене опитування.

Міра реакції – оновлення опитування відбувається не більше ніж за 1,5 секунди.

Той же сценарій:

Середа – погіршений режим (підвищено навантаження на сервер).

Міра реакції – оновлення опитування відбувається не більше ніж за 3 секунди..

Сценарій: зареєстрований користувач Manager бажає переглянути узагальнені результати опитування.

Джерело – зареєстрований користувач Manager.

Стимул – запит на перегляд результатів.

Артефакт – програмна система.

Середа – нормальна.

Реакція – графічна форма з результатами опитування.

Міра реакції – створення графічної форми та її відображення відбувається не більше ніж за 2 секунди.

Той же сценарій:

Середа – погіршений режим (підвищено навантаження на сервер).

Міра реакції – створення графічної форми та її відображення відбувається не більше ніж за 3 секунди.

Сценарій: зареєстрований користувач Interviewee (респондент) бажає переглянути результати власного опитування.

Джерело – зареєстрований користувач Interviewee.

Стимул – запит на перегляд результатів власного опитування.

Артефакт – програмна система.

Середа – нормальна.

Реакція – результати власного опитування.

Міра реакції – час на доступ до перегляду результатів власного опитування займає не більше ніж 2 секунди.

Той же сценарій:

Середа – режим підвищеного навантаження (прайм-тайм).

Міра реакції – час на доступ до перегляду результатів власного опитування займає не більше ніж 4 секунди.

Готовність

Сценарій – відображення результатів пошуку опитування.

Джерело – зареєстрований користувач.

Стимул – запит на пошук опитування.

Артефакт – процес.

Середа – нормальна.

Реакція – потрібне опитування знайдено.

Міра реакції – потрібне опитування знайдено не менш ніж у 95% випадків.

Той же сценарій:

Середа – режим підвищеного навантаження (прайм-тайм).

Міра реакції – потрібне опитування знайдено не менш ніж у 90% випадків.

Сценарій – редагування обраного опитування.

Джерело – програмна система.

Стимул – запит на редагування.

Артефакт – постійна пам’ять (власний жорсткий диск чи флеш-пам’ять, хмарні репозиторії – GoogleDisk, One-drive, Dropbox тощо).

Середа – нормальна.

Реакція – потрібне опитування відредаговане.

Міра реакції – потрібне опитування відредаговане не менш ніж у 93% випадків.

Той же сценарій:

Середа – погіршений режим.

Реакція – відкат процедури редагування обраного опитування, припинення оновлення.

Міра реакції – програмна система підтримує робочий стан не менш ніж у 94% випадків.

Сценарій – отримання опитування з власної БД.

Джерело – програмна система.

Стимул – запит на отримання опитування з БД.

Артефакт – постійна пам’ять (власний жорсткий диск чи флеш-пам’ять, хмарні репозиторії – GoogleDisk, One-drive, Dropbox тощо).

Середа – нормальна.

Реакція – надання результатів пошуку.

Міра реакції – програмна система готова до нового пошуку не більше ніж через 1 секунду після видачі результату.

Той же сценарій:

Середа – погіршена.

Реакція – сповіщення користувача про невдачу пошуку.

Міра реакції – програмна система готова до нового пошуку не більше ніж через 2 секунди після видачі сповіщення про попередню невдачу пошуку.

Супровід

Сценарій – удосконалення інтерфейсу програмної системи.

Джерело – розробник-програміст.

Стимул – налагодження зручності, виправлення помилок інтерфейсу.

Артефакт – програмний інтерфейс.

Середа – нормальна.

Реакція – розробник-програміст оновив інтерфейс та виправив заявлені помилки в інтерфейсі.

Міра реакції – користувачі отримують оновлений інтерфейс не більш ніж через 2 тижні.

Той же сценарій:

Середа – погіршена.

Реакція – розробник-програміст частково оновив інтерфейс та виправив деякі заявлені помилки в інтерфейсі.

Міра реакції – користувачі отримують частково оновлений інтерфейс не більш ніж через 3 тижні.

Сценарій – адаптація до портування системи на іншу платформу.

Джерело – розробник-програміст.

Стимул – оновлення функціоналу для портування системи на іншу платформу.

Артефакт – платформа.

Середа – нормальна.

Реакція – розробник-програміст оновив функціонал системи відповідно до вимог нової платформи.

Міра реакції – система буде адаптована до нової платформи не більше ніж за 2 місяця.

Той же сценарій:

Середа – погіршена.

Реакція – розробник-програміст частково оновив функціонал системи відповідно до вимог нової платформи.

Міра реакції – система буде адаптована до нової платформи з деякими функціональними обмеженнями не більше ніж за 3 місяця.

Безпека

Сценарій: авторизація користувача системи (зареєстрованого).

Джерело – користувач системи (неавторизований).

Стимул – спроба авторизуватися з неправильним паролем.

Артефакт – програмна система.

Середа – нормальна.

Реакція – авторизація не виконується, система пропонує повторну авторизацію.

Міра реакції – у 100% спроб.

Той же сценарій:

Джерело – користувач системи (авторизований).

Стимул – спроба повторної авторизації.

Реакція – повторна авторизація не виконується, система повідомляє про непотрібність цієї дії.

Міра реакції – у 100% спроб.

Той же сценарій:

Джерело – користувач системи (неавторизований).

Стимул – спроба авторизуватися з правильним логіном та паролем.

Артефакт – програмна система.

Середа – погіршена.

Реакція – авторизація виконується.

Міра реакції – у 99% спроб.

Сценарій: користувач респондент намагається звернутись до узагальнених результатів опитувань.

Джерело – авторизований користувач респондент.

Стимул – спроба отримати узагальнені результати опитувань.

Артефакт – програмна система.

Середа – нормальна.

Реакція – відмова у отриманні узагальнених результатів опитувань.

Міра реакції – у 100% спроб.

Той же сценарій:

Джерело – неавторизований користувач.

Реакція – відмова у отриманні узагальнених результатів опитувань, видача повідомлення про необхідність попередньої авторизації.

Міра реакції – у 100% спроб.

3.3 Висновки до розділу

Створено функціональні вимоги до розроблюваного програмного продукту. Вони представлені за допомогою Use Case діаграми та детальних сценаріїв використання. Нефункціональні вимоги до програмного продукту враховують наступних набір характеристик: функціональність, ефективність, переносимість, безпека програм, персональних даних та даних щодо тестування.

4 ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

4.1 Проектування архітектури програми

При проектуванні була використана клієнт-серверна архітектура, яка включає компоненти: Client - компонент клієнтської частини, Server - компонент серверної частини, File server.

Структура системи та взаємозв’язок її компонентів вказані на рис. 4.1.

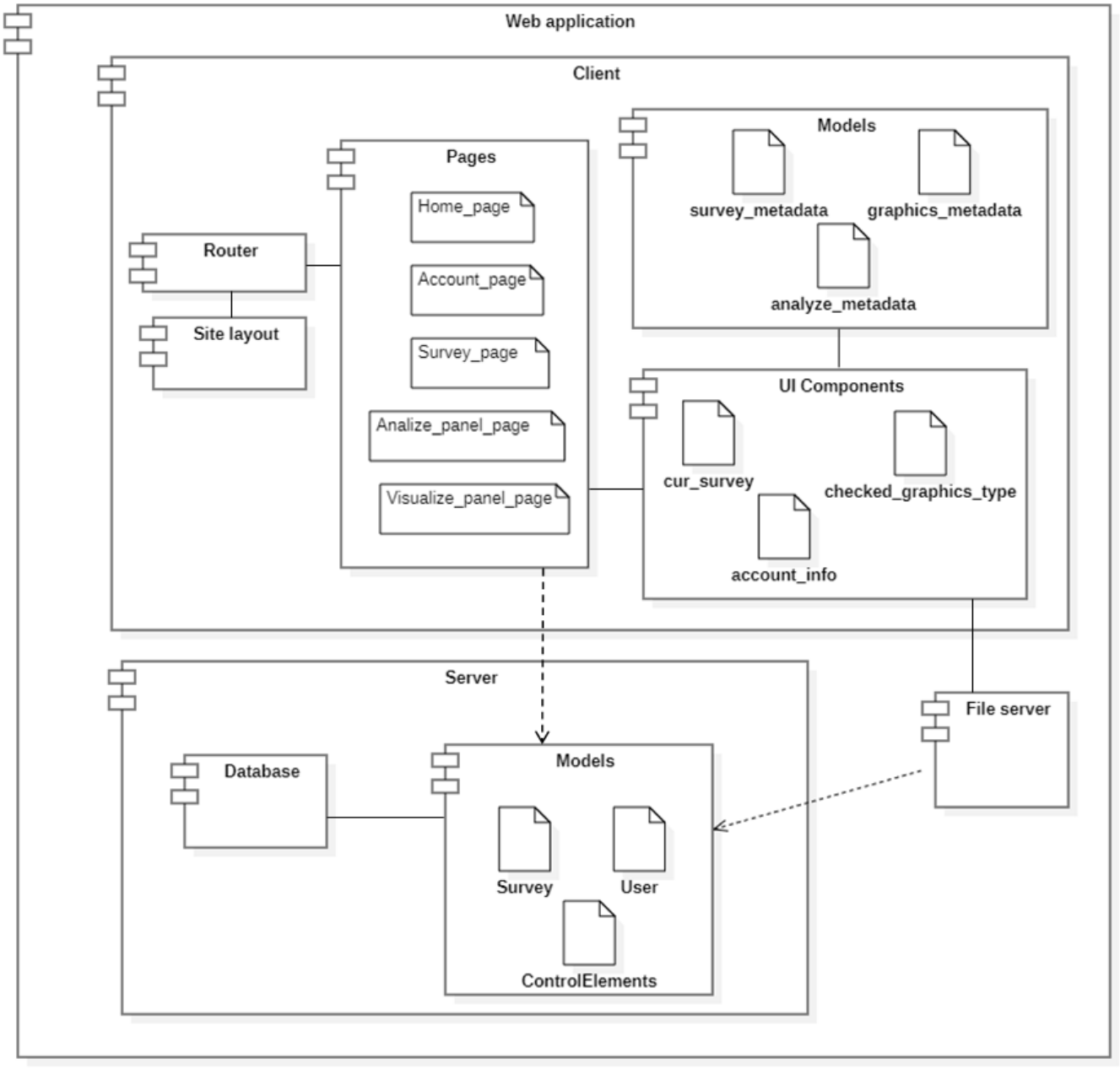

Рисунок 4.1 – Архітектура системи аналізу та візуалізації опитувань

Компоненти мають наступне призначення:

Client – збереження веб-сторінок макету системи, компоненти інтерфейсу користувача (для профілів Manager та Interviewee), а також моделі з метаданими.

Сторінки відповідають за представлення інформації про систему, реєстрацію та авторизацію користувача, проведення/проходження опитування, аналіз результатів опитування (надання узагальненої інформації), візуалізацію результатів опитування.

Моделі Client включають у себе моделі, що відповідають за опитування survey_metadata, аналіз analyze_metadata та візуалізацію інформації graphics_ metadata.

Server містить Datebase, моделі опитувань Survey, користувачів User та елементів управління ControlElements.

Стани та переходи між станами системи аналізу та візуалізації опитувань показані на рис. 4.2.

При вході в систему виконується перевірка профілю користувача – User identification. Якщо користувач зареєстрований, він повинен авторизуватись, інакше – спочатку зареєструватись. У разі провалу ідентифікації робота з системою неможлива.

Після цього система працює з «Info component for Interviewee» для респондентів чи «Info component for Manager» для дослідників (організаторів) опитувань.

Для користувача Manager доступні стани:

– Survey construction component – для роботи з опитуваннями (створення, редагування, видалення, видалення окремих тестів з опитувань, зміна назви опитування);

– Settings – встановлення налаштувань середовища системи;

– Surveys' result visualization – перегляд та аналіз узагальнених результатів опитувань.

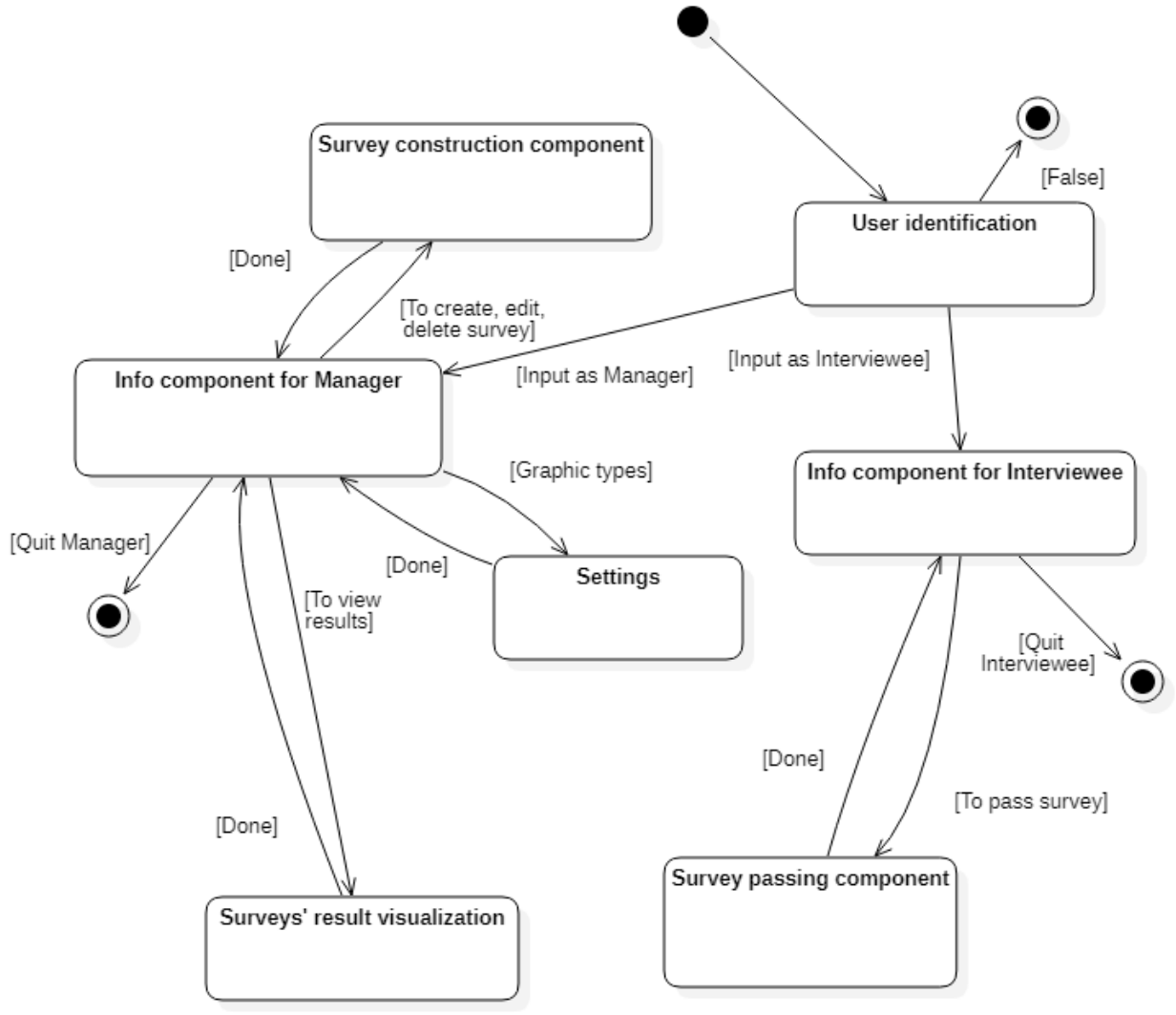

Рисунок 4.2 – Діаграма станів розроблюваної системи

Для користувача Interviewee доступне основний стан «Survey passing component» - проходження обраного опитування.

4.2 Основні діаграми роботи програмної системи

Для того, щоб продемонструвати які процеси відбуваються у системі опитувань використовуються діаграми діяльності.

На рис. 4.3 показана діаграма діяльності для роботи менеджера з опитуваннями.

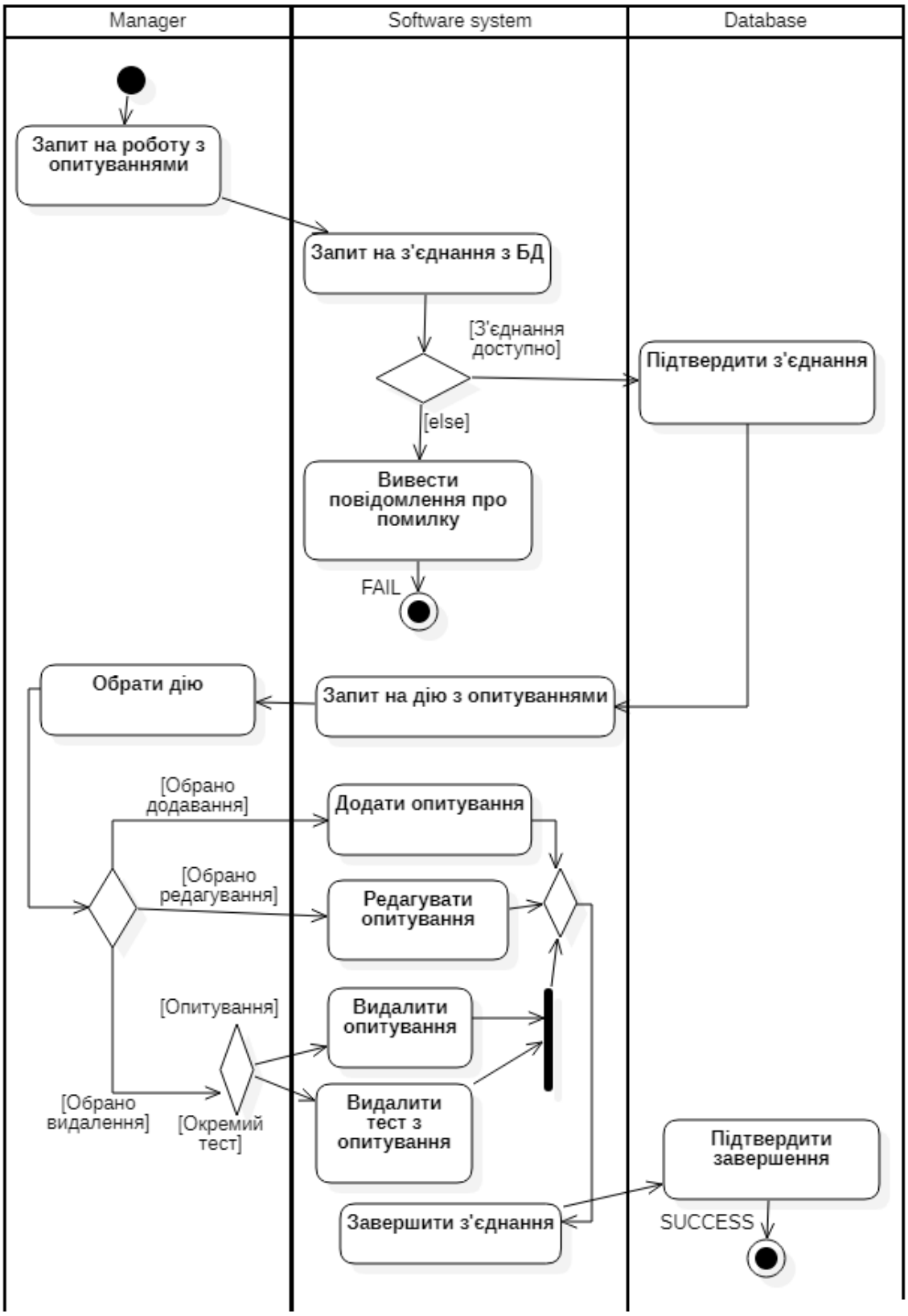

Рисунок 4.3 – Робота менеджера з опитуваннями

Менеджер вимагає з’єднання з базою даних, де зберігаються дані по опитуванням.

Після цього менеджер обирає дію, тобто що він бажає зробити:

– додати нове опитування;

– відредагувати будь-яке опитування системи (для цього потрібно спочатку обрати опитування);

– виконати видалення (можна обирати, чи це буде видалення всього опитування цілком, чи окремих тестів з опитування).

Діаграма є узагальненою, отже на ній не відображена можливість імпорту опитування, але це передбачається у блоці «Додати опитування».

Для того, щоб розглянути процес створення опитування більш детально, наведемо відповідну діаграму діяльності (рис. 4.4).

Після запиту на додавання нового опитування менеджер вводить його назву та визначає, хоче він створювати нове опитування власноруч чи імпортувати його з іншого опитування.

У разі вибору режиму імпорту система надає можливість обрати шлях до імпортуємого опитування та саме опитування, завантажує його у систему під введеним менеджером ім’ям. На цьому робота над створенням опитування завершується.

Якщо менеджер хоче формувати опитування власноруч, він обирає тип тесту і після цього вводить питання тесту. Після цього система визначає, чи потрібно продовжувати. Якщо так, то цикл формування опитування продовжується. Для кожного з тестів менеджер може обрати свій тип:

– Info block;

– Single choice;

– Multiple choice;

– Single text line;

– Multiple lines.

Після завершення додавання тестів в опитування у системі зберігається інформація про назву опитування, типи тестів, питання тестів, дату та час створення опитування.

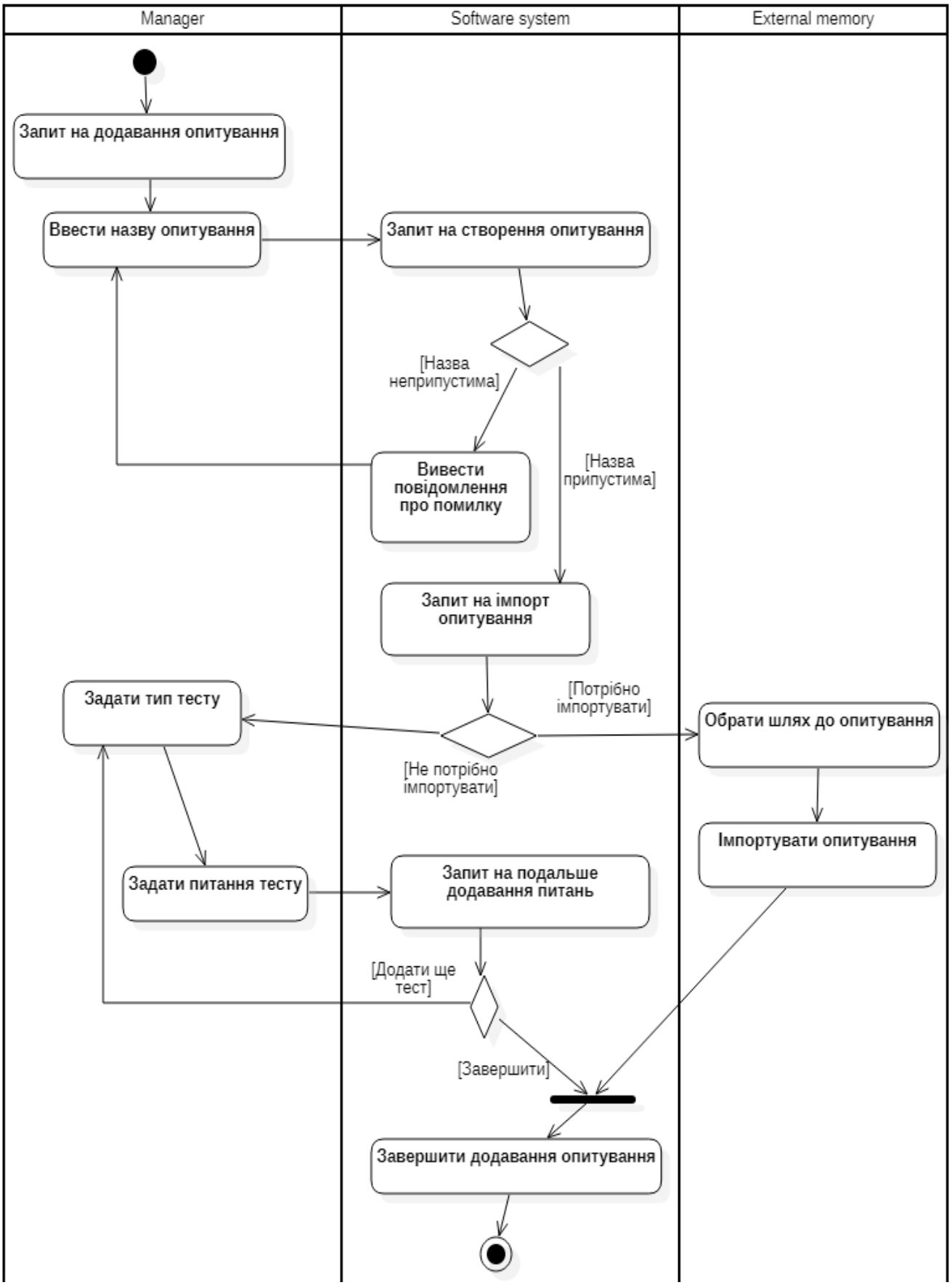

Рисунок 4.4 – Діаграма діяльності додавання менеджером нового опитування

На рис. 4.5 наведена діаграма діяльності роботи респондента Interviewee.

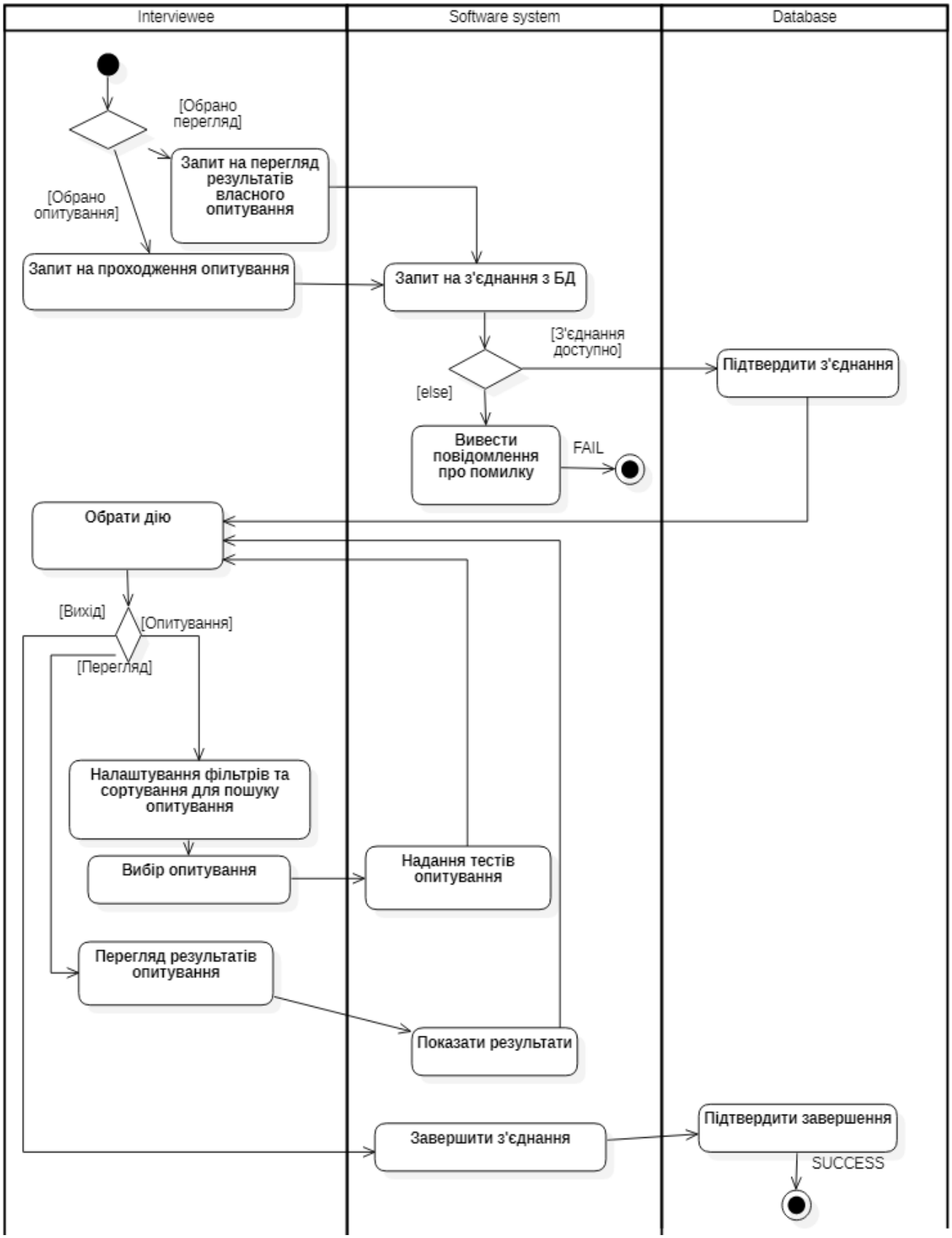

Рисунок 4.5 – Діаграма діяльності для роботи у режимі респондента

Interviewee обирає режим – проходження опитування чи перегляд результатів, та система надає йому відповідні можливості.

4.3 Розробка структури бази даних

Для збереження даних системи аналізу та візуалізації результатів опитувань використовується реляційна база даних. Вона має такі таблиці:

- User – будь-який користувач системи;
- Manager – таблиця з даними менеджерів (дослідників);
- Interviewee – таблиця з даними респондентів;
- Survey – таблиця для збереження даних опитувань;
- Interviewee+Survey – сеанси проходження опитувань;
- Result – таблиця результатів для окремих респондентів;
- SummaryResult – таблиця з узагальненими даними опитувань;
- SurveyTest – дані тестів опитувань;
- TestQuestion – питання тестів;
- ArchiveSurvey – опитування, з яких виконаний імпорт;
- TypeGraphics – типи графічної візуалізації результатів опитувань.

Тепер для кожної таблиці сформуємо структуру з визначенням полів та ключів.Структура таблиці User міститься у табл. 4.1.

Таблиця 4.1 – Структура даних таблиці User

Назва поля	Тип даних	Опис	Ключ
id	integer (8)	id користувача	PK
login	varchar (30)	Логін користувача системи	
password	varchar (30)	Пароль користувача системи	

Продовження табл. 4.1

Назва поля	Тип даних	Опис	Ключ
isInterviewee	boolean (1)	Прапорець чи є користувач тим, хто проходить опитування	
Avatar	varchar (2000)	Посилання на аватар системи	

Далі наведено структуру таблиці Manager (табл. 4.2).

Таблиця 4.2 – Структура таблиці Manager

Назва поля	Тип даних	Опис	Ключ
idManager	integer (8)	Ідентифікатор менеджера	PK
ManagerTelephone	integer (12)	Телефон менеджера	

Структура сутності Interviewee міститься у табл. 4.3.

Таблиця 4.3 – Структура таблиці Interviewee

Назва поля	Тип даних	Опис	Ключ
idInterviewee	integer (8)	Ідентифікатор особи, що проходить опитування	PK
Note	varchar (300)	Примітка щодо особи, що проходить опитування (є необов’язковою)	

Структури даних опитувань приведені у табл. 4.4.

Таблиця 4.4 – Структура таблиці Survey

Назва поля	Тип даних	Опис	Ключ
idSurvey	integer (8)	Ідентифікатор опитування	PK

Продовження табл. 4.4

Назва поля	Тип даних	Опис	Ключ
Name	varchar (50)	Назва опитування	
PageCount	integer (8)	Кількість сторінок в опитуванні	
TestCount	integer (8)	Кількість тестів в опитуванні	

У табл. 4.5 наведена структура сеансів проведення опитувань.

Таблиця 4.5 – Структура таблиці Interviewee+Survey

Назва поля	Тип даних	Опис	Ключ
idInterviewee	integer (8)	Ідентифікатор особи, що проходить опитування	PK
idSurvey	integer (8)	Ідентифікатор опитування	PK
IdResult	integer (8)	Ідентифікатор результату	
Date	Data (yyyy-mm-dd)	Дата проведення опитування	
Time	time (hh-mm-ss)	Час проведення опитування	

У табл. 4.6 – 4.7 містяться структури результатів опитувань – персональні та узагальнені.

Таблиця 4.6 – Структура таблиці Result

Назва поля	Тип даних	Опис	Ключ
IdResult	integer (8)	Ідентифікатор результату	PK
isInSummaryResult	bool (1)	Прапорець, який вказує чи увійшов цей результат у загальний	

Продовження табл. 4.6

Назва поля	Тип даних	Опис	Ключ
isShown	bool (1)	Прапорець, який вказує чи переглядається результат	
isSaved	bool (1)	Прапорець, який вказує чи зберігається результат	

Таблиця 4.7 – Структура таблиці SummaryResult

Назва поля	Тип даних	Опис	Ключ
IdSummaryResult	integer (8)	Ідентифікатор узагальненого результату	PK
idSurvey	integer (8)	Ідентифікатор опитування	
idTypeGraphics	integer (8)	Ідентифікатор графічного представлення узагальнених результатів	
isSummaryText	bool (1)	Прапорець, який вказує чи є результат узагальненням текстових відповідей	

У табл. 4.8 – 4.9знаходятся структури тестів та їх питань.

Таблиця 4.8 – Структура таблиці SurveyTest

Назва поля	Тип даних	Опис	Ключ
idSurveyTest	integer (8)	Ідентифікатор тесту опитування	PK
idTestQuestion	integer (8)	Ідентифікатор питання	

Продовження табл. 4.8.

Назва поля	Тип даних	Опис	Ключ
isText	bool (1)	Прапорець, який вказує чи є тест текстовим (Singletextline чи Multiplelines)	

Таблиця 4.9 – Структура таблиці TestQuestion

Назва поля	Тип даних	Опис	Ключ
idTestQuestion	integer (8)	Ідентифікатор питання тесту	PK
TextQuestion	text (1000)	Текст питання	
Answer	integer (3)	Номер правильної відповіді (якщо це передбачається типом тесту)	

Для збереження даних опитувань, з яких були імпортовані тести, використовується ArchiveSurvey (табл. 4.10).

Таблиця 4.10 – Структура таблиці ArchiveSurvey

Назва поля	Тип даних	Опис	Ключ
IdArchiveSurvey	integer (8)	Ідентифікатор імпортованого опитування	PK
idSurvey	integer (8)	Ідентифікатор опитування	
Link	varchar (1000)	Посилання на імпортоване опитування	
Note	varchar (1000)	Примітка (необов’язкова)	

Продовження табл. 4.10.

Назва поля	Тип даних	Опис	Ключ
Date	Data (yyyy-mm-dd)	Дата імпортування опитування	
Time	time (hh-mm-ss)	Час імпортування опитування	

Наприкінці наведемо структуру типів графічної візуалізації результатів опитувань (табл. 4.11).

Таблиця 4.11 – Структура таблиці TypeGraphics

Назва поля	Тип даних	Опис	Ключ
IdTypeGraphics	integer (8)	Ідентифікатор типу візуалізації	PK
NameTypeGraphics	integer (8)	Назва типу візуалізації	
isValid	bool (1)	Прапорець, який вказує чи є даний тип припустимим для використання	

Структура таблиць реляційної бази даних з взаємними зв’язками типів один-до-багатьох та один-до-одного наведена на рис.4.6.

Представлена база даних є нормалізованою, у ній усунуті надмірності даних, а також виявлені функціональні залежності. У разі виключення надмірності даних гарантується компактність наборів даних, уникнення зайвого дублювання даних та відсутність аномалій вставки, видалення, редагування після програмної реалізації БД.

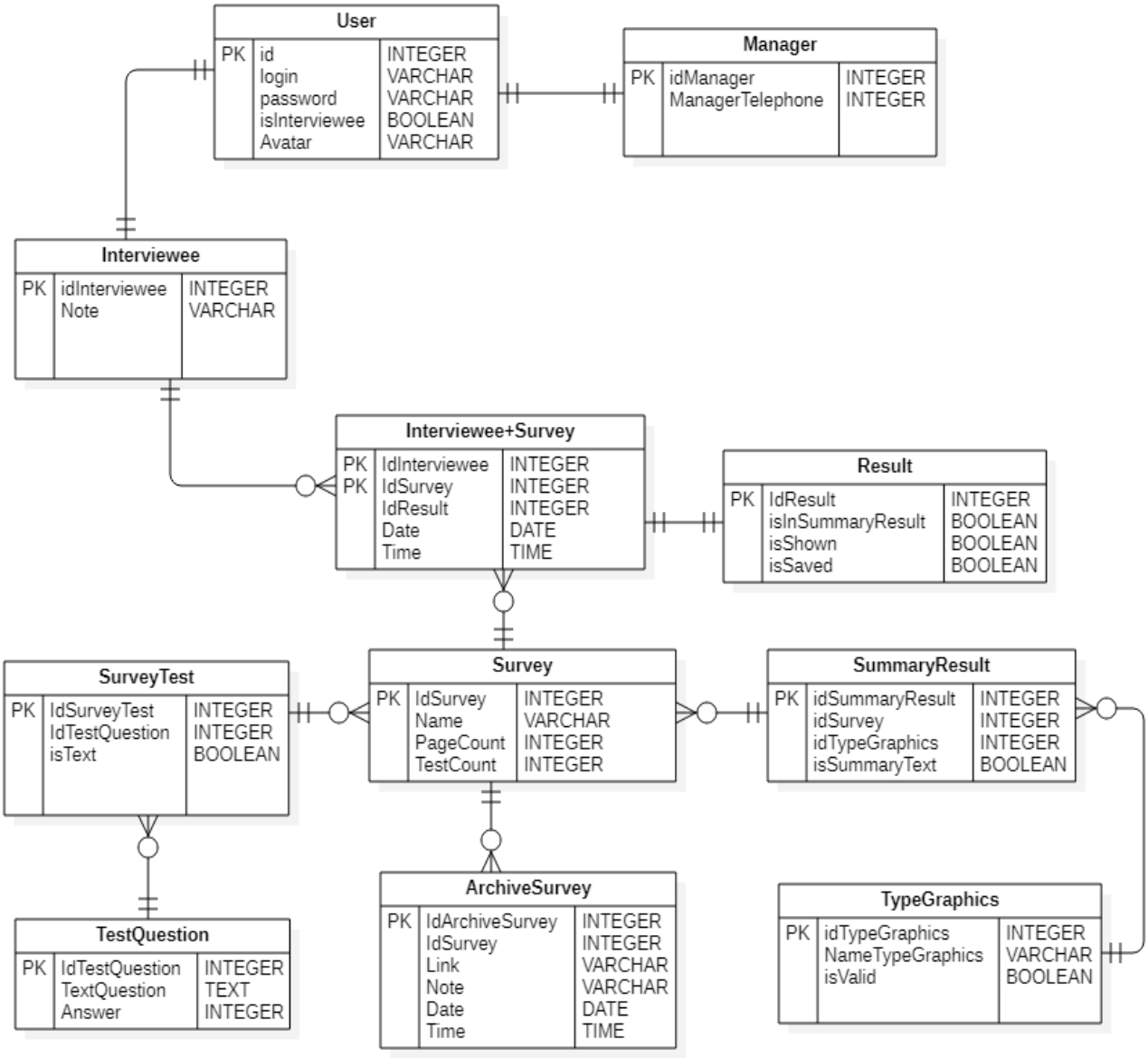

Рисунок 4.6 – Схема бази даних

4.4 Проектування структури класів

Шаблон проектування програмного забезпечення Model-View-Controller (MVC) сприяє поділу програмних систем на елементи моделі, представлення та контролера. Представлення представляють інтерфейс користувача, моделі представляють системні дані, а контролери обробляють запити, надіслані представленнями, і координують взаємодію між представленнями та моделями.

Серверна частина програмної системи побудована за архітектурним шаблоном MVC (рис. 4.7).

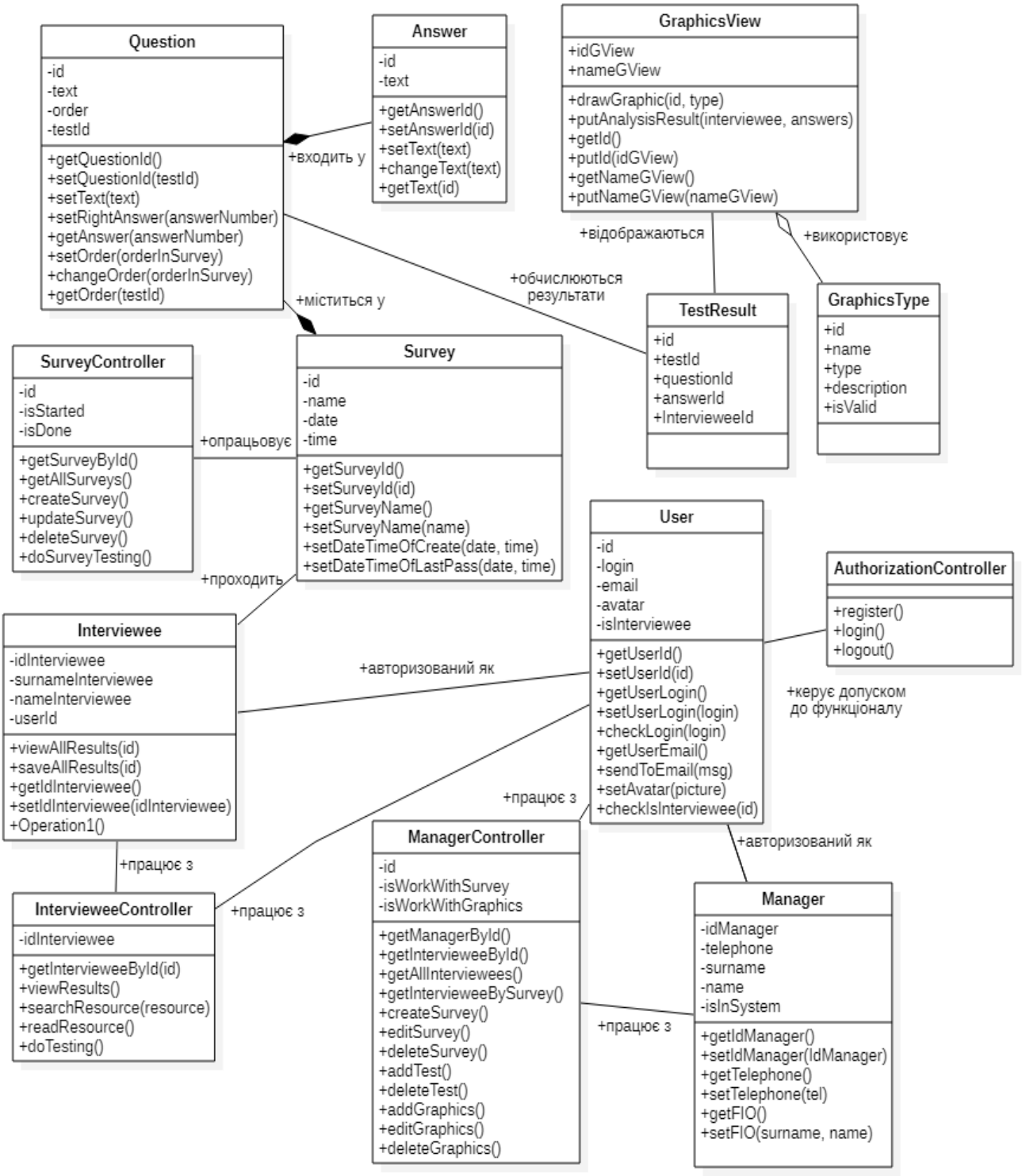

Рисунок 4.7 – Діаграма класів системи

Інструменти опитування використовуються для проведення онлайн-опитувань з метою збору цінних даних від учасників опитування. Інструменти опитування можуть використовувати маркетологи, дослідники та власники

бізнесу. Дані про те, що подобається споживачам, а що ні, можуть сформувати вашу бізнес-стратегію.

Контролер AuthorizationController виконує авторизацію користувачів Manager та Interviewee на сервері; ManagerController управляє даними дослідників (менеджерів), IntervieweeController керує даними респондентів, SurveyController відповідає за управління опитуваннями у БД.

Клас AuthorizationController має метод login(), що використовується для авторизації користувачів.

Клас ManagerController має наступні методи:

метод getManagerById() – завантаження даних менеджера за його ідентифікатором;

метод getIntervieweeById() – завантаження даних респондента за його ідентифікатором;

метод getAllInterviewees() – отримання даних всіх респондентів;

метод getIntervieweeBySurvey() – завантаження даних респондентів, які пройшло певне опитування;

метод createSurvey() – створення опитування та збереження у БД;

метод editSurvey() – редагування опитування та збереження у БД;

метод deleteSurvey() – видалення опитування з БД;

метод addTest() – додавання до опитування нового тесту;

метод deleteTest() – видалення з опитування певного тесту;

метод addGraphics() – додавання нового типу графічного представлення результатів опитування;

метод editGraphics() – редагування типу графічного представлення результатів опитування;

метод deleteGraphics() – видалення типу графічного представлення результатів опитування.

Клас IntervieweeController має наступні методи:

метод getIntervieweeById() – отримання респондента за вказаним ідентифікатором;

метод viewResults() – запит на перегляд респондентом результатів власного опитування (чи опитувань, якщо їх було декілька);

метод searchResource() – запит на пошук респондентом всіх опитувань;

метод readResource() – запит на перегляд респондентом певного опитування;

метод doTesting() – проходження опитування респондентом.

Клас SurveyController має наступні методи:

метод getSurveyById() – отримання опитування з БД за його ідентифікатором;

метод getAllSurveys() – завантаження всіх опитувань, що є створеними та збереженими у БД;

метод createSurvey() – створення нового опитування;

метод updateSurvey() – редагування опитування, яке вибрано;

метод deleteSurvey() – видалення опитування та всіх його тестів;

метод doSurveyTesting() – проведення певного опитування та проходження всіх тестів.

Клас User містить наступні дані: ідентифікатор, логін, електронна адреса, аватар, чи є користувач респондентом.

Клас Interviewee містить дані: ідентифікатор респондента, його номер телефону, прізвище та ім’я, код користувача.

Клас IntervieweeSurvey містить дані: ідентифікатор респондента, назву (ім’я) респондента.

Клас Survey пов’язаний з класом Question з використанням відношення композиції, включає ідентифікатор опитування, його та назву опитування, та порядок питань.

Клас Question містить ідентифікатор питання, його текст, вагу питання, порядок у в опитуванні та ідентифікатор тесту опитування.

Об’єкт класу Answer пов’язаний з об’єктом класу Question відношенням композиції, клас Answer містить ідентифікатор відповіді на тест опитування.

Клас TestResult містить дані: ідентифікатори результату опитування, ідентифікатор тексту питання, ідентифікатор тексту відповіді, ідентифікатор респонденту.

4.5 Висновки до розділу

У четвертому розділі було виконано проектування архітектуру програмноїсистеми відповідно до шаблону проектування MVC.

Створено діаграму станів та переходів для загального розуміння роботи системи та діаграми діяльності для уточнення процесів роботи менеджера з опитуваннями, додавання менеджером нового опитування та роботи у режимі респондента. Виконано проектування структури реляційної бази даних та програмних класів системи.

5 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ОПИТУВАНЬ

5.1 Набір інструментальних засобів розробки

При розробці програмної системи для аналізу та візуалізації результатів опитувань було використано мову програмування Java для реалізації серверної частини, засоби JavaScript з бібліотекою Redux, HTML та CSS для клієнтської частини, середовище розробки IntelliJIDEA, реляційну базі даних MySQL, фреймворк Bootstrap та систему контролю версій Git.

Їх взаємозв’язок показано на рис. 5.1.

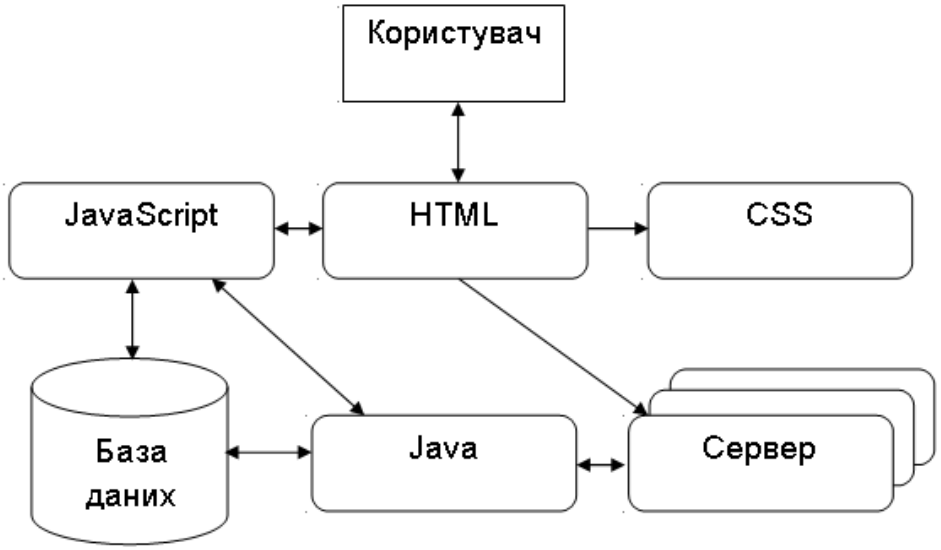

Рисунок 5.1 – Стек використаних технологій при розробці програмної системи

Розглянемо інструменти розробки більш детально.

«Java є мовою програмування, за допомогою якої розробники програмного забезпечення створюють різни прикладні застосунки для комп’ютерів, смартфонів, планшетів та інших інтелектуальних пристроїв. Особливістю програм на Java є те, що вони можуть запускатись на будь-яких комп’ютеризованих пристроях, які працюють під різними операційними системами, причому без повторної компіляції коду.Для їх виконання необхідно лише встановити середовище для виконання – JRE (Java Runtime Environment).

JRE розроблені для багатьох операційних систем – Linux(x86,x64), Mac OS X64, Solaris, Windows (x86,x64), завдяки чому код Java працює майже на всіх різновидах комп’ютерів та операційних систем.

JRE забезпечує безпечну та зручну роботу застосунків на Java, тому користувачі можуть не турбуватись про несанкціоноване втручання до ресурсів свого персонального комп’ютера з боку стороннього Java коду. Необхідно лише періодично оновлювати JRE. Вбудована технологія забезпечення безпеки Java включає в себе значний набір API (Application Programming Interface) механізмів та додаткових інструментів, включаючи широковідомі та надійні алгоритми та протоколи безпеки.» [9]

Для розробки клієнтської частини використані складові мови JavaScript [10].

«JavaScript – це мова програмування, що дозволяє зробити Web-сторінку інтерактивною, тобто такою що реагує на дії користувача.

Послідовність інструкцій (що називається програмою, скриптом або сценарієм) виконується інтерпретатором, вбудованим в звичайний Web -браузер. Іншими словами, код програми вбудовується в HTML-документ і виконується на боці клієнта. Для виконання програми не потрібно навіть перезавантажувати Web-сторінку, всі програми виконуються в відповідь на будь-яку подію. Наприклад, перед відправленням даних форми можна перевірити їх на допустимі значення і, якщо значення не відповідають очікуваним, заборонити відправлення даних.

JavaScript - об'єктно-орієнтована скриптова мова програмування і є діалектом мови ECMAScript.

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів застосунків. Найбільш широке застосування знаходить у браузерах як мова сценаріїв для надання інтерактивності веб-сторінкам.

Основні архітектурні риси:

- динамічна типізація,
- автоматичне керування пам'яттю,
- прототипне програмування,

– функції як об'єкти першого класу.» [11]

Реляційні бази даних є зручним способом збереження усіх даних програмної системи [12].

«Sqlite – це вбудована реляційна база даних, з відкритим вихідним кодом. Тобто вона не використовує звичну нам модель роботи бази даних клієнт-сервер і не є окремим працюючим процесом. Наприклад, у базі даних Mysql движок Sqlite стає якби частиною веб-застосунку. При такому підході база даних Sqlite (з усіма таблицями) являє собою звичайний текстовий файл, який можна розташувати в зручному місці.

Розглянемо основні переваги Sqlite.

Самодостатність – як було сказано, базі даних Sqlite не потрібен окремий сервер для роботи. Движок Sqlite вбудовується прямо в застосунок і потребує лише доступ до файлів.

Простота встановлення та налаштування нової бази дуже проста і не потребує втручання системних адміністраторів.

Надає як процедурний, так і об’єктно-орієнтований інтерфейс для роботи.

Висока продуктивність – двигун споживає дуже мало ресурсів і не витрачається час на відправку даних до окремого виділеного сервера.

Sqlite чудово підходить для веб-застосунків, коли основна маса запитів являє собою запити на читання.» [13]

«Redux - це бібліотека JavaScript з відкритим кодом для управління станом програми. Redux зазвичай використовується з бібліотеками, такими як Angular або React, для створення інтерфейсів користувача.

Керувати станом кожного компонента в застосунку стає важко, коли розмір програми стає надзвичайно великим. Redux допомагає в оновленні та підтримці стану кожного компонента в застосунку.» [14]

Для створення зручного та привабливого програмного інтерфейсу потрібно використовувати найкращі практики та інструменти [15, 16].

5.2 Реалізація інтерфейсу користувача

Розглянемо декілька прикладів форм інтерфейсу користувача. На рис. 5.1 показано початкове вікно системи, де можна побачити поточні можливості щодо створення опитування та налаштування системи.

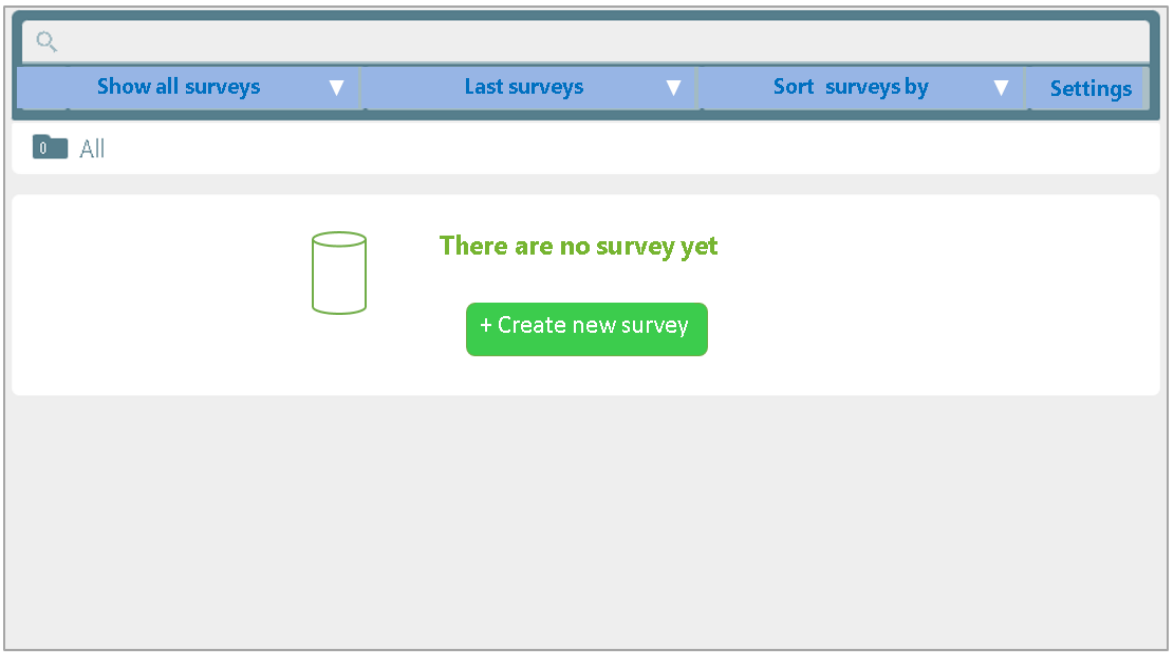

Рисунок 5.1 – Початкова форма програми

На рис. 5.2 показано створення нового опитування. Прицьому користувач повинен натиснути на кнопку для підтвердження дій та ввести назву для нового опитування.

Після вибору опитування та типу тесту користувач потрапляє у вікно вибору типу тесту (рис. 5.3). Користувач Manager повинен визначитись стосовно тестування по опитуванням. Відповідно до обраного типу опитування користувачеві подальше надаються тестові питання. Для виходу з режиму достатньо натиснути кнопку «Cancel».

На рис. 5.4 показаний приклад тесту типу «Single choice». Це достатньо розповсюджений тип тесту, він передбачає вибір однієї відповіді на питання з декількох варіантів.

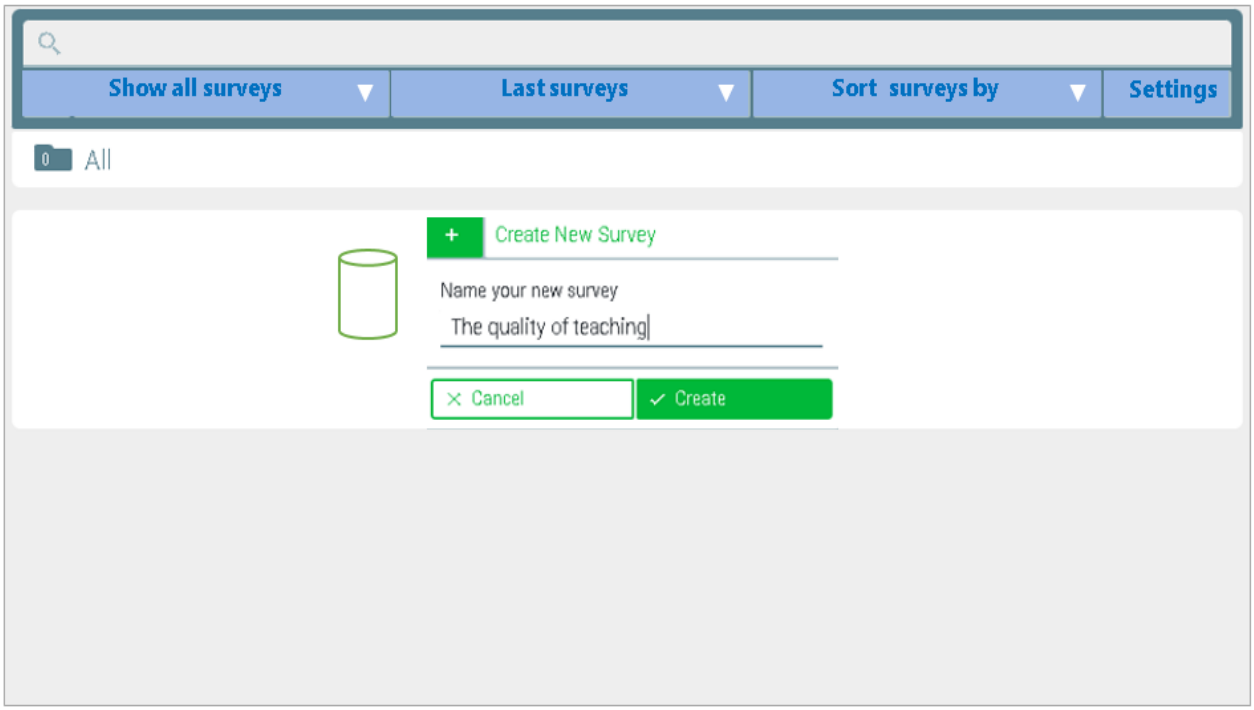

Рисунок 5.2 – Створення нового опитування

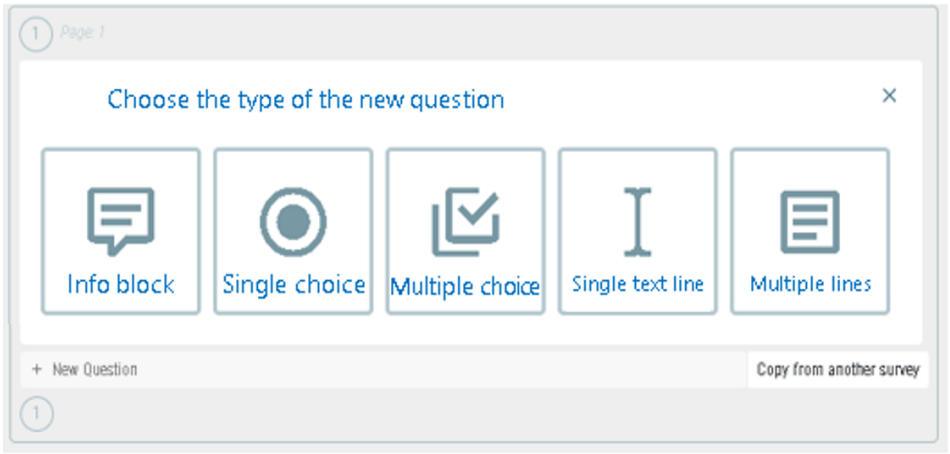

Рисунок 5.3 – Вибір типу тесту для опитування

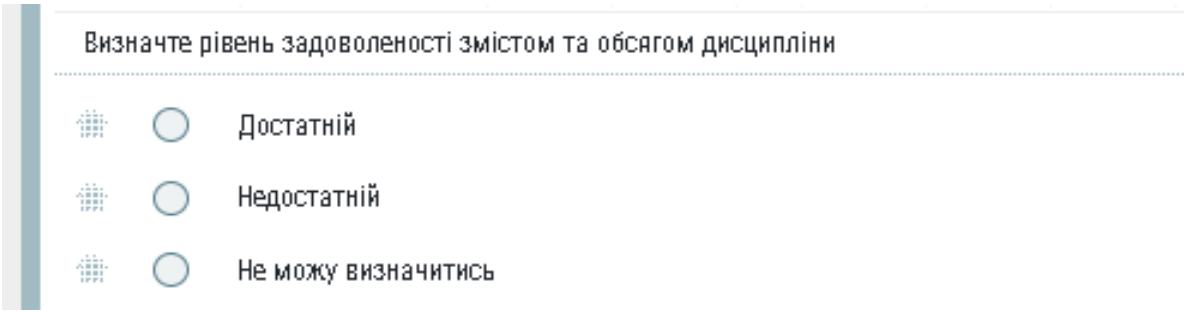

Рисунок 5.4 – Формат тесту типу «Single choice»

На рис. 5.5 показана форма тесту для користувача Manager з кнопкою «Add option», яка дозволяє додавати необхідну кількість питань у тест. Для створення нового тесту потрібно натиснути кнопку «New Question».

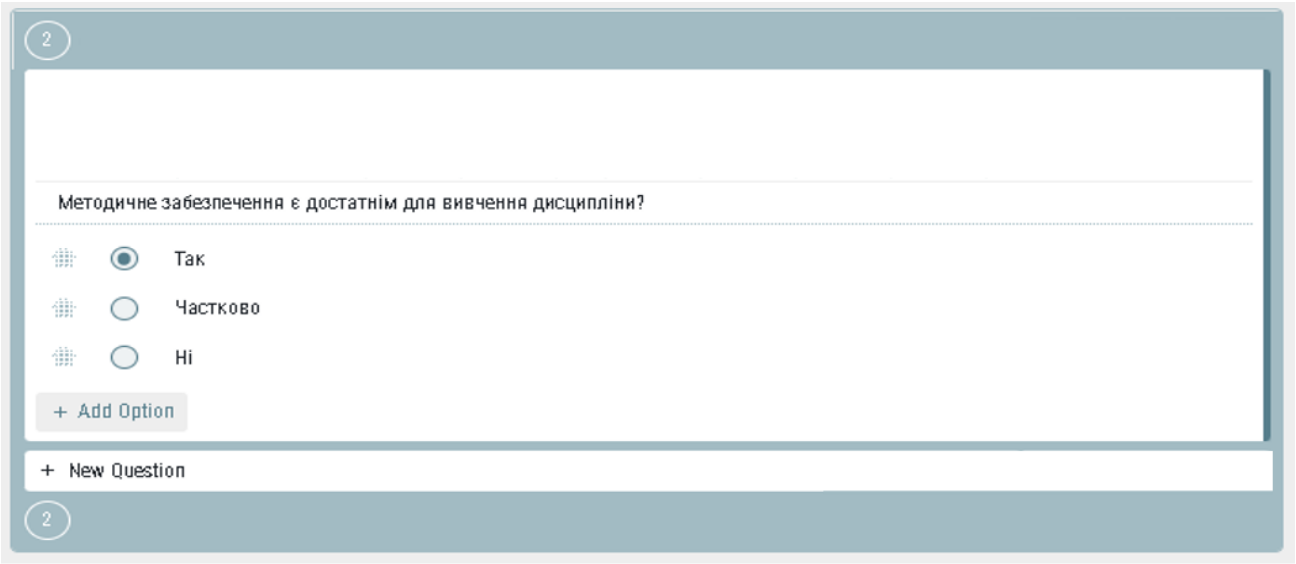

Рисунок 5.5 – Приклад тесту типу «Single choice»

На рис. 5.6 наведена форма тесту типу «Multiple choice» для користувача Manager. Цей тип передбачає вибір декількох варіантів відповіді на поставлене питання.

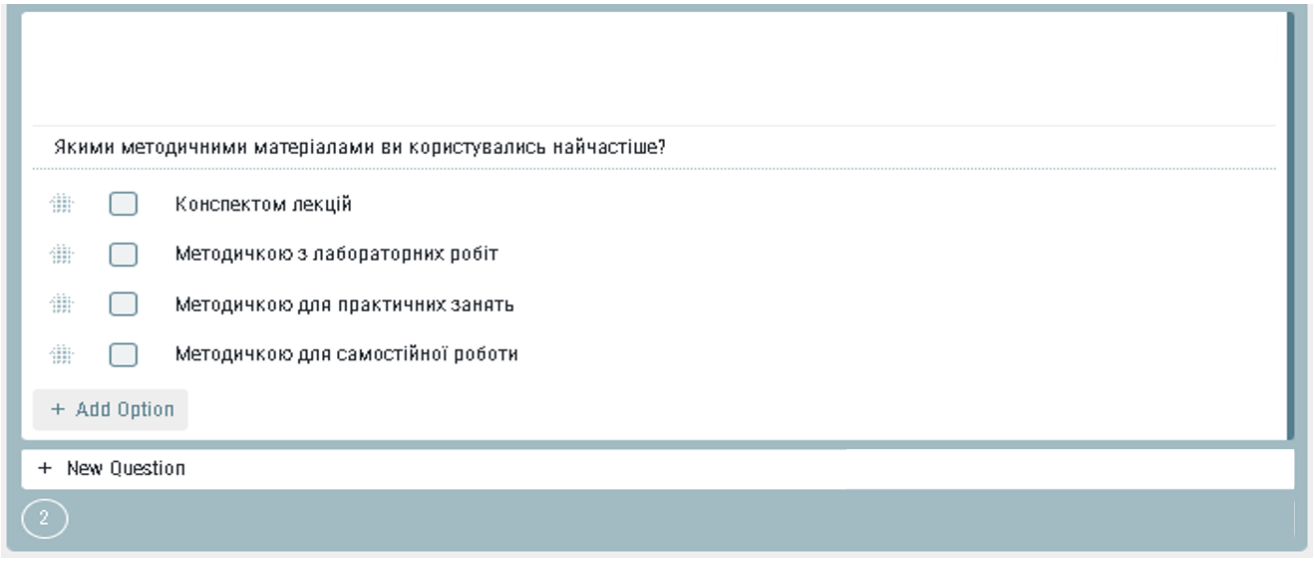

Рисунок 5.6 – Форма тесту типу «Multiple choice»

Після проходження опитування респондент може переглянути власні результати. При цьому він може обрати перегляд всіх результатів чи вибрати деякі окремі. Так, на рис. 5.7 показаний перегляд результатів першого та третього тестів в опитуванні.

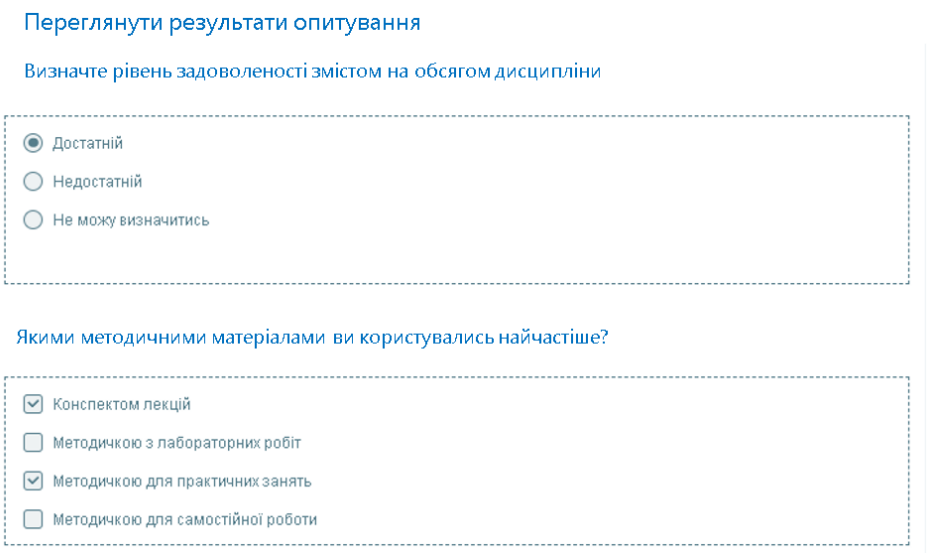

Рисунок 5.7 – Вибірковий перегляд результатів опитування

Користувач Manager може переглядати характеристики створеного опитування (рис. 5.8):

Created/Modified – час створення/останнього редагування,

Status – New для нового опитування,

Pages – кількість сторінок,

Questions – кількість питань,

Translations – скільки разів було пройдено дане опитування.

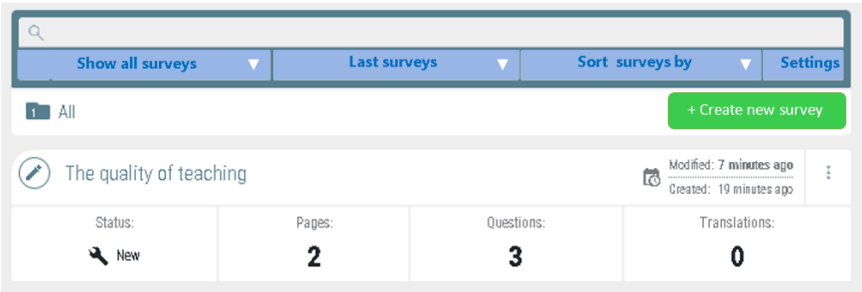

Рисунок 5.8 – Характеристики опитування

На рис. 5.9 показана форма для вибору користувачем Manager типу візуалізації узагальненого результату.

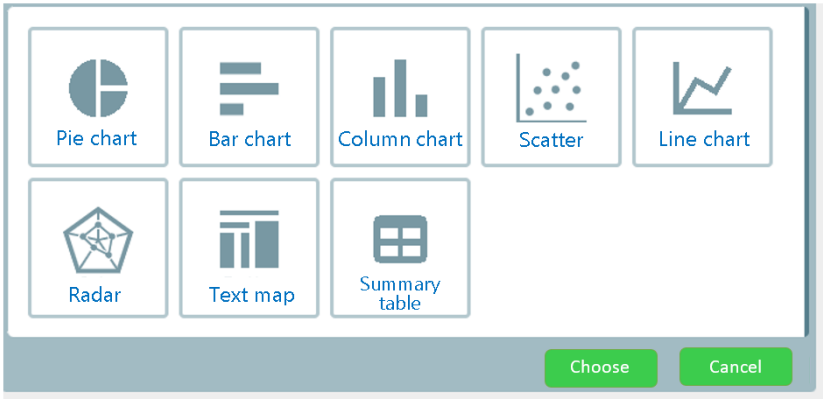

Рисунок 5.9 – Вибір типу візуалізації результатів опитування

Якщо типом тесту був Single text line або Multiple lines, то при виборі типу візуалізації «Text map» можна подивитись біграми – пари слів, які зустрічаються поряд в опитуваннях (рис. 5.10). Інтенсивність лежить у межах від 0 до 1 та розраховується відповідно до показника TF-IDF. Колір комірки показує інтенсивність: зелений – найменш інтенсивний, червоний – найбільш.

Опитування / Біграми	Survey1. Answer 1	Survey1. Answer 2	Survey1. Answer 3	Survey1. Answer 4	Survey1. Answer 5	Survey1. Answer 6	Survey1. Answer 7	Survey1. Answer 8	Survey1. Answer 9	Survey1. Answer 10	Survey1. Answer 11	Survey1. Answer 12	Survey1. Answer 13	Survey1. Answer 14
контрольна робота			0.02	0.01		0.06	0.01			0.05	0.43	0.08	0.02	0.34
програмний код			0.20			0.03	0.04					0.01	0.01	0.01
самостійна робота				0.02	0.03		0.01	0.44	0.25	0.07	0.04		0.02	0.09
виключення світла	0.10	0.02	0.01			0.07	0.04		0.11			0.23	0.17	
лабораторна робота							0.26		0.30		0.08		0.18	0.04
платформа el			0.03		0.02	0.11		0.04			0.17			
поганий інтернет	0.12		0.01		0.05	0.02		0.03	0.02		0.02	0.46		0.01
об'ємне завдання		0.02	0.05	0.01	0.24	0.13	0.02	0.03	0.04	0.02	0.02	0.03	0.06	
створення проекту		0.52	0.02	0.14		0.12	0.02	0.01	0.02	0.27		0.01		
протокол роботи		0.08		0.15				0.01	0.03	0.03		0.01		
склад команди		0.03		0.06					0.03	0.23		0.09		
тестування коду		0.21		0.80		0.02		0.07	0.14	0.24	0.01	0.06		
приклад роботи		0.03					0.78		0.14	0.03			0.03	
замало часу		0.02	0.01	0.01		0.22		0.06		0.03	0.02			0.07

Рисунок 5.10 – Приклад візуалізації результатів обробки текстових питань

Цей тип візуалізації дозволяє користувачеві Manager оцінити які проблеми хвилюють респондентів.

На рис. 5.11 показано приклади візуалізації при виборі типів Pie chart, Bar chart, Column chart та Scatter. Це – узагальнені результати відповідей 20-ти респондентів на тестове питання стосовно рівня задоволеності змістом та обсягом певної дисципліни.

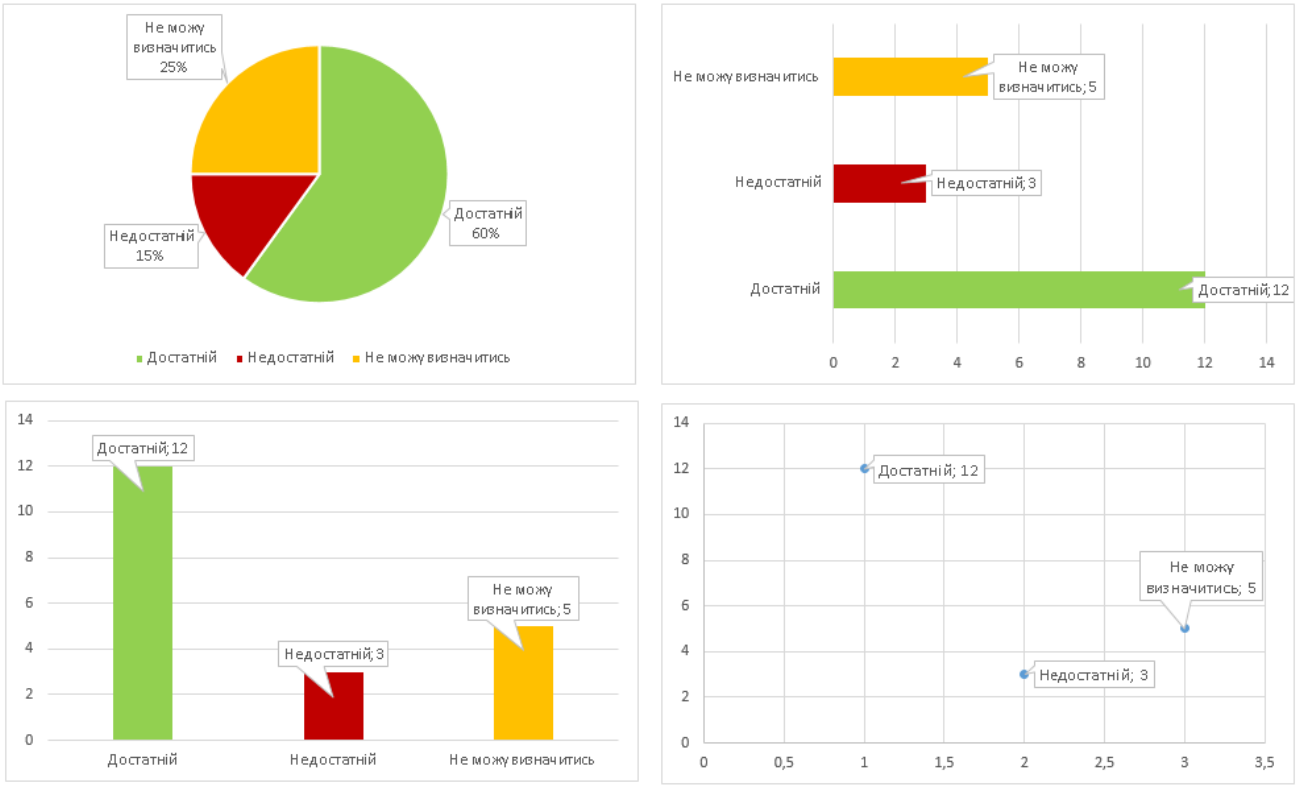

Рисунок 5.11 – Візуалізація

5.3 Висновки до розділу

У даному розділі проаналізовано інструменти розробки – мови та бібліотеки для реалізації серверної та клієнтської частин. Наведено приклади інтерфейсу програмної системи для роботи в режимах Manager та Interviewee.

6 ТЕСТУВАННЯ РОБОТИ ПРОГРАМИ АНАЛІЗУ ТА ВІЗУАЛІЗАЦІЇ РЕЗУЛЬТАТІВ ОПИТУВАНЬ

6.1. Сценарії функціонального тестування

Для перевірки коректності роботи програмної системи створено тестові сценарії щодо функціональних вимог (табл. 6.1).

Таблиця 6.1 – Тестові сценарії

Функціональна вимога до програми	Ідентифікатор тестового сценарію	Вхідні дані тесту	Мета проведення тесту
1	2	3	4
Створення опитування	TC1	Набір тестів для опитування	Перевірити можливість створення нового опитування
Імпорт опитування	TC2	Опитування на зовнішньому носії, яке можна імпортувати	Перевірити можливість імпортування опитування
Створення тесту	TC3	Набір питань для тесту, тип тесту	Перевірити можливість додавання тесту до опитування
Конструювання тесту	TC4	Набір питань для тесту, тип тесту, графічні форми	Перевірити можливість візуалізації форми для тесту на екрані

Продовження таблиці 6.1

1	2	3	4
Імпорт тесту	ТС5	Опитування на зовнішньому носії, з якого можна імпортувати тест, номер тесту	Перевірити можливість імпортування тесту
Редагування опитування	ТС6	Набір оновлених тестів для редагування опитування	Перевірити можливість редагування опитування
Редагування тесту	ТС7	Набір оновлених питань для редагування тесту	Перевірити можливість редагування тесту
Видалення тесту	ТС8	Дані опитування, номер тесту для видалення	Перевірити можливість видалення тесту
Видалення опитування	ТС9	Назва опитування для видалення	Перевірити можливість видалення опитування
Перегляд узагальнених результатів	ТС10	Дані проведення опитування, тип візуалізації	Перевірити можливість відтворення узагальнених результатів на екрані
Вибір типу візуалізації	ТС11	Набір типів візуалізації	Перевірити можливість вибору типу для налаштування діаграми

Продовження таблиці 6.1

1	2	3	4
Авторизація у системі	ТС12	Набір даних менеджера чи респондента	Перевірити можливість авторизації користувача
Реєстрація у системі	ТС13	Набір даних про нового менеджера чи респондента	Перевірити можливість реєстрації користувача
Проходження опитування	ТС14	Дані тестів опитування, авторизований респондент	Перевірити можливість проходження поточного опитування
Вибір опитування	ТС15	Список опитувань	Перевірити можливість вибору потрібного опитування
Перегляд результатів	ТС16	Результати проходження тесту	Перевірити можливість перегляду респондентом власних результатів

Сценарії ТС1 – ТС16 були використані для проведення функціонального тестування, результати підтвердили коректну роботу програмної системи.

6.2 Експериментальна перевірка зручності роботи

У експериментальному тестуванні прийняли участь 20 осіб. Вони повинні були оцінити за шкалою від 1 (дуже погано) до 10 (дуже добре) характеристики зручності.

З них 10 осіб працювали без системи, користуючись іншими програмними продуктами, та 10 осіб використовували дану систему.

Учасники експерименту вносили оцінки у спеціально розроблені анкети. Усереднені результати оцінок зведені у табл. 6.2.

Таблиця 6.2 – Експериментальне тестування зручності використання

Характеристики	Без програмної системи	З програмною системою
Зручність візуалізації графічних елементів	5,6	7,8
Зручність пошуку опитувань	4,5	8,6
Зручність аналізу результатів опитувань	6,3	8,9
Усереднена оцінка	5,5	8,4

Отже, є підстави вважати, що з використанням розробленої програмної системи зручність аналізу та візуалізації результатів опитувань зросла у 1,54 рази.

6.3 Висновки до розділу

У шостому розділі проведено функціональне тестування програмної системи на базі тестових сценаріїв. Проведено експерименти з визначенням зручності використання за участю 20 осіб, які працювали у режимах менеджера та респондента. Отримані результати щодо підвищення зручності у 1,54 разів.

ВИСНОВКИ

У роботі розроблено програмну систему, що дозволяє організовувати опитування, обробляти їх результати, аналізувати та візуалізувати результати відповідно до обраного користувачем виду представлення. Використання програми дозволило підвищити зручність аналізу та візуалізації результатів опитувань у середньому у 1,54 рази.

Для досягнення мети роботи виконано наступні кроки:

– вивчення предметної області та проблем, пов’язаних з проведенням опитувань;

– аналіз переваг та недоліків програмних аналогів, які дозволяють проводити опитування та візуалізувати результати;

– визначення типів питань, які можуть входити в опитування та систематизація графічного представлення результатів;

– побудова засобів візуалізації текстової інформації як складової результату опитування;

– визначення вимог для розробки програмної системи;

– створення архітектурного дизайну застосунку, визначення складових елементів системи, формалізація основних процесів системи;

– аналіз способів збереження даних та проектування структури бази даних;

– проектування структури класів та їх взаємозв’язків;

– вибір інструментів розробки, створення програмного інтерфейсу та програмування функціоналу системи;

– тестування роботи функціоналу та визначення характеристики системи «зручність використання».

Результати роботи висвітлювались у матеріалах XXII Всеукраїнської науково–технічної конференції молодих вчених, аспірантів та студентів «Стан, досягнення та перспективи інформаційних систем і технологій», що відбулась 21-22 квітня 2022 р.

СПИСОК ЛІТЕРАТУРИ

1. SurveyGizmo [Електронний ресурс] – Режим доступу: URL: https://www.alchemer.com/ (Дата останнього звернення: 01.10.2022).

2. OnlineTestPad [Електронний ресурс] – Режим доступу: URL: https://onlinetestpad.com/ (Дата останнього звернення: 01.10.2022).

3. MySurveylab [Електронний ресурс] – Режим доступу: URL: https://www.surveylab.com/ (Дата останнього звернення: 01.10.2022).

4. Surveynuts [Електронний ресурс] – Режим доступу: URL:https://surveynuts.com/ (Дата останнього звернення: 03.10.2022).

5. SurveyMonkey [Електронний ресурс] – Режим доступу: URL: https://www.surveymonkey.com/ (Дата останнього звернення: 03.10.2022).

6. Нір Еяль, Раян Гувер. На гачку. Як створити продукт, що чіпляє. Наш Формат – 2017 р. – 192 с.

7. Петрик М.Р. Моделювання програмного забезпечення: науково методичний посібник / М.Р. Петрик, О.Ю. Петрик – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2015. – 200 с.

8. Левус Є.В., Мельник Н.Б. Вступ до інженерії програмного забезпечення: навч. посіб.− Л.:Видав. Львівської політехніки, 2018. – 246с.

9. Мова програмування Java та платформа Javafx: приклади застосування [Електронний ресурс] – Режим доступу: URL: https://dut.edu.ua/ua/news-1-626-6031-mova-programuvannya-java-ta-platforma-javafx-prikladi-zastosuvannya_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy (Дата останнього звернення: 10.10.2022).

10. Браун Этан. Изучаем JavaScript: руководство по созданию современных веб-сайтов/Вильямс, 2017 - 368с.

11. Мова JavaScript та її можливості [Електронний ресурс] – Режим доступу: URL: https://sites.google.com/site/webtehnologiietawebdizajn/mova-javascript-ta-ieie-mozlivosti (Дата останнього звернення: 10.10.2022).

12. Малахов, Є. В. Проектування баз даних та їх реалізація засобами стандартного SQL та PostgreSQL : навч. посібник / Є.В. Малахов, О.А. Блажко, М.Г. Глава. – О. : ВМВ, 2012.- 248 с.

13. Введення в базу даних Sqlite [Електронний ресурс] – Режим доступу: URL: http://yoip.com.ua/vvedennya-v-bazu-danih-sqlite/ (Дата останнього звернення: 10.10.2022).

14. Що таке Redux? [Електронний ресурс] – Режим доступу: URL: https://uk.education-wiki.com/3707369-what-is-redux.

15. Alan Cooper, Robert Reimann, David Cronin, Christopher Noessel. About Face: The Essentials of Interaction Design 4th Edition – Wiley – Cooper – 2019 p. – 720 с.

16. Еллен Лаптон, Дженніфер Коул Філліпс – «Основи. Графічний дизайн 04: Нові основи», ArtHuss, 2020. – 147 с.

17. Методичні вказівки для виконання дипломної роботи бакалаврів для студентів спеціальності 121 – Інженерія програмного забезпечення / Укл. В.В.Любченко. – Одеса: ОНПУ, 2019. – 23 с.

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

```
package diploma;

public  class Main {
    public static void main(String[] args) {
        ObjectArray list1 = new ObjectArray(5);
        ObjectArray list2= new ObjectArray(4);
        ObjectArray list3 = new ObjectArray(3);

        list1.add("Перша відповідь на опитування");
        list1.add("Друга відповідь на опитування");
        list1.add("Третя відповідь на опитування");
        list1.add("Четверта відповідь на опитування");
        list1.add("П'ята відповідь на опитування");

        list1.print();
        list2 = list1;
        list1.equals(list2);
        for (int i = 0; i < list3.size(); i++) {
            list3.add("Нова відповідь " + i);
        }
        list3.print();
        list3.equals(list2);
        list3.add(2, "Так");
        list3.print();
        list3.delete("Так");
        list3.print();

    }
}


***Class ObjectArray***

        import java.util.Arrays;
        import java.util.Collection;
        import java.util.Objects;

public class ObjectArray implements Functional{
    private int count;
    private Object[] arrayListCustom;

    public ObjectArray(int length) {
        arrayListCustom = new Object[length];
        this.count = 0;
    }

    @Override
    public boolean add(Object object) {
        checkAndChangeArraySize();
        arrayListCustom[count] = object;
        count++;
        return false;
    }

    @Override
    public boolean add(int index, Object object) {
        checkAndChangeArraySize();
        if (index > count) {
            add(object);
        }
```

```
        if (index <= count) {
            for (int i = arrayListCustom.length - 1; i > index; i--) {
                arrayListCustom[i] = arrayListCustom[i - 1];
            }
            arrayListCustom[index] = object;
        }
        count++;
        return false;
    }

    @Override
    public boolean delete(Object object) {
        for (int i = 0; i < arrayListCustom.length - 1; i++) {
            if (Objects.equals(arrayListCustom[i], object)) {
                delete(i + 1);

            }
        }
        return false;
    }

    @Override
    public Object get(int index) {
        for (int i = index; i < arrayListCustom.length-1; i++) {
            arrayListCustom[i - 1] = arrayListCustom[i];
            arrayListCustom[i] = null;
        }
        count--;
        return null;
    }

    @Override
    public boolean contain(Object object) {
        for (int i = 0; i < count; i++) {
            if (object.equals(arrayListCustom[i])) {
                return true;
            }
        }
        return false;
    }

    @Override
    public boolean equals(Collection string) {
        if (string == this) {
            return true;
        }
        if (this.getClass() != string.getClass()) {
            return false;
        }
        ObjectArray customArrayList = (ObjectArray) string;
        return this.count == customArrayList.count&&
Arrays.equals(this.arrayListCustom, customArrayList.arrayListCustom);

    }


    @Override
    public boolean clear() {
        arrayListCustom = new ObjectArray[10];
        this.count = 0;
        return true;
    }
```

```
    private void checkAndChangeArraySize() {
        if (arrayListCustom.length < count + 1) {
            int doubledSize = (arrayListCustom.length * 2);
            String[] array1 = new String[doubledSize];
            System.arraycopy(arrayListCustom, 0, array1, 0,
                             arrayListCustom.length);
            arrayListCustom = array1;
        }

    }

    @Override
    public void print() {
        for (Object obj : arrayListCustom) {

            System.out.print(obj + " ");
        }
    }

    @Override
    public int size() {
        return count;
    }
}
```

Interface Functional

```
    import java.util.Collection;

    public interface Functional {

    public boolean add(Object object);

    public boolean add(int index, Object object);

    public boolean delete(Object object);

    public Object get(int index);

    public boolean contain(Object object);

    public boolean equals(Collection string);

    public boolean clear();

    public void print();

    public int size();
}
```

```
package diploma;
import org.json.JSONObject;

public class Construct
  {                             public String construct_type;
public float quadrate_a;
                                public float quadrate_b;
                                public int count_ground;
```

```
    public ConstructDesign (String constructType, float quadrateA, float
quadrateB, int countGround){

    this.construct_type = constructType;
    this.quadrate_a = quadrateA; this.quadrate_b = quadrateB; this.count_ground
=countGround;
    }

    public JSONObject seJson(){
    JSONObject constructdesign = new JSONObject();
    constructdesign.put("construct_type",                    this.construct_type);
constructdesign.put("quadrateA",                                 this.quadrate_a);
constructdesign.put("quadrateB",                                 this.quadrate_b);
constructdesign.put("count_ground", this.count_ground);
    return constructdesign;
    }

    static public ConstructDesign deJson(JSONObject o){ String construct_type =
o.getString("construct_type");
    float quadrateA = o.getFloat("quadrateA");
    float quadrateB = o.getFloat("quadrateB");
    int count_ground = o.getInt("count_ground");

    return new ConstructDesign (construct_type, quadrateA, quadrateB,
count_ground);
    }

    @Override
    public boolean equals(Object o){
    if (this == o) return true;
    if(o == null) return false;
    if (!(o instanceof ConstructDesign)) return false;
    ConstructDesign other = (ConstructDesign) o;
    return this.construct_type.equals(other.construct_type)
    && this.quadrate_a == other.quadrate_b
    && this.count_ground == other.count_ground;
    }
    }


    package diploma;
    import org.json.JSONObject;

    public class ConstructAddress {
    public String city;
    public String str;
    public String phone;
    public ConstructAddress(String city, String str, String  phone){
    this.city = city;
    this.str = str;
    this.phone = phone;
    }


    public String getData(){
    return String.format("%s, %s, %s", city, str,  phone);
    }
    public JSONObject seJson(){
    JSONObject        construct_address        =        new        JSONObject();
construct_address.put("city", this.city); construct_address.put("str", this.str);
    construct_address.put("phone", this.phone);
    return construct_address;
    }
```

```
    static public ConstructAddress deJson(JSONObject o){
    String city = o.getString("city");
    String str = o.getString("str");
    String phone = o.getString("phone");

    return new ConstructAddress(city, str, phone);
    }

    @Override
    public boolean equals(Object o){
    if (this == o)
    return true;
    if(o == null)
    return false;
    if (!(o instanceof ConstructAddress)) return false;

    ConstructAddress other = (ConstructAddress) o;
    return this.phone.equals(other.phone) &&
    this.str.equals(other.str)&& this.city.equals(other.city);
    }
    }


    package diploma;
    import org.json.JSONObject;
    import java.util.Objects;

    public class Respondent extends Person {
                                    private   ConstructAddress   construct_address;
private ConstructDesign construct_design;
                                    private Long id;
    public   Respondent(String   surname,   String   name,   PhoneNumber   phone,
ConstructAddress adds, ConstructDesign design) {
    super(surname,     name,     phone);     this.construct_adds     =     adds;
this.construct_design = design; this.id = null;
    }


    public ConstructAdds getAdds(){ return this.construct_adds;
    }


    public ConstructDesign getDesign(){ return this.construct_design;
    }


    public Long getId(){
    return this.id;
    }


    public boolean setId(Long Id){
    if (this.id == null){
    this.id = Id;
    return true;
    }else{
    this.id = Id;
    return false;
    }

    }

    public JSONObject seJson(){
```

```
    JSONObject respondent = new JSONObject();
    respondent.put("surname", this.surname); respondent.put("name", this.name);
    respondent.put("phone", this.getPhone().seJson());
    respondent.put("construct_adds", this.construct_adds.seJson());
    respondent.put("construct_design", this.construct_design.seJson());
    respondent.put("id", this.id);
    return respondent;
    }

    static public Respondent deJson(JSONObject o){ String surname =
o.getString("surname"); String name = o.getString("name");
    PhoneNumber phone = PhoneNumber.deJson((JSONObject) o.get("phone"));
    ConstructAdds constructAdds = ConstructAdds.deJson((JSONObject)
o.get("construct_adds"));
    ConstructDesign constructDesign = ConstructDesign.deJson((JSONObject)
o.get("construct_design"));

    Respondent respondent = new
Respondent(surname,name,phone,constructAdds,constructDesign);
    respondent.setId(o.getLong("id"));
    return respondent;
    }

    @Override
    public boolean equals(Object o){

    if (this == o)
    return true;
    if(o == null)
    return false;
    if (!(o instanceof Respondent))
    return false;

    Respondent other = (Respondent) o;
    return this.id.equals(other.id) && this.getName().equals(other.getName()) &&
     this.getSurName().equals(other.getSurName())&&
    this.getPhone().equals(other.getPhone()) &&
this.construct_adds.equals(other.construct_adds) &&
     this.construct_design.equals(other.construct_design);

    }
    }


    package diploma;
    import org.json.JSONObject;

    public class RespondentAdmin extends Person {
    static String name_of_diploma = "Star";

    public RespondentAdmin(String surname, String name, PhoneNumber phone) {
    super(surname, name, phone);
    }

    public JSONObject seJson(){
    JSONObject admin = new JSONObject(); admin.put("surname", this.surname);
admin.put("name", this.name);
    admin.put("phone", this.getPhone().seJson());
    return admin;
    }

    static public RespondentAdmin deJson(JSONObject o){
    String surname = o.getString("surname");
```

```
    String name = o.getString("name");
    PhoneNumber phone = PhoneNumber.deJson((JSONObject)o.get("phone"));
    return new RespondentAdmin(surname, name, phone);
    }

    @Override
    public boolean equals(Object o){
    if (this == o)
    return true;
    if (o == null)
    return false;
    if (!(o instanceof RespondentAdmin))
    return false;
     RespondentAdmin other = (RespondentAdmin) o;

    return                other.getName().equals(this.getName())                &&
other.getPhone().equals(this.getPhone());
    }
    }


    package diploma;

    import jdk.nashorn.internal.ir.debug.JSONWriter;
    import netscape.javascript.JSObject;
    import org.json.JSONObject;

    public class Convention { public long id;
    public ConstructDesign construct_design;
    public ConstructAdds construct_adds;
    public Respondent respondent;
    public RespondentAdmin admin;
    public long cost;
    public Convention (long id, ConstructDesign construct_design, ConstructAdds
construct_adds, Respondent respondent, RespondentAdmin admin, long cost){
    this.id = id;
    this.construct_design    =    construct_design;    this.construct_adds    =
construct_adds;
    this.respondent = respondent;
    this.admin = admin;
    this.cost = cost;
    }

    public JSONObject seJson(){
    JSONObject convention = new JSONObject(); convention.put("id", this.id);
    convention.put("respondent", this.respondent.seJson());
    convention.put("admin", this.admin.seJson());
    convention.put("cost", this.cost);
    return convention;
    }

    static Convention deJson(JSONObject convention_j){
    long id = convention_j.getLong("id");
    long cost = convention_j.getLong("cost");
    Respondent respondent = Respondent.deJson((JSONObject)
    convention_j.get("respondent"));
    RespondentAdmin       admin       =       RespondentAdmin.deJson((JSONObject)
convention_j.get("admin"));

    return   new   Convention(id,   respondent.getDesign(),   respondent.getAdds(),
respondent, admin, cost);
    }
    }
```

```
package diploma;

import org.json.JSONArray;
import org.json.JSONException; import org.json.JSONObject;
import java.util.*;


public class ConventionSystem {
static final String_NAME OF DIPLOMA = "Diploma";
private ArrayList<Convention> conventions;
private ArrayList<Respondent> respondents;
private ArrayList<RespondentAdmin> admins = new ArrayList<RespondentAdmin>();
public Rate rate;
public ConventiontSystem(){
this.conventions = new ArrayList<Convention>(); this.respondents = new
ArrayList<Respondent>(); this.rate = new Rate();
}

public int regRespondent(String Name, String adds, String phone, String
constructType, String quadrateA, String quadrateB, String grounds){

String[] names = Name.split(" if (names.length <2){return 1;}
");

String[] addses = adds.split(", ");
 if (addses.length != 5){return 2;}
ConstructAdds constructAdds = new ConstructAdds(addses[0], addses[1],
addses[2], addses[3], addses[4]);

if (phone.length() != 13){return 3;}
String phoneCountry = phone.substring(0, 4); String phoneReg =
phone.substring(4, 8); String phoneRespondent = phone.substring(8);

Phone phone = new Phone (phoneCountry, phoneReg, phoneRespondent);

float quadrate_a, quadrate_b;
int count_ground;
try{
quadrate_a = new Float(quadrateA); quadrate_b = new Float(quadrateB);
count_ground = new Integer(grounds);
}catch (NumberFormatException e){ return 4;}

ConstructDesign constructDesign = new ConstructDesign (constructType,
quadrate_a, quadrate_b, count_ground);

Respondent respondent = new Respondent(names[0], phone, constructAdds,
constructDesign);
this.respondents.add(respondent);
return 0;
}

public int regAdmin(String Name, String phone){ String[] names = Name.split("
");
if (names.length <2){return 1;}

if (phone.length() != 13){return 2;}
String phoneCountry = phone.substring(0, 4); String phoneReg =
phone.substring(4, 8);
String phoneRespondent = phone.substring(8);

Phone phone = new Phone (phoneCountry, phoneReg, phoneRespondent);

RespondentAdmin admin = new RespondentAdmin(names[0], phone);
```

```
    this.admins.add(admin);
    return 0;
    }


    public ArrayList<String[]> conventionList(){ ArrayList<String[]> reply = new
ArrayList<String[]>(); String[] t;

    for (Convention c:conventions) {
    t    =    new    String[]{String.valueOf(c.id),    c.construct_adds.getData(),
String.valueOf(c.construct_design.quadrate_b),
    String.valueOf(c.construct_design.count_ground), String.valueOf(c.cost)};

    reply.add(t);
    }
    return reply;
    }


    public float averageQuadrate(){
    float square = 0;
    for(Convention convention: this.conventions){

    quadrate += convention.construct_design.quadrate_b;
    }
    if (this.conventions.size() == 0) return 0;
     return quadrate / this.conventions.size();
    }



    public   int   regConvention   (String   respondentName,   String   adminName){
Respondent respondent = null;
    RespondentAdmin admin = null;
    for (Respondent c: this.respondents) {     // ------
    respondent find
    if (c.getName().equals(respondentName)) respondent = c;
    }
    if(respondent == null) return 1;

    for (RespondentAdmin m: this.admins){      // ------
    admin find
    if(m.getName().equals(adminName)) admin = m;
    }
    if(admin == null) return 2;

    if (respondent.getId() != null) return 3;  // ------
    respondent have id before

    long new_id = this.conventions.size()+1;
    String construct_type = respondent.getDesign().construct_type;

    float quadr = respondent.getDesign().quadrate_b;
    int grd = respondent.getDesign().count_ground;
    long price = this.rate.getPrice(construct_type, quadr, grd);

    Convention   convention   =   new   Convention(new_id,   respondent.getDesign(),
respondent.getAdds(), respondent, admin, price);

    this.conventions.add(convention); respondent.setId(id);

    return 0;
    }
```

```
public int getCountRespondents(){
return this.respondents.size();
}
public int getCountAdmins(){
return this.admins.size();
}
public int getCountConventions(){
return this.conventions.size();
}
public String[] getAdminsName(){
int ln = this.admins.size();
String[] names = new String[ln];

int i = 0;
for (RespondentAdmin c: this.admins) {
names[i] = c.getName();
i++;
}
return names;
}

public String[] getRespondentsName(){
int ln = this.respondents.size();
String[] t_names = new String[ln];

int i = 0;
for (Respondent c:this.respondents) {
if (c.getId() == null){ t_names[i]=c.getName();
i++;
}
}

String[] names = new String[i];
if (i > 0) System.arraycopy(t_names, 0, names, 0, i);

return names;
}



package diploma;

abstract class Person {
public String surname; public String name;
private PhoneNumber phone;

public Person(String surname, String name, PhoneNumber phone){

this.surname = surname;
this.name = name;
this.phone = phone;
}


public String getName(){
return this.surname + " " + this.name;
}


public PhoneNumber getPhone()
{
 return this.phone;
}
```

```
}


package interface;

import javax.swing.*; import java.awt.*;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener; import java.awt.event.WindowEvent;

public class RespondentReg extends JFrame {

JTextField nameField, addsField, phoneField, quadrateAField, quadrateBField,
groundField;
JComboBox<String> constructCombo; private Menu Window;
RespondentReg (Menu Window){
super("Реєстрація нового відповідача");
this.Window = Window;


Tool tool = Tool.getDefaultTool();
this.setSize(700,330);
this.setRes(false);
Dimension size_screen = tool.getSizeScreen ();
this.setLocation((size_screen.width/2) - 380,
(size_screen.height/2)-160);
this.setImage(new Image ("img.png").getImage());

JLabel Label = new JLabel();
Label.setText("ПІБ:");
Label.setFont(new Font("Times new roman", Font.PLAIN, 25));
Label.setBounds(10, 20, 40, 10);
this.add(Label);

Field = new JTextField();
Field.setFont(new Font("Times new roman", Font.PLAIN, 25));
Field.setBounds(120, 16, 240, 20);
this.add(Field);


JLabel addsLabel = new JLabel();
addsLabel.setText("Адреса:");
addsLabel.setFont(new Font("Times new roman", Font.PLAIN, 25));
addsLabel.setBounds(10, 70, 80, 10);
this.add(addsLabel);

addsField = new JTextField();
addsField.setFont(new Font("Times new roman", Font.PLAIN, 25));
addsField.setBounds(120, 66, 240, 20);
this.add(addsField);


JLabel phoneL = new JLabel();
phoneL.setText("Номер телефону:");
phoneL.setFont(new Font("Times new roman", Font.PLAIN, 25));
phoneL.setBounds(20, 130, 105, 20);
this.add(phoneL);

phoneF = new JTextField();
phoneF.setFont(new Font("Times new roman", Font.PLAIN, 25));
phoneF.setBounds(120, 116, 240, 20);
this.add(phoneF);

constructCombo = new JComboBox<String>();
```

```
    constructCombo.setFont(new Font("Times new roman", Font.PLAIN, 25));
    constructCombo.setBounds(500, 16, 150, 20);
    constructCombo.setMaxCount(5);
    this.add(constructCombo);

    JLabel quadrateLabel = new JLabel();
    quadrateLabel.setText("Розмір вікна:");
    quadrateLabel.setFont(new   Font("Times   new   roman",   Font.PLAIN,   25));
quadrateLabel.setBounds(400, 70, 165, 15);
    this.add(quadrateLabel);

    quadrateAField = new JTextField();
    quadrateAField.setFont(new   Font("Times   new   roman",   Font.PLAIN,   25));
quadrateAField.setBounds(500, 66, 160, 20);
    this.add(quadrateAField);

    quadrateBField = new JTextField();
    quadrateBField.setFont(new Font("Times new roman", Font.PLAIN, 25));
    quadrateBField.setBounds(500, 116, 160, 20);
    this.add(quadrateBField);


    JLabel groundLabel = new JLabel();
    groundLabel.setText("Кількість стр:");
    groundLabel.setFont(new Font("Times new roman", Font.PLAIN, 25));
    groundLabel.setBounds(400, 170, 200, 10);
    this.add(groundLabel);

    groundField = new JTextField();
    groundField.setFont(new    Font("Times    new    roman",    Font.PLAIN,    25));
groundField.setBounds(620, 156, 120, 20);
    this.add(groundField);

    JButton   regBut   =   new   JButton();   regBut.setText("Реєстрація   нового
відповідача");
    regBut.setFont(new     Font("Times     new     roman",     Font.PLAIN,     18));
regBut.setBounds(315, 215, 140, 40);
    regBut.addActionListener((e)->{ this.register();});
    this.add(regBut);

    nameField.addFocusListener(new FocusListener() {
    @Override
    public void focusG (FocusE e) {
    if (Field.getText().equals("Імя та Прізвище відповідача")) {
     Field.setText("");
    Field.setForeground(Color.BLACK);
    }
    }
    @Override
    public void focusL (FocusE e) {
    if (Field.getText().isEmpty()) {
    Field.setForeground(Color.GRAY);
    Field.setText("Імя та Прізвище відповідача");
    }
    }
    });

    addsField.addFocusListener(new FocusListener() { @Override
    public void focusG (FocusE e)
    {
    if (addsField.getText().equals("Назва Країни, назва міста, назва вулиці та
    номер телефону"))
    {
```

```
    addsField.setText("");
    addsField.setForeground(Color.BLACK);

    }}


    @Override
    public void focusL(FocusE e) {
    if (addsField.getText().isEmpty())
    {
    addsField.setForeground(Color.GRAY);
    addsField.setText("Назва Країни, назва міста, назва вулиці та номер
телефону");
    }
    }
    });

    phoneF.addFocusListener(new FocusListener() { @Override
    public void focusG (FocusE e) {
    if (phoneF.getText().equals("+380*********")) {
    phoneF.setText("");
    phoneF.setForeground(Color.BLACK);
    }
    }
    @Override
    public void focusL (FocusE e) {
    if (phoneF.getText().isEmpty())
    {
    System.out.println("Error");
    phoneF.setForeground(Color.GRAY); phoneF.setText("+380*********");
    }
    }
    });


    private void reg (){
    String name = Field.getText();
    String adds = addsField.getText();
     String phone = phoneF.getText();
    String construct_type;
    String quadrate_a = quadrateAField.getText();
    String quadrate_b = quadrateBField.getText();
    String count_g = groundField.getText();

    try{
    construct_type = constructCombo.getSelectedItem().toString();
    }catch (Exception e){
    JOptionPane.showMessageDialog(this,   "Не   вдалося   зареєструвати   нового
відповідача", JOptionPane.ERROR_MESSAGE);
    return;
    }

    int   res   =   Window.rsystem.regRespondent(name,adds,phone,construct_type,
quadrate_a, quadrate_b, count_g);


    if (result == 0){

    JOptionPane.showMessageDialog(this, "Зареєстровано нового     відповідача",
JOptionPane.INFORMATION_MESSAGE);
    Field.setText("");
    addsField.setText("");
```

```
    phoneF.setText("");
    quadrateAField.setText("");                    quadrateBField.setText("");
groundField.setText("");

    Window.updateCounters(); this.setVisible(false); Window.setVisible(true);
    }
    else
    if (result == 1){
    JOptionPane.showMessageDialog(this,   "Не   вдалося   зареєструвати   нового
відповідача", JOptionPane.ERROR_MESSAGE);
    }
    else
    if (result == 2){
    JOptionPane.showMessageDialog(this,   "Не   вдалося   зареєструвати   нового
відповідача, перевірте правильність введення адреси", JOptionPane.ERROR_MESSAGE);
    }else if (result == 3){
    JOptionPane.showMessageDialog(this,   "Не   вдалося   зареєструвати   нового
відповідача,    перевірте    правильність    введення    номеру    телефону",
JOptionPane.ERROR_MESSAGE);
    }
    else
    if (result == 4){
    JOptionPane.showMessageDialog(this,   "   Не   вдалося   зареєструвати   нового
відповідача, невірне введення, JOptionPane.ERROR_MESSAGE);
    }
    }
```