МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Інформаційна система оцінювання знань в області тестування програмного забезпечення»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студентка групи ІТ.мз-91с Іванченко Ірина Миколаївна

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2020 р.

Науковий керівник _____ к. т. н. Бойко О.В
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2020

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____
В. В. Шендрик
«___» _____ 2020 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Іванченко Ірина Миколаївна
(прізвище, ім'я, по батькові)

1 Тема проекту *Інформаційна система оцінювання знань в області тестування програмного забезпечення / Information System for Assessing Knowledge in the Field of Software Testing*

затверджена наказом по університету від «16» листопада 2020 р. №1773-III

2 Термін здачі студентом закінченого проекту «_07__» ___грудня__ 2020 р.
3 Вхідні дані до проекту __Загальна інформація про тестування, Інформаційні системи-аналоги, Засоби реалізації веб-додатків_____
_____

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) ____Аналіз предметної області, Постановка задачі та методи дослідження, Моделювання інформаційної системи, Реалізація інформаційної системи_____
_____

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) ____Інформаційна система, База Даних, Презентація_____
_____

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускного проекту	Термін виконання етапів проекту	Примітка
1.	Ідентифікація ідеї проекту	01.09.2020 - 04.09.2020	
2.	Дослідження предметної області	07.09.2020 - 18.09.2020	
3.	Вибір методів та засобів розробки	21.09.2020 - 02.10.2020	
4.	Моделювання роботи інформаційної системи	05.10.2020 - 09.10.2020	
5.	Розробка шаблону сайту	12.10.2020 - 16.10.2020	
6.	Розробка Бази Даних	19.10.2020 - 30.10.2020	
7.	Розробка підсистеми тестування	02.11.2020 - 06.11.2020	
8.	Розробка особистих кабінетів роботодавця та претендента	09.11.2020 - 13.11.2020	
9.	Перенесення сайту на сервер	16.11.2020 - 20.11.2020	
10.	Тестування системи	23.11.2020 - 27.11.2020	

Магістрант _____ Іванченко І.М.

Керівник роботи _____ к. т. н. Бойко О.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Інформаційна система оцінювання знань в області тестування програмного забезпечення».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 26 найменувань, додатків. Загальний обсяг роботи – 128 сторінок, у тому числі 61 сторінка основного тексту, 3 сторінки списку використаних джерел, 64 сторінки додатків.

Кваліфікаційну роботу магістра присвячено розробці інформаційної системи оцінювання знань в області тестування програмного забезпечення.

В роботі проведено пошук теоретичного характеру, що виконувався з метою визначення актуальності, необхідності та технічної можливості створення нової інформаційної системи. Проведено аналіз предметної області, шляхом виявлення проектної ідеї за наперед заданими критеріями проекту. Проведено порівняльний аналіз інформаційних систем-аналогів. Викладено загальний опис роботи майбутньої інформаційної системи. Обрано методи та засоби для розробки даної інформаційної системи.

У роботі виконано планування змісту та структури робіт, поетапне моделювання процесу тестування, на основі розроблених моделей запропоновано відповідну інформаційну систему оцінювання знань, представлено архітектуру даної інформаційної системи, описано функціональні модулі, приведені приклади роботи.

Результатом проведеної роботи є інформаційна система оцінювання знань в області тестування програмного забезпечення.

Практичне значення роботи полягає у розробленні інформаційної системи, яка надає користувачеві – роботодавцю можливість створювати тести, користувачеві – претенденту пройти тестування, всім бажаючим спілкуватися через чат-повідомлення в реальному часі.

Ключові слова: інформаційна система, web-додаток, тестування, html5, JavaScript, mysql, Vue.js, laravel.

ЗМІСТ

ВСТУП

У 2020 році пандемія Covid19 спричинила різкі зміни на ринку праці в багатьох сферах діяльності. По-перше, сьогодні майже 400 тисяч людей шукають роботу. Тож люди часом мусять опановувати нові спеціальності. По-друге, бізнес переходить на віддалену онлайн роботу, що викликає збільшення попиту не тільки на фахівців з програмування та розробки програмного забезпечення, а і тестувальників (QA engineer) [1].

Кількість позицій у галузі тестування комп'ютерних технологій сьогодні найбільша за останні три роки. При чому нова професія тестувальник програмного забезпечення – це професія, на яку можна відносно швидко перекваліфікуватися.

Звичайно, будь-яке працевлаштування передбачає перевірку знань. Перевірка кваліфікації – це спосіб роботодавців оцінити здібності кандидата за допомогою різноманітних форматів тестування. Наприклад, тести на перевірку здатності виконувати мануальне тестування та реагувати на ситуації на роботі. Це включає вирішення проблем, встановлення пріоритетів та числові навички, серед іншого. Кандидати можуть вирішити практичні проблеми чи завдання наближені до них, з якими вони можуть зіткнутися в компанії [2].

Справжня вартість невдалого найму коштує роботодавцям великі кошти, тому що, коли найманий працівник не справляється зі своєю роботою, хороші працівники «згорають», компенсуючи це.

На жаль, поганих працівників не завжди легко відразу помітити. На першій співбесіді HR можуть неякісно діагностувати претендентів, а коли це буде помічено, роботодавець вже витратить дорогоцінний час і гроші, як на невдалий найм, так і на пошук наступного претендента.

Тому на сьогодні є актуальним полегшення процесу відбору майбутніх співробітників шляхом автоматизованого тестування знань, адже це спростить

і пришвидчить процедуру знаходження потрібних співробітників з великої кількості претендентів.

Щоб збільшити відсоток вдалого найму необхідна інформаційна система оцінювання знань в області тестування програмного забезпечення. Ця система спростить відбір найкраще підготовлених початківеців-тестувальників, які не тільки мають необхідні знання, але і вже визначилися, в якій області тестування вони хочуть розвиватися (тестування desktop-додатків, веб-тестування, мобільне тестування, тестування ігор або інше).

Розроблена інформаційна система будє представляти собою платформу для роботодавців та кандидатів.

Головна ідея сайту – перевірити та оцінити знання претендентів на вакантні посади в області тестування програмного забезпечення.

Допуск тільки підготовлених фахівців до наступних етапів співбесід збільшить швидкість знаходження висококваліфікованих фахівців, а головне – зекономить компанії багато часу та коштів.

Метою даного дипломного проекту є створення інформаційної системи оцінювання знань в області тестування програмного забезпечення.

Основними задачами дослідження при створенні даної інформаційної системи є:

- аналіз існуючих аналогів;
- формування загальних вимог до інформаційної системи;
- побудова моделі роботи інформаційної системи;
- обрання методів та засобів реалізації інформаційної системи;
- розробка бази даних;
- виконання програмної розробки та тестування розробленої інформаційної системи.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Огляд останніх досліджень і публікацій

Управління людськими ресурсами - це термін, що використовується для опису комплексу завдань, спрямованих на ефективне управління працівниками організації, широко відомий як людські ресурси або людський капітал. Спеціалісти з управління персоналом здійснюють нагляд за управлінням людьми в організації, що включає компенсації, пільги, навчання та розвиток, персонал, стратегічне управління персоналом та інші функції [3].

Працівники з персоналу структурують кадрові програми для набору та утримання найкращих співробітників, роблячи компанію конкурентоспроможною з точки зору її привабливості для потенційних кандидатів, щоб вони вирішили прийняти посаду та залишатися працювати у роботодавця. У сучасному конкурентному середовищі управління людським капіталом надзвичайно важливо для збереження життєздатності на світовому ринку. Як результат, HR відіграє ключову роль у світі - адже люди справді єдине, що відрізняє один бізнес від іншого [4]. Організації можуть повторювати процеси, матеріали та структури інших успішних організацій, але лише талант організації робить її унікальною та відрізняє від усіх її конкурентів.

Робота в HR потребує особливий тип людини, якій зручно вирішувати проблеми, вдосконалювати процеси, вимірювати досягнення, розробляти системи, мати справу з культурою організації та, головне, працювати з людьми.

Спеціаліст, менеджер чи директор з персоналу виконує найрізноманітніші ролі в організаціях. Залежно від розміру організації, ці робочі місця з персоналу можуть нести відповідальність. У більших організаціях кадровий спеціаліст, менеджер та директор чітко визначили окремі ролі в управлінні персоналом.

Ці ролі поступово приносять більше повноважень та відповідальності в руках менеджера, потім директора і, врешті-решт, віце-президента, який може керувати декількома департаментами, включаючи адміністрацію, компенсації та навчання та розвиток співробітників.

Директори з персоналу, а іноді і менеджери з персоналу, можуть очолювати декілька різних підрозділів, кожен з яких очолює функціональний або спеціалізований персонал, такий як менеджер з навчання, менеджер з питань компенсацій або менеджер з найму [5].

Працівники відділу кадрів є адвокатами як компанії, так і людей, які працюють у компанії. Отже, хороший HR-фахівець здійснює постійний баланс, щоб успішно задовольнити обидві потреби.

Перевірка кваліфікації – це спосіб роботодавців оцінити здібності кандидата за допомогою різноманітних форматів тестування. Тести на здатність перевірять вашу здатність виконувати завдання та реагувати на ситуації на роботі. Це включає вирішення проблем, встановлення пріоритетів та числові навички, серед іншого [6]. Психометричні тести можна вибрати з декількома варіантами, й існує лише одна правильна відповідь, тоді ваш бал ставиться, а ваш рівень порівнюється з іншими кандидатами, які пройшли той самий тест, що і ви. Існують безкоштовні тести в Інтернеті, які ви можете взяти, щоб визначити, на які питання чекати, коли складаєте іспит.

Розглянемо перевірку класифікації с точки зору підбору фахівців у IT сфері, на сам перед в області тестування програмного забезпечення.

Оцінка навичок програмістів стала важливим кроком у будь-якому процесі технічного набору персоналу. Під час збору інформації було виявлено, що головною проблемою, з якою стикаються 60% рекрутерів під час найму розробників – знайти достатньо кваліфікованих кандидатів. Дійсно, більшість непрацездатних працівників у підборі технічних працівників зводиться до браку технічних навичок.

Різноманітні здібності можуть бути притаманні гарному спеціалісту, але технічні навички - це те, що можна оцінити найбільш точно. Розглянемо декілька способів перевірити розробників [7] перед наймом:

− Портфоліо - найкращий спосіб для розробників продемонструвати свої навички кодування за допомогою особистих проектів та унікальний шанс справити тривале враження. Вони бувають різних форм і розмірів, починаючи від окремих веб-сторінок, для тих, хто шукає першу роль молодшого класу, до складних програмних проектів для старших та керівних заявників.

− Інший вид – швидко перевірити репутацію розробників у Stack Overflow, а також їх найпопулярніші відповіді. Розробники можуть задавати та відповідати на запитання щодо будь-якого відношення до програмування, а активні учасники отримують бали та значки за свої відповіді та внески [6]. Сайт має багаті знання, і будь-який розробник буде користуватися ним щодня. Він ідеально підходить для вимірювання рівня знань кандидата та залучення його до спільноти розробників.

− Тестування - один із найефективніших способів перевірити розробників перед наймом.

Вони забезпечують перевірену модель виявлення та найму досвідчених розробників. Кандидати можуть вирішити практичні проблеми чи завдання наближені до них, з якими вони можуть зіткнутися в компанії. Наприклад, виявлення помилки в дефектному фрагменті коду або правильна синхронізація багато потокової програми, що дозволяє рекрутерам об'єктивно та ефективно вимірювати навички кандидата.

1.2. Аналіз програмних продуктів

На даний момент часу існує безліч додатків чи сайтів для перевірки чи тестування кандидатів за різними напрямками. Розглянемо декілька з них та дізнаємося ті чи інші переваги та недоліки даних ресурсів.

1.2.1. CodinGame

Технічні рекрутери до менеджерів з персоналу можуть налаштовувати тести програмування на таких платформах, як CodinGame , вибираючи тестувати заявників на одній конкретній мові програмування або на декількох технологіях [8].

Код кандидатів автоматично аналізується, а рекрутери отримують показники ефективності, починаючи від володіння мовою і закінчуючи дизайном коду, читабельністю та надійністю. Після завершення випуск тесту можна завантажити та поділитися для легкого порівняння та включення до списку кандидатів.

Окрім оптимізації процесу найму, ці тести також забезпечують покращений досвід роботи з кандидатами, порівняно з нетехнічними дискусіями, які занадто часто проводяться під час першого співбесіди. Тести програмування - це спосіб для рекрутерів показати розробникам, що вони технічно безпечні.

Проблеми з кодуванням заощаджують вербувальникам багато часу, забезпечуючи наймитів з технічної сторони. Тести технічного програмування в Інтернеті корисно замінюють тести на папері чи дошці, даючи розробникам можливість продемонструвати свої вміння.

1.2.2. Interview Mocha

Interview Mocha - це хмарне рішення для онлайн-оцінки, яке дозволяє компаніям відбирати та фільтрувати кандидатів шляхом попереднього працевлаштування та перевірки кваліфікації працівників [9]. Платформа охоплює широкий спектр навичок, включаючи SAP, бухгалтерський облік, технології здібностей, IT, ділові навички та інші попередньо побудовані та спеціальні тести.

Доступно понад п’ятнадцять сотень готових оцінок навичок. Усі вони були підтверджені експертами з предметних питань у всьому світі. Завдяки цьому інтерв’ю Mocha може допомогти вашій компанії швидше знайти кандидатів, які підходять до роботи, і точно їх оцінити.

Процес скринінгу можна розбити на три простих кроки: виберіть бажаний тест, запросіть кандидатів одним клацанням миші та отримайте результати, як тільки вони закінчать. Ви можете скоротити час, який зазвичай витрачаєте на співбесіду з нерелевантними кандидатами. Відомо, що готові та індивідуальні тести Mocha скоротили час інтерв'ю на 40% і більше.

Крім того, легко зробити і власний тест. Щоб додати власні запитання, вам не потрібні навички кодування. Ви можете вибрати різні типи запитань, такі як MCQ, Coding або LogicBox. Оскільки платформа заснована на Інтернеті, вам не доведеться турбуватися про вільні папери. Вся необхідна інформація буде знаходитись безпосередньо в інтерв'ю Mocha.

Головний недолік даної системи – відсутність безкоштовної версії чи тестового періоду. На мою думку, кожному користувачу потрібен певний час на ознайомлення з додатком та вирішити, підходить система чи ні.

Крім того, ціна початкового плану складає 150 доларів на місяць (щорічно сплачується) або 1800 доларів на рік [10]. У даний пакет послуг входить:

- 2 користувачі;
- 300 спроб тесту на рік;
- Жодних спеціальних тестів.

Інші пакети надають більше можливостей, але за вищу ціну, що не завжди підходить да компаній середнього розміру.

Інтерв'ю Mocha – це необхідний інструмент для кожної HR-команди, але з деякими обмеженнями та певної складності використання.

1.2.3. Assessment Day

З постійним зростанням популярності психометричного тестування при прийомі на роботу роль даних інформаційних систем стає все більш важливою. Зусилля, для досягнення успіху, зосереджені на наданні порад та експертних матеріалів для тестування. Дана система націлена на подальше підвищення прозорості та розуміння того, що може бути складним процесом оцінки.

На даному веб-сайті є окрема область, яка буде присвячена вашій організації та вашим студентам. Усі питання та відповіді на практичних тестах написані психометристами з досвідом роботи у великих міжнародних видавцях тестів, таких як SHL, Kenexa, Talent Q та TalentLens [11].

Студенти реєструються за допомогою електронної адреси університету. Після реєстрації їм надсилається електронне повідомлення, яке включає посилання для підтвердження облікового запису та пароль для входу. Кандидати не потребують участі співробітників університету, оскільки всі адміністратори, необхідні для реєстрації студентів, обробляються автоматично.

Вартість буде змінюватися залежно від кількості та типу тестів, до яких ви хотіли б мати доступ ваші студенти. Тести мають ексклюзивний зміст - ті самі запитання не надаються на головному веб-сайті.

1.2.4. Google Forms

Послуга Google Formsзазнала декількох оновлень за ці роки [12]. Функції включають, але не обмежуючись цим, пошук у меню, перемішування питань для рандомізованого замовлення, обмеження кількості відповідей один раз на людину, коротші URL-адреси, власні теми, автоматичне формування пропозицій відповідей під час створення форм та "Завантажити файл "опція для користувачів, які відповідають на запитання, за якими вони мають ділитися вмістом або файлами зі свого комп’ютера чи Google Drive. Функція завантаження доступна лише через Google Workspace.

У жовтні 2014 року Google було представлено доповнення для Google Forms, які дозволяють стороннім розробникам додавати нові функції до опитувань. У 2017 року Google було оновлено Форми, щоб додати кілька нових функцій. "Інтелектуальна перевірка відповіді" здатна виявляти введення тексту в полях форми, щоб ідентифікувати написане і попросити користувача виправити інформацію, якщо неправильно введено.

Залежно від налаштувань спільного використання файлів на Google Диску, користувачі можуть вимагати завантаження файлів від осіб, які не

мають варіантів відповідей у таблиці. У налаштуваннях користувачі можуть вносити зміни, що стосуються всіх нових форм, наприклад, завжди збирати адреси електронної пошти.

Google Forms містить усі функції для спільної роботи та спільного використання, знайдені в Документах, Таблицях та Презентаціях.

Створимо порівняльну характеристику всіх аналогів у табл.1.1

Таблиця 1.1 – Переваги та недоліки аналогів

№	Критерії	CodinGame	Interview Mocha	Assessment Day	Google Forms
1	Наявність реклами	+	+	+	-
2	Поділення на категорії	+	+	+	-
3	Доступність	+	-	-	+
4	Зручність використання	+	+	-	+
5	Зрозумілий інтерфейс	+	+	+	+
6	Онлайн доступ	+	+	+	+
7	Перегляд статистики	-	+	-	+

Виходячи з вище зазначеного помітно, що необхідно створити інформаційну систему оцінювання знань в області тестування програмного забезпечення, що буде містити в собі переваги існуючих систем-аналогів, та долаючи їх недоліки. Додатковими вимогами є: відсутність реклами, поділ на категорії, доступність усім бажаючим. Претенденти матимуть можливість проходити тестування, а HR-менеджери – створювати свої тести за допомогою конструктора, робити публікації на сторінці, збирати статистику, складати рейтинги оцінок кандидатів за результатами перевірочних тестувань.

2. ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1. Мета та задачі дослідження

Мета роботи – розробка інформаційної системи для створення тестових завдань, проведення тестувань та формування списків висококваліфікованих фахівців за результатам цих тестів.

Основними задачами при створенні інформаційної системи оцінювання знань в області тестування програмного забезпечення є:

- дослідження предметної області;
- формування технічного завдання;
- вибір методів розробки;
- тестування та перенесення сайту на хостинг;
- розробка самої інформаційного системи;
- моделювання роботи системи.

Інформаційна система оцінювання знань – це он-лайн платформа для роботодавців та кандидатів, що знаходяться у пошуку роботи. При реєстрації в системі користувач матиме можливість обрати тип акаунта. HR-менеджер матиме можливість створювати тести за допомого конструктора, робити публікації на сторінці та збирати статистику й результати з даних тестів.

Головна сторінка сайту – це сторінка з короткою інформацією про дану інформаційну систему. Кожен авторизований користувач може виконати підписку на оновлення іншого користувача. На сторінці новин відображаються публікації користувачів, на яких оформлена підписка. Також присутнє оповіщення про нові тестування. Функціонал тестування змінюється в залежності від типу акаунту. Кандидат може перегляди пройдені тести та тести, що були додані до бажаних для проходження в майбутньому. Роботодавець може створювати тести. На сторінці редагування тесту можна редагувати як запитання тесту, так і час його проведення чи іншу інформацію

про тест. На сторінці з інформацією про користувача є можливість перегляду та редагування власних постів та створення нових.

Перелік головних функцій даної інформаційної системи:

1. реєстрація як «Кандидат» чи «HR»;
2. авторизація на сайті;
3. налаштування особистого кабінету;
4. оформлення підписки на оновлення іншого користувача;
5. перегляд оновлень та постів;
6. створення постів;
7. перегляд та проходження тестування;
8. створення тесту за допомогою конструктора;
9. налаштування створеного тесту;
10. перегляд історії пройдених тестів та результатів;
11. додавання тестів до «Бажаних» для проходження їх в подальшому;
12. листування з іншими користувачами через чат-повідомлення в реальному часі.

2.2. Вибір засобів реалізації

Розробка інформаційної системи поділяється на розробку візуальної частини, функціональна частина. Для сучасної front-end розробки використовують: HTML, CSS, JavaScript, Vue.js.

Vue.js на даний момент розвитку, прогресивний фреймворк для JavaScript, який використовується для побудови звичних веб-інтерфейсів та односторінкових додатків. Не тільки для веб-інтерфейсів, Vue.js також використовується як для розробки настільних ПК, так і для мобільних додатків з фреймворком Electron [13]. Розширення HTML і база JS швидко зробили Vue улюбленим інтерфейсним інструментом, про що свідчать прийняття таких гігантів, як Adobe, Behance, Alibaba, Gitlab та Xiaomi.

Назва фреймворку - Vue - на англійській мові таке саме фонетичне, як і view. Не дивно, що воно відповідає традиційній архітектурі Model-View-Controller (MVC). Простіше кажучи - це інтерфейс користувача програми / веб-сайту, а основна бібліотека Vue.js за замовчуванням фокусує шар представлення. Не дивлячи на це, MVC не означає, що даний фрейворк не можна використовувати з іншим архітектурним підходом, як відома архітектура на основі компонентів (CBA), що використовується в React.

Головні переваги та недоліки даного обраного фреймворку розглянуто в таблиці 2.1.

Таблиця 2.1 – Переваги та недоліки Vue.js

№	Переваги	Недоліки
1	Не великий розмір проекту при релізі	Мовний бар'єр
2	Віртуальний рендеринг і продуктивність DOM	Складність реактивності
3	Реактивне двостороннє прив'язка даних	Відсутність досвідчених розробників
4	Інтеграційні можливості та гнучкість	
5	Стисла документація	

Фреймфорк з відкритим кодом Vue може допомогти додати до існуючої програми інтерактивність. Оскільки Vue заснований на JavaScript, його можна легко інтегрувати в будь-який проект за допомогою JS. Більше того, він сумісний з багатьма внутрішніми технологіями та фреймворками, такими як Laravel, Express, Rails та Django [14].

Навіть враховуючи мінуси Vue.js, його все одно можна використовувати у великих проектах. Більшість питань розглядаються в документації, тож це питання пошуку.

У той же час, для розробки back-end частини буде використовуватися такі технології як PHP, MySQL, Laravel. Це одна з найвидатніших архітектур PHP, яка відповідає моделі Model-View-Controller або моделі MVC. Безкоштовний фреймворк з відкритим кодом був розроблений Тейлором Отуеллом і був запущений ще в 2011 році.

Laravel - одна з найвидатніших фреймворків на сьогоднішній день, і завдяки мові програмування на стороні сервера PHP вона дозволяє розробникам зосередитись на основних засадах, завдяки простим варіантам масштабування та швидкій розробці з кодами, якими легко керувати.

Однією відмінною перевагою, яку розробники отримують від використання Laravel, є той факт, що він включає всі новітні функції PHP. Якщо ви думаєте про те, щоб спробувати Laravel, знайте, що ви зможете використати найкращі речі, які зараз може запропонувати PHP [15]. Такі функції, як простори імен, перекриття, анонімні функції, інтерфейси та коротший синтаксис масиву.

Laravel має власний інтегрований інтерфейс командного рядка. Artisan дозволяє розробникам створювати скелетні коди. Він також контролює систему баз даних, тому розробникам не потрібно виконувати рутинні завдання програмування. Artisan також є чудовим інструментом, коли йдеться про створення, а також підтримку простих файлів MVC з відповідними налаштуваннями.

Говорячи про недоліки даної системи у порівнянні з іншими фреймворками, такими як Ruby on Rails та Django, Laravel має обмежену вбудовану підтримку завдяки своїй невеликій вазі. Цю проблему можна вирішити за допомогою сторонніх інструментів, і ви можете якнайшвидше повернутися до шляху.

Загалом у платформ PHP є кілька питань щодо версій із довгостроковою підтримкою, і Laravel іноді отримує критику через це. Це правда, що оновлення можуть спричинити незначні проблеми, однак, при належній увазі розробники можуть згладити процес.

Загалом, Laravel має свої власні плюси і мінуси, як і будь-який інший фреймворк. Однак можна з упевненістю сказати, що в більшості випадків Laravel зі своїми PHP-функціями та простими принципами забезпечить все, що потрібно розробникам для створення красивих та функціональних веб-програм.

3. МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

3.1. Структура програмного додатку

Клієнт-серверна архітектура стає популярною завдяки динамічному розвитку всесвітньої глобальної мережі Інтернет і зберіганню значної частини інформації на серверах в базах даних.

До певного моменту на систему управління базами даних (СУБД) покладалися лише завдання зберігання даних і організація доступу до них. З розвитком технологій до складу СУБД розробники стали включати новий компонент - процедурну мову програмування. З її допомогою в СУБД стало можливим створювати процедури для обробки даних, які можна викликати повторно. Такі процедури називаються збереженими процедурами. Наявність процедур дало можливість здійснювати деяку частину обробки даних на сервері.

Клієнт-серверна архітектура дозволяє розвантажити мережу і підтримувати несуперечливість даних за рахунок їх централізованої обробки. Однак, мови збережених процедур не пристосовані для повноцінної реалізації бізнес-логіки. Тому бізнес-логіка в клієнт-серверних інформаційних системах, як і раніше, реалізується на клієнтських комп'ютерах (рис.3.1).

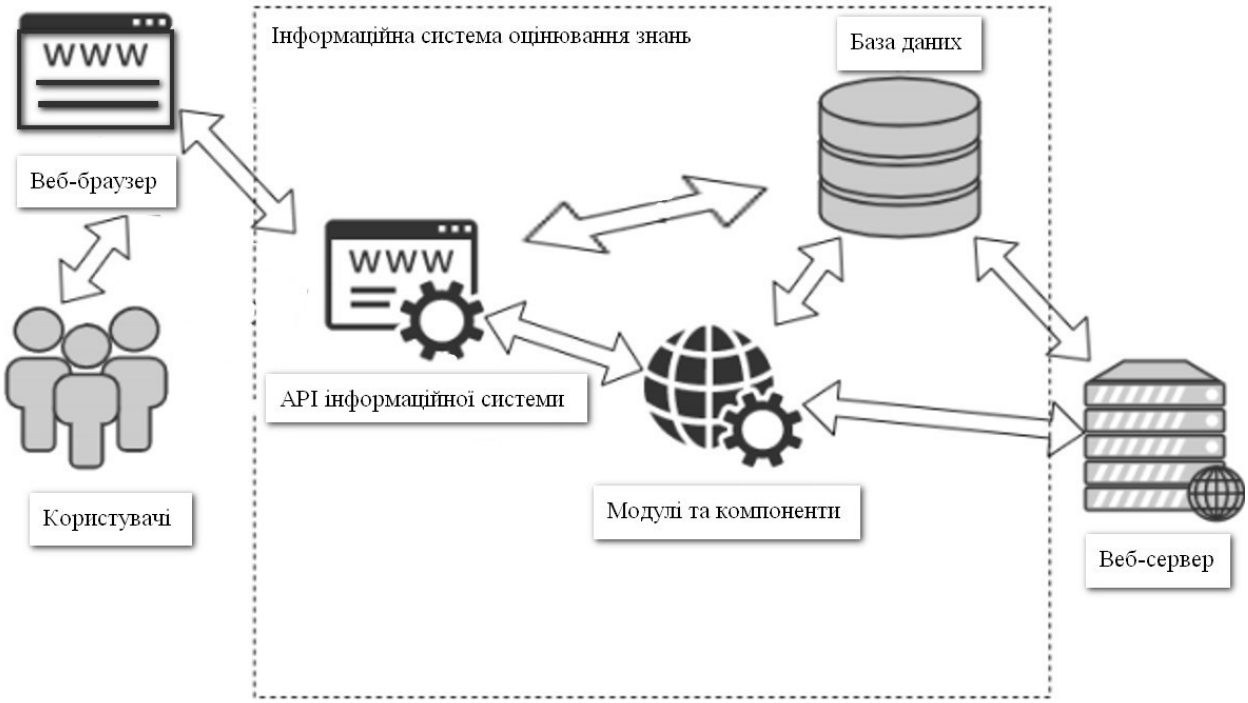

Рисунок 3.1 – Архітектура інформаційної системи

3.2. Структурно-функціональне моделювання процесу

Першим кроком при моделюванні інформаційної системи необхідно побудувати контекстну діаграму IDEF0 і діаграму декомпозиції IDEF1 контекстної діаграми IDEF0. Це наочно демонструє структуру необхідної інформації для підтримки роботи функцій системи. Дані діаграми проектуються, насамперед, для розробників, щоб полегшити проектування функцій і розуміння залежностей даних до тих чи інших функцій системи.

Першим кроком в моделюванні інформаційної системи була побудова контекстної діаграми IDEF0. Контекстна діаграма має рівень А-0 – це найвищий рівень абстракції для даного завдання. На рівні А-0 весь процес розглядається як функціональний блок з усіма відповідними робочими та керуючими об'єктами.

Центральним елементом моделі IDEF0 з точки зору роботодавця є функція «Організація процесу тестування», яка на схемі відображається у

вигляді функціонального блоку – прямокутника., а центральним елементом моделі IDEF0 з точки зору претендента є функція «Тестування Претендентів».

Згідно з методологією функціонального моделювання контекстної діаграми IDEF0 для роботи були визначені наступні дані, які зображуються на діаграмі стрілками:

- вхідні дані: інформація про користувачів;
- вихідні дані: список претендентів, список тестів та статистика відповідей;
- дані управління: перелік тестів, функції системи, час виконання проекту;
- дані механізмів: користувач, інформаційна система, технічне забезпечення.

Графічні зображення контекстних діаграм IDEF0 для даного проекту з точки зору роботодавця і претендента, представлені на рисунках 3.2 – 3.3.

Рисунок 3.2 – Контекстна діаграма IDEF0 з точки зору роботодавця

Рисунок 3.3 – Контекстна діаграма IDEF0 з точки зору претендента

Для формалізації процесів були розроблені і діаграми декомпозиції IDEF1 (для роботодавців та претендентів) контекстних діаграм IDEF0.

Діаграма декомпозиції детально описує функцію обробки нульового рівня. Тому функціональний блок рівня A-0 розкладається на набір взаємопов'язаних підфункцій. В даному проекті було спроектовано дві діаграми декомпозиції для розподілення функціональних можливостей: роботодавця та претендента. Кожен крок на діаграмах IDEF1 демонструє функції роботи сайту для кожної ролі від авторизації до отримання результатів. А також, діаграми IDEF1 відображають всі залежності, необхідні для роботи на кожному кроці.

Діаграми декомпозиції IDEF1 для роботодавців та претендентів показують, насамперед, залежність авторизованого користувача до кожного кроку.

Роботодавець має блоки створення тесту, збір даних та отримання списку претендентів. Ці кроки залежать від функцій системи, переліку тестів та часу виконання тесту (вказується при створенні тесту, в результатах

тестування, а також в списку претендентів, як додаткова інформація про результати тесту).

Претендент має блоки пошук тестів, тому що користувач, насамперед, виконує пошук бажаного тесту, а потім проходження тесту та отримання результатів. Проходження та результати тестів залежать від часу виконання тесту.

Всі блоки залежать від технічного забезпечення, роботи системи та ролі користувача.

Графічні зображення діаграм декомпозиції IDEF1 (для роботодавців та претендентів) контекстної діаграми IDEF0 представлені на рисунках 3.4 – 3.5.

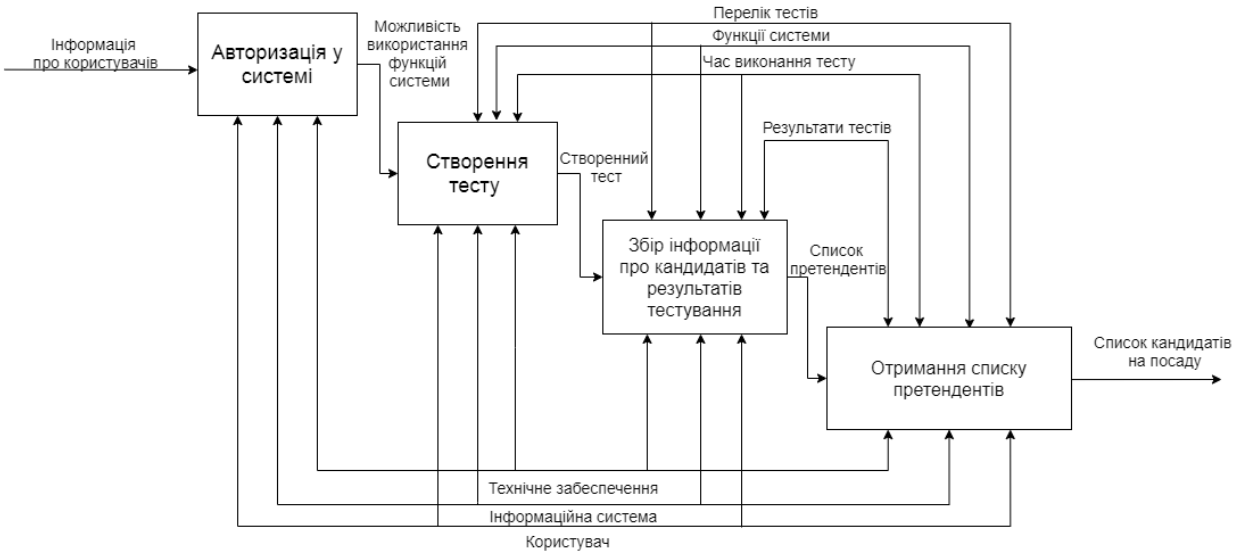

Рисунок 3.4 –Діаграма IDEF1 з точки зору роботодавця

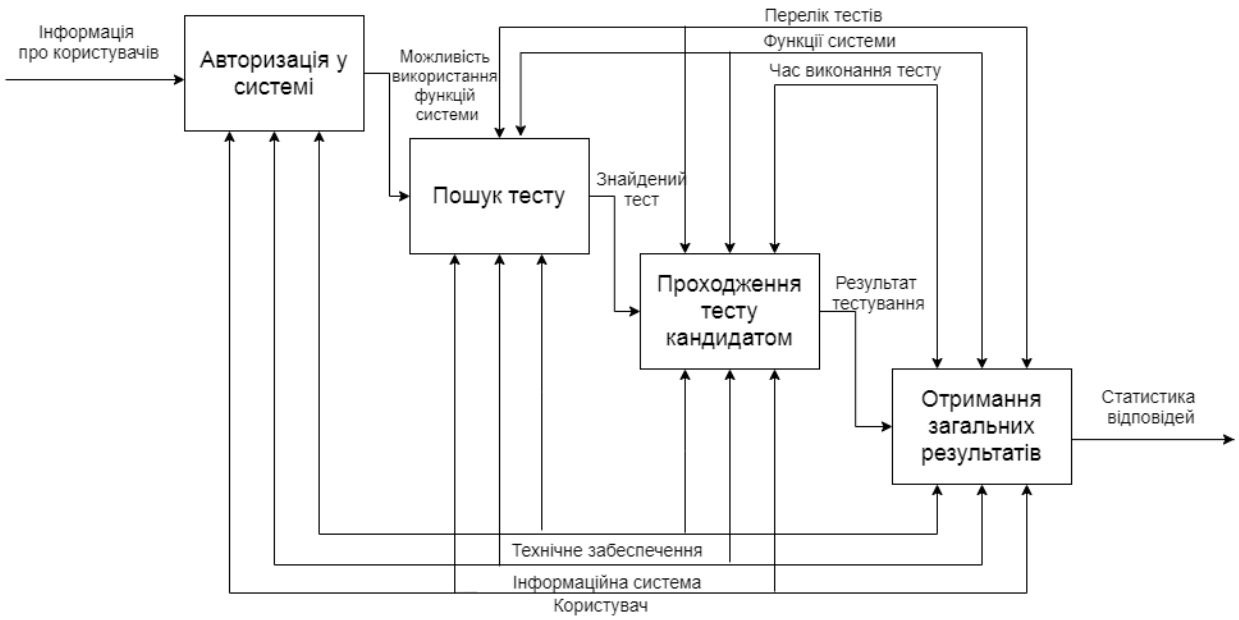

Рисунок 3.5 – Діаграма IDEF1 з точки зору претендента

3.3. Моделювання діаграми варіантів використання

В уніфікованій мові об'єктно-орієнтованого моделювання (UML) діаграма варіантів використання інформаційної системи узагальнює деталі користувачів інформаційної системи та моделює функціональність системи за допомогою акторів та варіантів використання. Варіанти використання - це набір дій, послуг та функцій, які система повинна виконувати.

Для даної інформаційної системи оцінювання знань була розроблена проста діаграма варіантів використання, яка не показує деталей її використання, а лише узагальнює деякі взаємозв'язки між акторами та варіантами використання ними цієї інформаційної системи. (рис.3.6).

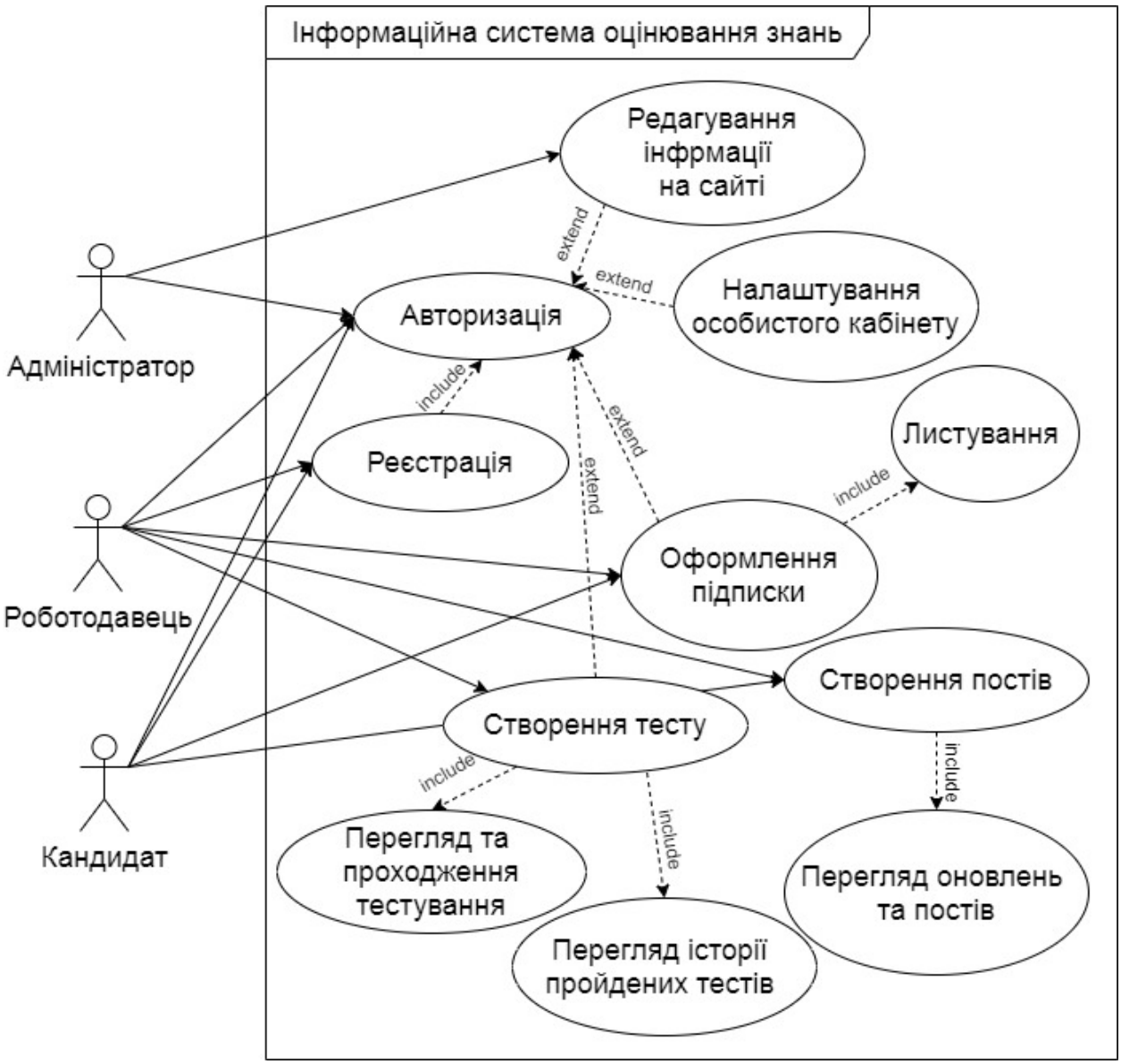

Рисунок 3.6 – Діаграма варіантів використання

Розглянемо детальніше акторів діаграми:

− адміністратор: має доступ до загальних даних на сайті із можливістю редагувати її;

− hr-спеціаліст: головна відмінність – створення тестів.

− кандидат: проходження тестування та слідкування оновлень.

Розглянемо детальніше варіанти використання:

− авторизація: авторизація на сайті;

− реєстрація: реєстрація нового користувача на сайті;

− редагування інформації на сайті: користувач може редагувати інформацію на особистій сторінці;

− налаштування особистого кабінету: зареєстрованому користувачеві відкриті можливості налаштування сторінки;

− оформлення підписки: можлива підписка на іншого користувача;

− листування: обмін повідомленнями з іншими користувачами;

− створення постів: створення інформаційних постів на сторінці;

− створення тесту: користувач типу «Роботодавець» може створювати тести;

− перегляд оновлень: перегляд новин та підписок;

− перегляд та проходження тесту: проходження створених тестів;

− перегляд історії пройдених тестів: роботодавець може переглядати, хто та з яким результатом пройшов тест.

3.4. Проектування бази даних

Етап планування бази даних (БД) передбачає розробку загального стратегічного плану, який дозволить ефективно реалізувати етапи життєвого циклу БД. По-перше, розробка бази даних до даної інформаційної системи повинна вирішувати такі питання:

− аналіз існуючих інформаційних систем;

− доцільність створення нової інформаційної системи;

− обсяг робіт і ресурсів, вартість проекту;

− розробка методології збору даних, визначення їх формату;

− визначення послідовності проектування і реалізації застосувань.

Процес проектування БД являє собою послідовність переходів від неформального мовного опису інформаційної структури предметної області до формалізованого опису об'єктів предметної області в термінах деякої

моделі. Проектування БД складається з таких етапів як системний аналіз предметної області, концептуальне, логічне та фізичне проектування.

Завдяки фреймворку Laravel ми не створюємо SQL-запити до БД, а використовували готову бібліотеку «Laravel query builder», що спрощує роботу з запитами (не потрібно створювати власні) (рис. 3.7).

```
$email = DB::table('users')->where('name', 'John')->value('email');

$user = DB::table('users')->find(3);
```

Рисунок 3.7 – Приклад готових запитів до БД з бібліотеки «Laravel query builder»

На основі зібраної інформації створимо ERD діаграму інформаційної системи (Entity-Relationship model, тобто модель «сутність - зв'язок») (рис.3.8) та створимо детальний опис даних (табл.3.1).

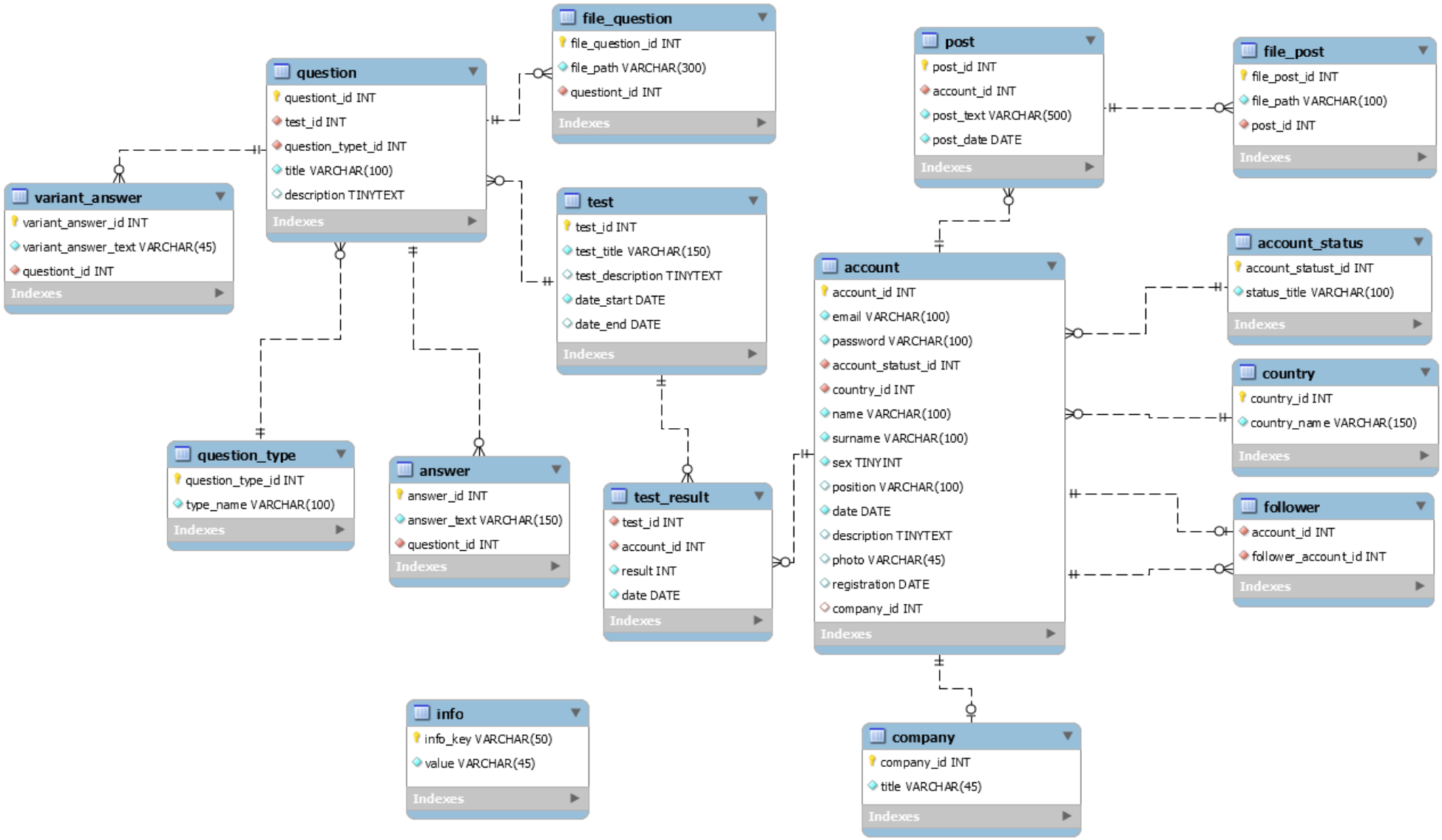

Рисунок 3.8 – ERD діаграма інформаційної системи

Таблиця 3.1 – Опис таблиць бази даних

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
1	account	account_id	Ідентифікатор аккаунта	INTEGER	PK	Не пустий
		email	Електронна пошта	VARCHAR(100)		Не пустий
		password	Пароль від аккаунта	VARCHAR(100)		Не пустий
		account_status_id	Ідентифікатор статусу аккаунта	INTEGER	FK	Не пустий
		country_id	Ідентифікатор країни	INTEGER	FK	Не пустий
		name	Ім’я користувача	VARCHAR(100)		Не пустий
		surname	Прізвище користувача	VARCHAR(100)		Не пустий
		position	Посада HR менеджера	VARCHAR(100)		
		date	Дата народження	DATE		Не пустий
		description	Коротка інформація (резюме)	TEXT		
		photo	Фото аккаунта	VARCHAR(100)		
		sex	Стать користувача аккаунта	BOOLEAN		Не пустий
		registration_date	Дата реєстрації	DATE		Не пустий
2	account_status	account_status_id	Ідентифікатор статусу аккаунта	INTEGER	PK	Не пустий
		account_status_title	Назва статусу	VARCHAR(100)		Не пустий
3	country	country_id	Ідентифікатор країни	INTEGER	PK	Не пустий
		country_name	Назва країни	VARCHAR(100)		Не пустий
4	post	post_id	Ідентифікатор посту блога	INTEGER	PK	Не пустий
		account_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
		post_text	Текст посту	TEXT		Не пустий
		post_date	Дата написання посту	DATE		Не пустий
5	file_question	file_id	Ідентифікатор файлу	INTEGER	PK	Не пустий
		file_path	Посилання на файл	VARCHAR(100)		Не пустий
		questiont_id	Ідентифікатор завдання	INTEGER	FK	Не пустий

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
6	file_post	file_id	Ідентифікатор файлу	INTEGER	PK	Не пустий
		file_path	Посилання на файл	VARCHAR(100)		Не пустий
		post_id	Ідентифікатор поста	INTEGER	FK	Не пустий
7	question	questiont_id	Ідентифікатор завдання	INTEGER	PK	Не пустий
		test_id	Ідентифікатор тесту	INTEGER	FK	Не пустий
		question_typet_id	Ідентифікатор типу завдання	INTEGER	FK	Не пустий
		title	Назва завдання	VARCHAR(100)		Не пустий
		description	Опис завдання	TEXT		
8	test	test_id	Ідентифікатор тесту	INTEGER	PK	Не пустий
		test_title	Назва тесту	VARCHAR(100)		Не пустий
		test_description	Опис тесту	TEXT		
		date_start	Початок тесту	DATETIME		Не пустий
		date_end	Кінець тесту	DATETIME		
9	test_result	test_id	Ідентифікатор тесту	INTEGER	FK	Не пустий
		account_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
		result	Результат	INTEGER		Не пустий
10	variant_answer	variant_answer_id	Ідентифікатор варіанту відповіді	INTEGER	PK	Не пустий
		variant_answer_text	Текст варіанту відповіді	VARCHAR(100)		Не пустий
		questiont_id	Ідентифікатор завдання	INTEGER	FK	Не пустий
11	question_type	question_type_id	Ідентифікатор типу завдання	INTEGER	PK	Не пустий
		type_name	Назва типу завдання	VARCHAR(100)		Не пустий
12	answer	answer_id	Ідентифікатор відповіді	INTEGER	PK	Не пустий
		answer_text	Текст відповіді	VARCHAR(100)		Не пустий
		questiont_id	Ідентифікатор завдання	INTEGER	FK	Не пустий
13	follower	account_id	Ідентифікатор аккаунта	INTEGER	FK	Не пустий
		follower_account_id	Ідентифікатор аккаунта на який підписаний користувач	INTEGER	FK	Не пустий

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
14	Info	info_key	Ключ запису	VARCHAR(100)	PK	Не пустий
		value	Значення	TEXT		Не пустий

4. ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОЕКТУ

4.1 Програмна реалізація

Інформаційна система оцінювання знань в області тестування програмного забезпечення була реалізована у вигляді SPA-додатків (Single Page Application), тобто інформаційна система розміщена на одній сторінці, яка завантажує одночасно весь код разом із завантаженням самої сторінки. Це дозволяє при переході на інші сторінки не перезавантажувати їх. Це здійснено за допомогою фреймфорка Vue.js та бібліотеки маршрутизації «Vue Router».

При створенні інформаційної системи серверна частина була розроблена за допомогою фреймворка Laravel версії 7.3 (рис. 4.1).

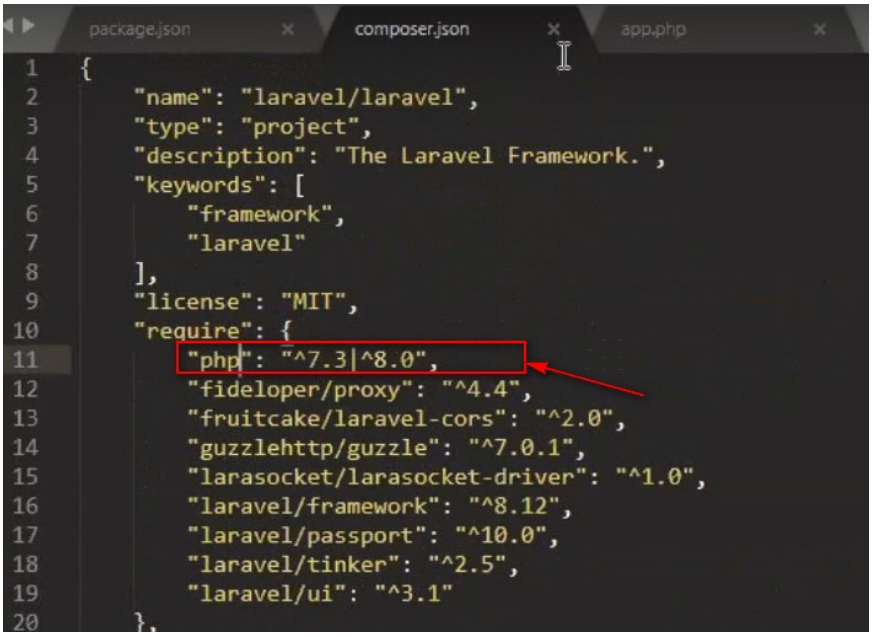

Рисунок 4.1 – Фреймворк Laravel, версія 7.3

Система має декілька типів користувачів системи. А саме – адміністратор, роботодавець та кандидат. При реєстрації можна обрати між «Кандидат» та «Роботодавець».

Для реєстрації необхідно заповни обовязкові поля даними, які система буде використовувати в подальшому. Реєстрація відбувається за допомогою API-запитів на сервер до БД з використанням бібліотеки “Passport” (рис. 4.2).

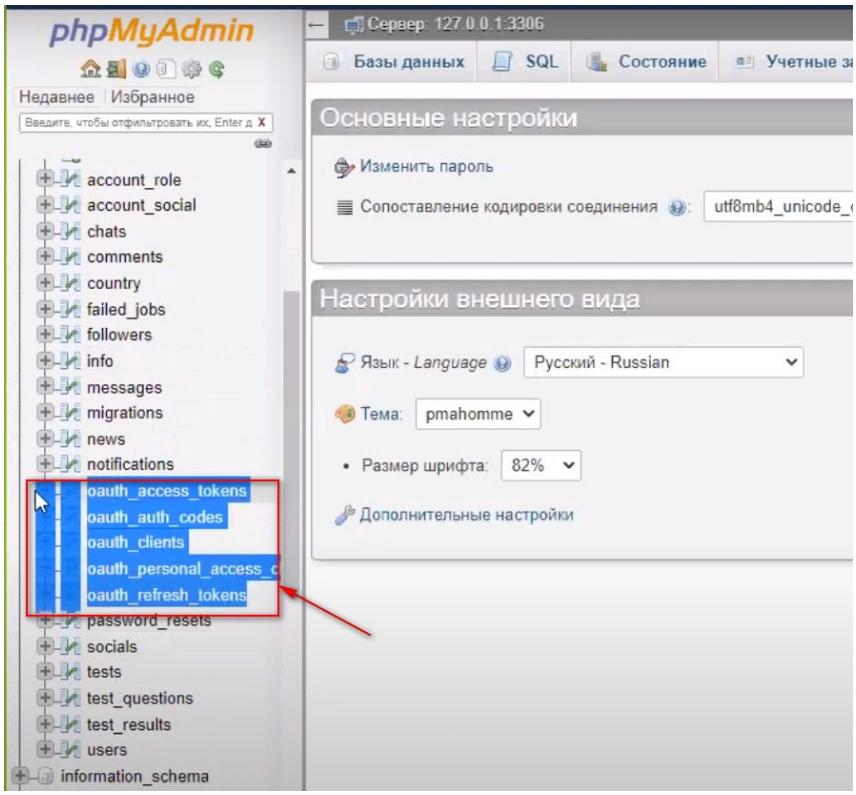

Рисунок 4.2 – Таблиці БД, які використовує бібліотека “Passport”

Роботодавець може створювати власні тести, з трьома різними типами відповідей: написати відповідь текстом, обрати один чи де-кілька чек-боксів, та вказати правильну послідовність.

Випадаючий список (Drag and Drop) реалізовано на стороні front-end за допомогою фреймворка vue.js і його бібліотеки «Vuedraggable» (рис.4.3).

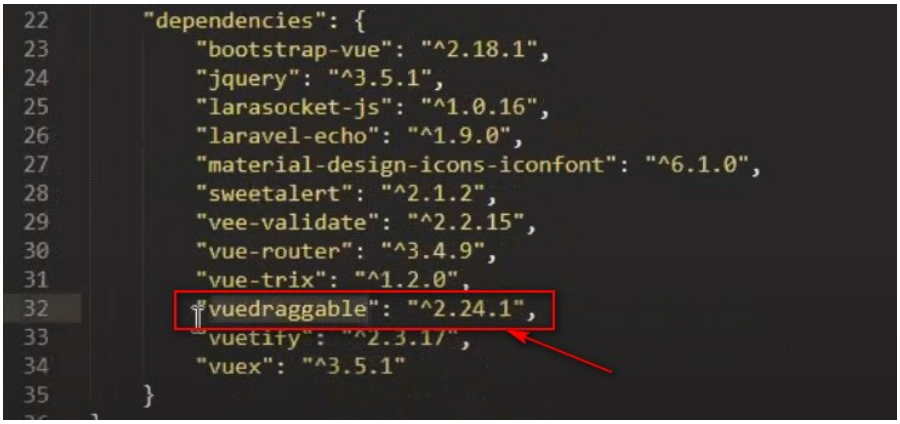

Рисунок 4.3 – Використання бібліотеки «Vuedraggable» (фреймворк vue.js)

В інформаційній системі реалізована функція пошуку, який здійснюється за ключовими словами.

Також, в системі реалізована складна функція чату з безліччю кімнат. Листування ведеться в режимі реального часі. Все листування зберігається в БД. Чат реалізовано за допомогою бібліотеки «Larasocket», яка відповідає за отримання, обробку та відправлення повідомлень у відповідь.

Також, для роботи чату, ми використовували канали «Слухачі» в режимі реального часу. Кожен чат ведеться по окремому каналу (рис.4.4).

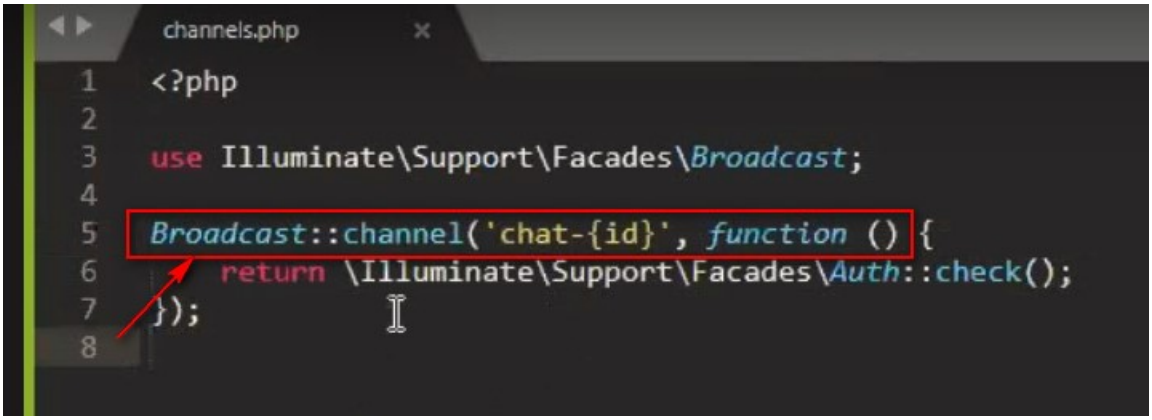

Рисунок 4.4 – Канали чата для роботи в режимі реального часу

Для того, щоб повідомлення, яке прийшло, було автоматично додано до чату, використано інструмент «Laravel Echo», який спрощує найнеобхідніші і найважчі аспекти побудови складних взаємодій WebSockets (рис. 4.5).

Для функціонування чату в особистому кабінеті Larasocket отримується ключ «token».

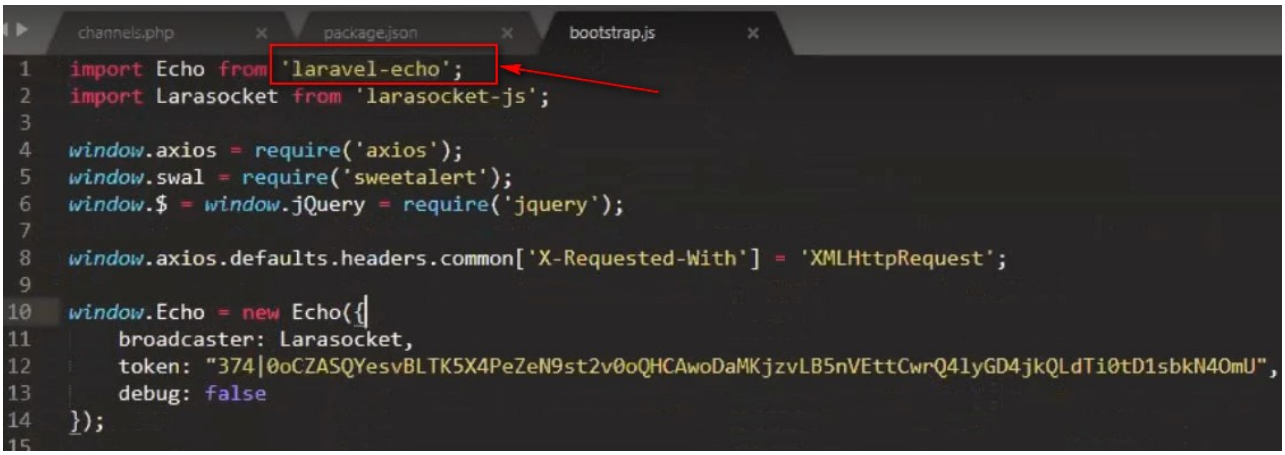

Рисунок 4.5 – Інструмент «Laravel Echo» для відображення повідомлень в режимі реального часу

Для безпеки користувачів, навіть адміністратор сайту, не може в БД переглядати паролі користувачів. Для кодування паролів використовується стандартна бібліотека «Laravel» і функція «Hash», яка створює хеш паролі, використовуючи сильний, незворотний алгоритм хешування (рис 4.6).

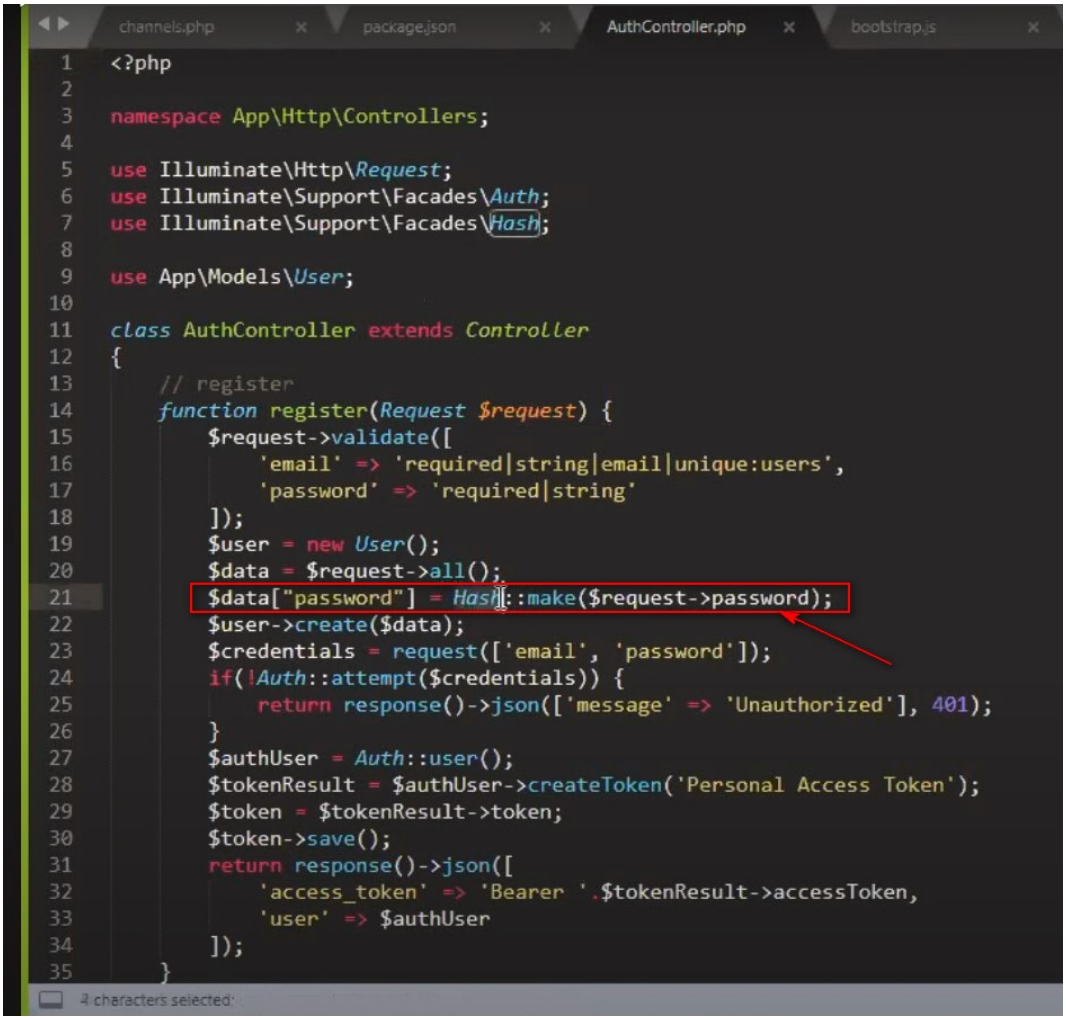

Рисунок 4.6 – Функція Hash для кодування паролів користувачів

Розглянемо розподіл функціоналу детальніше в табл.4.1.

Таблиця 4.1 – Функціонал за групами користувачів

№	Користувач	Можливості користувача
1	Адміністратор	Перегляд загальної інформації;
		Підтримка сайту;
		Блокування користувачів;
		Перевірка контенту.
2	Кандидат	Створення посту;

№	Користувач	Можливості користувача
		Перегляд інформації про іншого користувача;
		Оформлення підписки на інших користувачів;
		Листування з іншими користувачами;
		Проходження тестування;
		Додавання тестів до «Бажаних» для проходження їх в подальшому;
		Перегляд історії пройдених тестів та їх результатів;
		Написання повідомлення адміністратору;
		Виконувати фільтрацію та пошук на сайті.
3	Роботодавець	Створення посту;
		Перегляд інформації про іншого користувача;
		Оформлення підписки на інших користувачів;
		Листування з іншими користувачами;
		Створення тесту за допомогою конструктора;
		Перегляд результатів тестування за тим чи іншим створеним тестом;
		Редагування та налаштування створеного тесту;
		Написання повідомлення адміністратору;
		Виконувати фільтрацію та пошук на сайті.

Для детального представлення інтерфейсу інформаційної системи тестування та комунікації розглянемо табл.4.2.

Таблиця 4.2 – Опис сторінок додатку

№	Назва	Опис сторінки
1	Головна сторінка	Сторінка з короткою інформацією про дану інформаційну систему.
2	Реєстрація	Сторінка для реєстрації користувача. Користувач повинен ввести такі дані для створення аккаунту на даному сервісі: − Ім’я; − Прізвище; − Електронна пошта; − Пароль; − Дата народження; − Стать; − Тип аккаунту (HR / Кандидат).
3	Авторизація	Сторінка для авторизації. Користувач повинен ввести такі дані: − Електронна пошта; − Пароль.
4	Підписки	Кожен авторизований користувач може виконати підписку на оновлення іншого користувача.
5	Новини	На даній сторінці відображаються публікації користувачів, на яких оформлена підписка. Також присутнє оповіщення про нові тестування.
6	Тестування	Функціонал змінюється в залежності від типу аккаунту. Кандидат – перегляд пройдених тестів та тестів, що були додані до бажаних для проходження в

№	Назва	Опис сторінки
		майбутньому. Користувач може переглянути результати пройдених тестів. Рекрутер чи HR – створені тести. Також користувач може перейти до конструктора тесту.
7	Конструктор тесту	Конструктор для створення тесту. Користувач може додати будь-яку кількість запитань. Типи запитання: − Одна відповідь; − Декілька відповідей; − Власна відповідь; − Співвідношення; − Порядкові відповіді.
8	Налаштування створеного тесту	Сторінка для редагування уже створеного тесту. На даній сторінці можна редагувати як запитання, так і час проведення чи іншу інформацію.
9	Особистий кабінет	Сторінка з інформацією про користувача. На даній сторінці є можливість перегляду власних постів та створення нових.
10	Налаштування аккаунту	Сторінка для редагування основної інформації про користувача та додаткове її налаштування.
11	Листування	Невід’ємний засіб зв’язку між підприємствами, партнерами та клієнтами. Отже, дана сторінка допоможе підтримати ділове листування.

4.2 Використання програмного додатку

Робота інформаційної системи оцінювання знань в області тестування програмного забезпечення представлена на рисунках 4.7 – 4.34.

Головна сторінка інформаційної системи складається з навігаційних конопок та опису переваг (рис. 4.7).

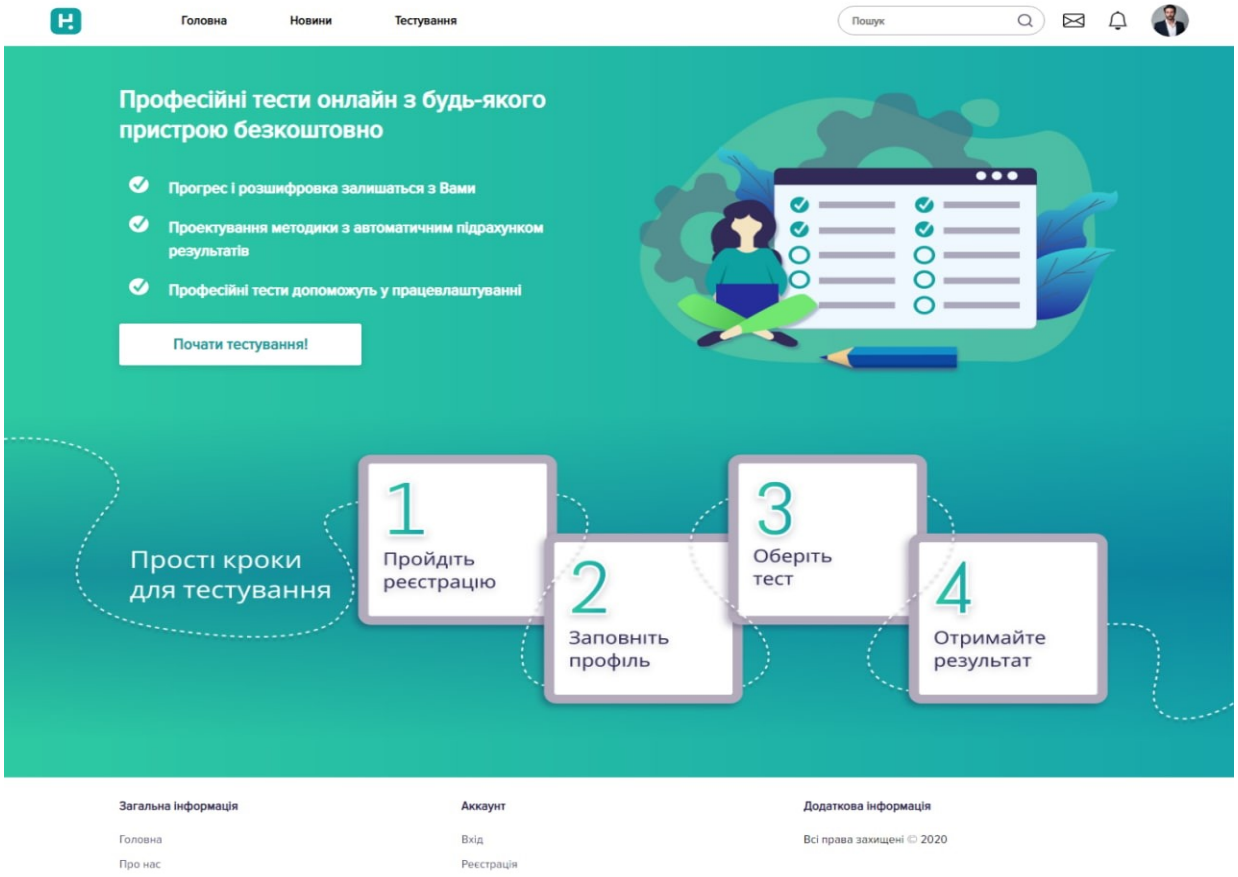

Рисунок 4.7 – Головна сторінка інформаційної системи

Вікно реєстрації має поля для обовязкового заповнення. Якщо ці поля не заповнені система підсвічує їх красною рамкою. Також, реєстрація не відбувається, якщо не відмічено чек-бокс «Продовжуючи відвідування сайту, ви погоджуєтесь на обробку Ваших особистих персональних даних відповідно до

Закону України «Про захист персональних даних» від 01.06.2010 р. № 2297-VI» (рис. 4.8).

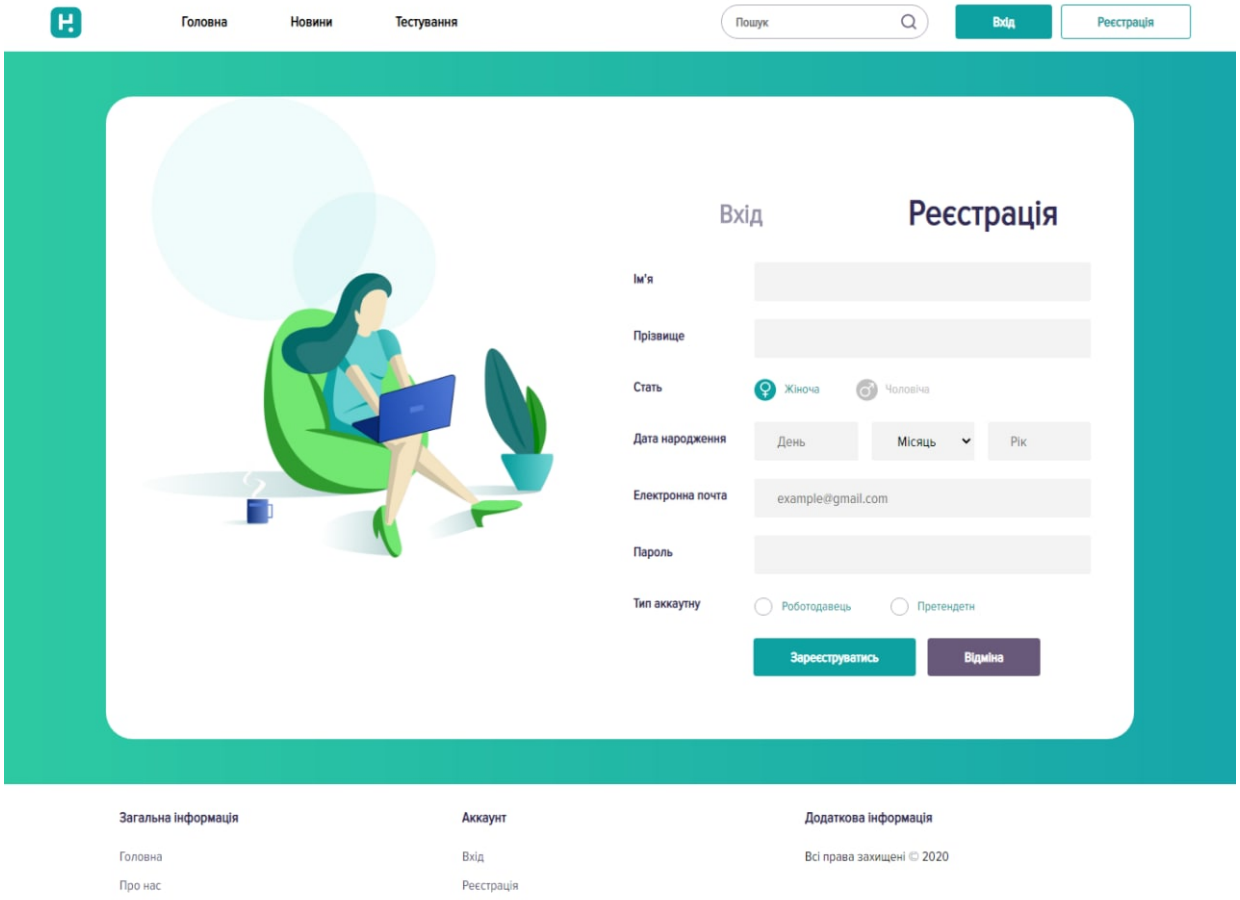

Рисунок 4.8 – Сторінка реєстрації

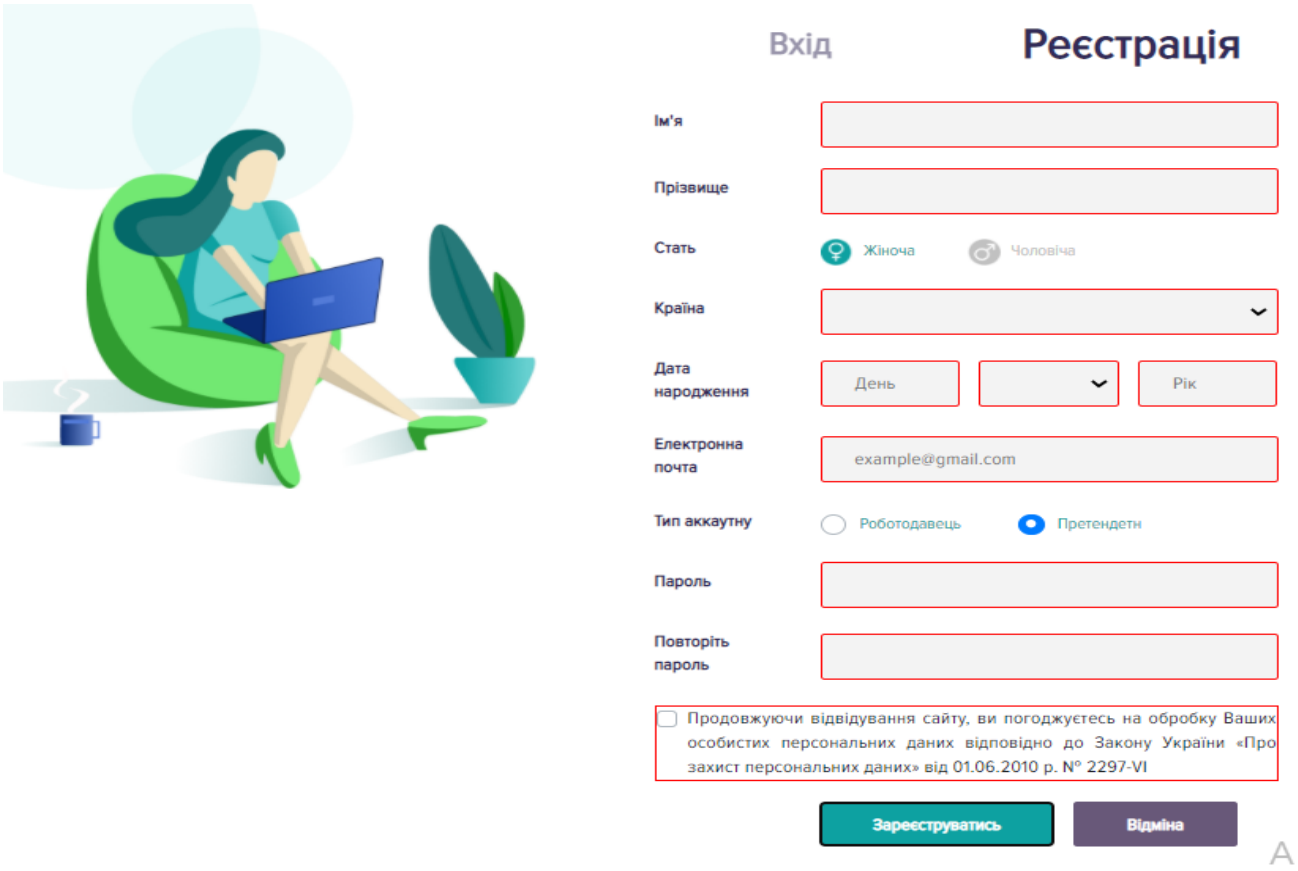

Рисунок 4.9 – Обовязкові поля для заповнення

Після реєстрації або авторизації користувача на головній сторінці зявляються навігаційні посилання (рис. 4.10).

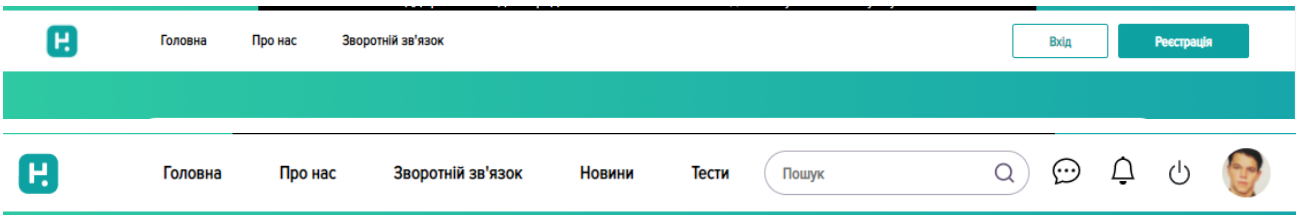

Рисунок 4.10 – Навігаційні посилання після реєстрації або авторизації користувача

На вкладці «Про нас» вказана додаткова інформація про сайт (рис. 4.11).

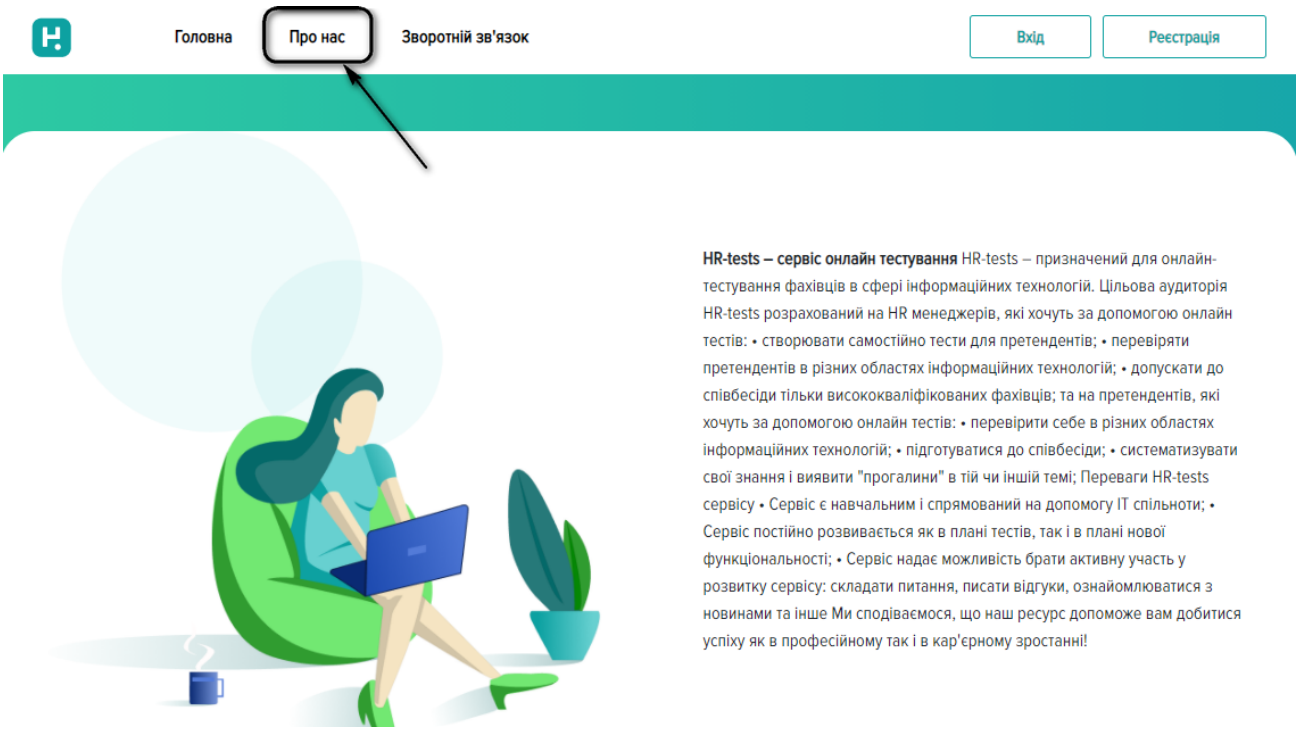

Рисунок 4.11 – Пункт меню «Про нас»

За посиланням «Новини» знаходяться статті та останні новини з життя сайту (рис.4.12).

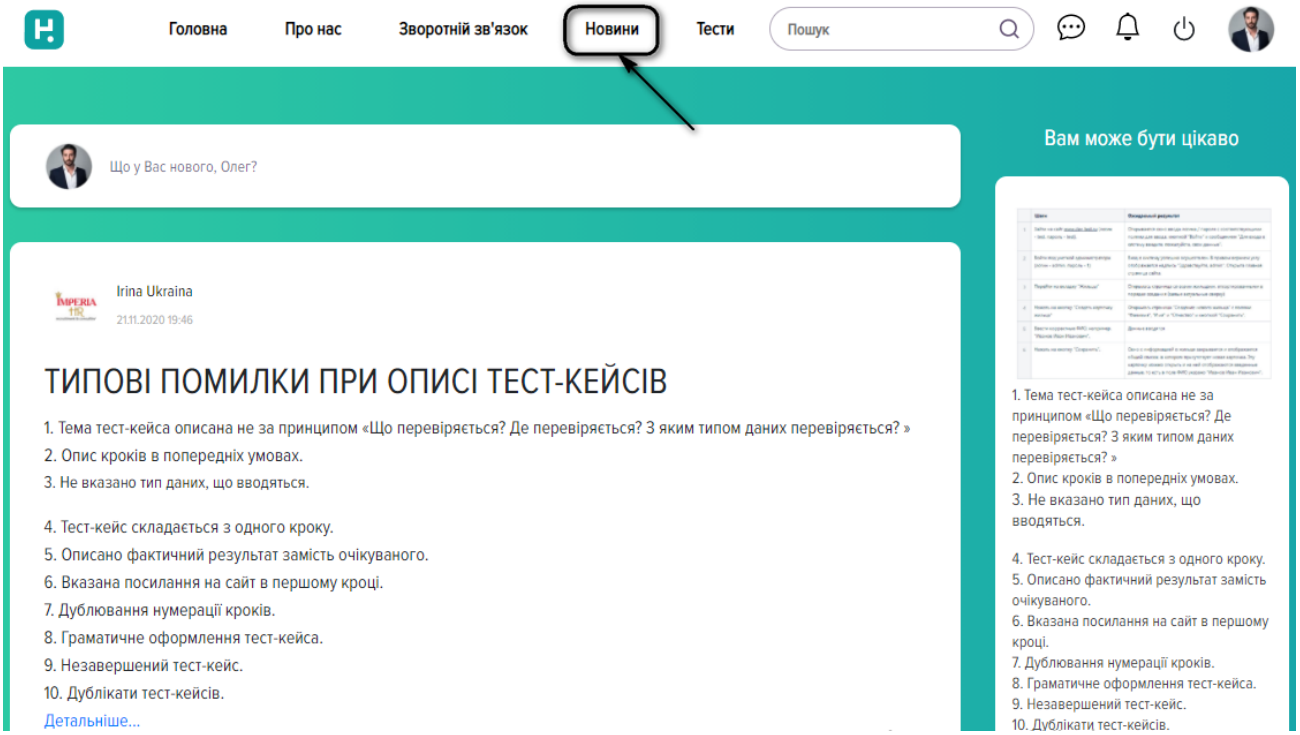

Рисунок 4.12 – Пункт меню «Новини»

За посиланням «Тести» відкривається перелік створених роботодавцями тестів. У претендента є можливість обрати тест і пройти його (рис. 4.13).

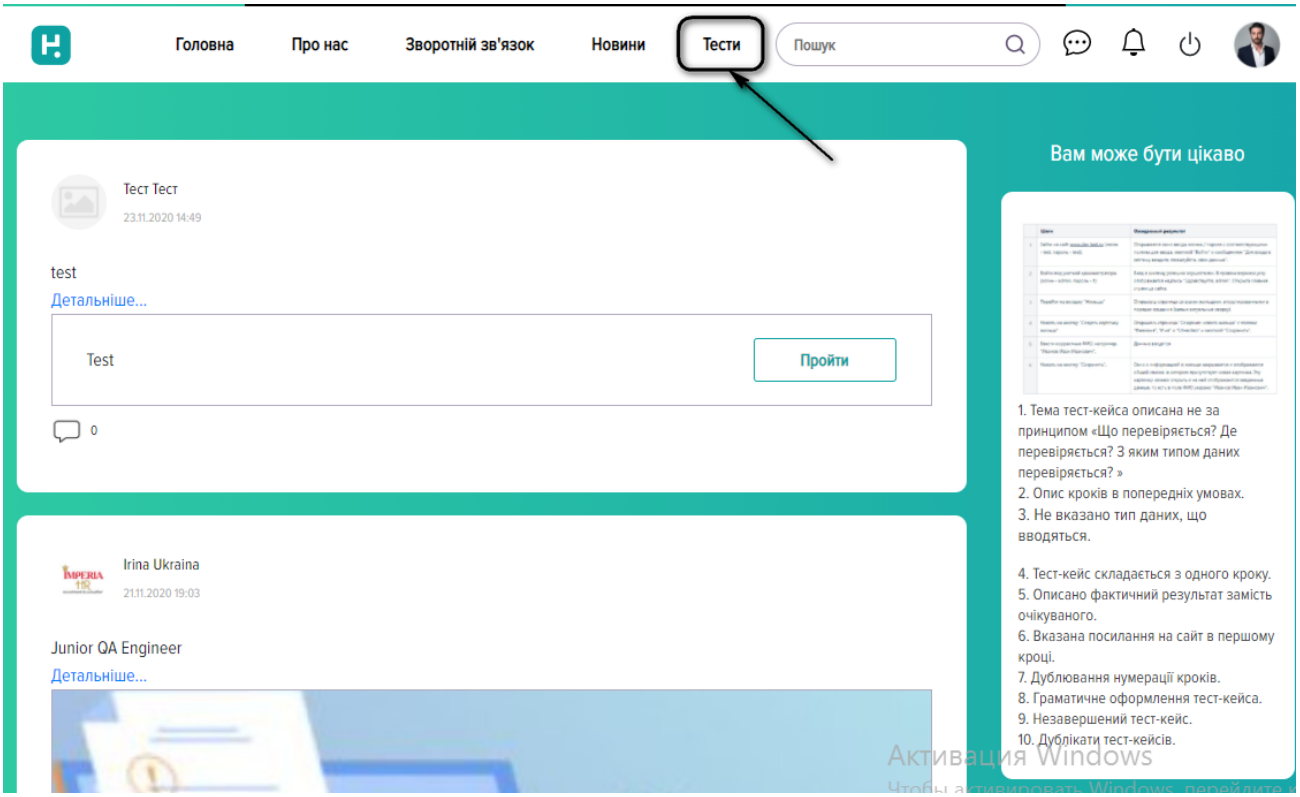

Рисунок 4.13 – Пункт меню «Тести»

Після обрання тесту і натискання кнопки «Пройти» буде показано детальний опис тесту (рис. 4.14)..

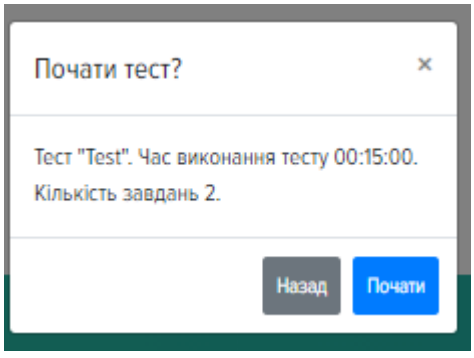

Рисунок 4.14 – Початок проходження тесту

В правому верхньому куті зявляється таймер часу, який відображає скільки часу залишилося до закінчення тестування.

Тест може мати три різних видів відповідей:

- написати відповідь у текстове поле;
- обрати правильну відповідь зі списку;
- проставити правильну послідовність (рис. 4.15).

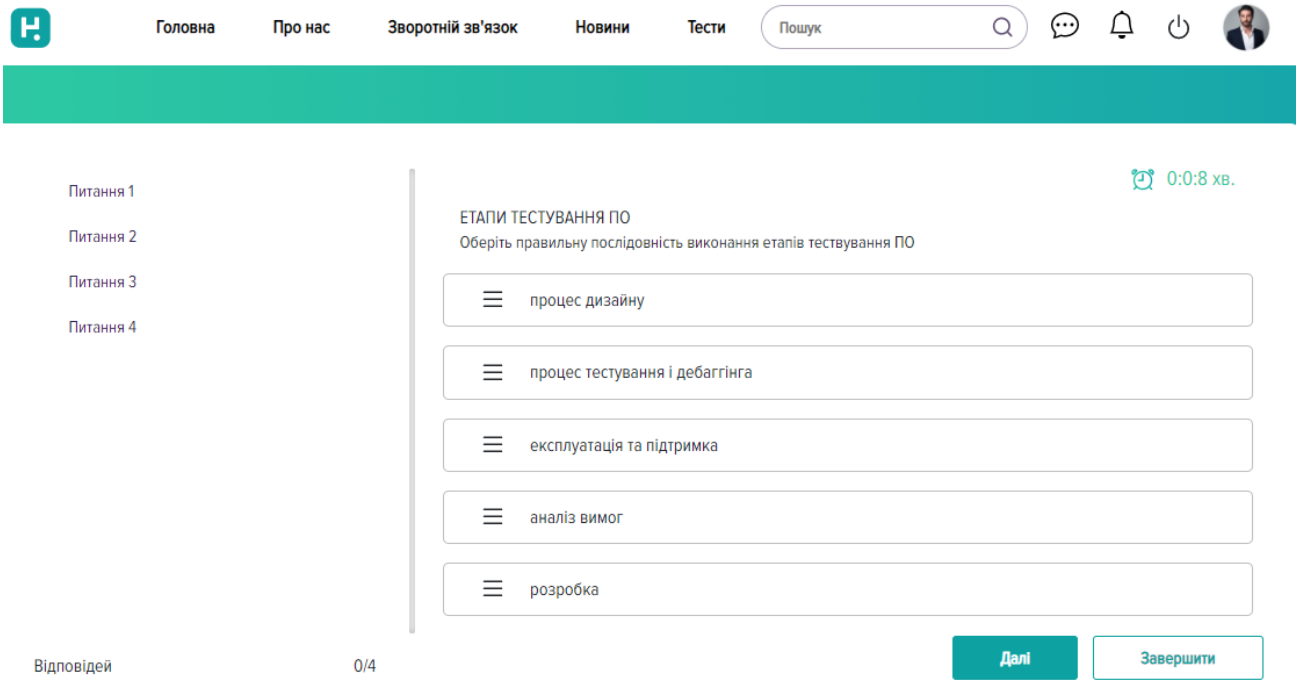

Рисунок 4.15 – Приклад тесту (проставити правильну послідовність)

Після завершення тесту у всплиаючому віконці – повідомленні буде відображено результат проходження даного тесту (рис. 4.16).

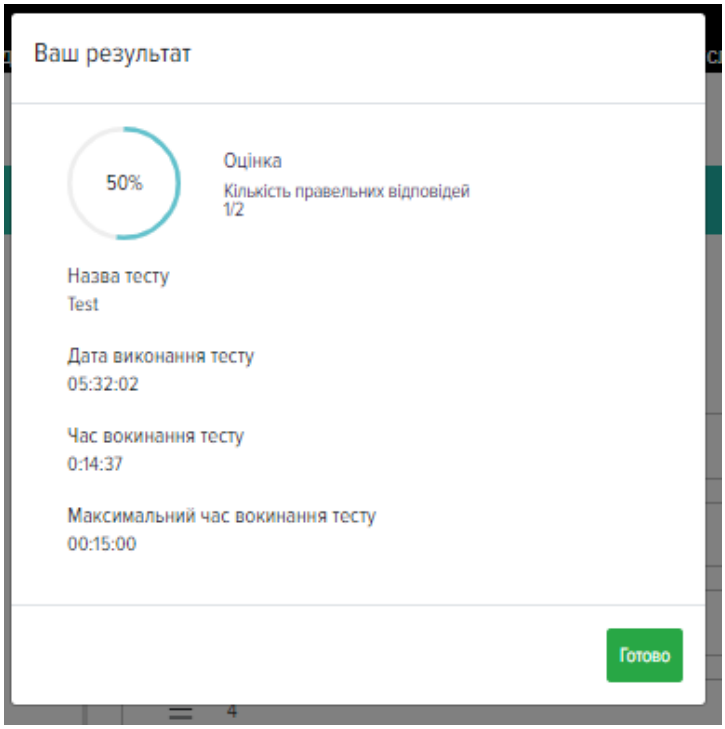

Рисунок 4.16 – Результат тестування (повідомлення)

Результати тестування будуть відображені в акаунті в меню Мої тести (рис.4.17).

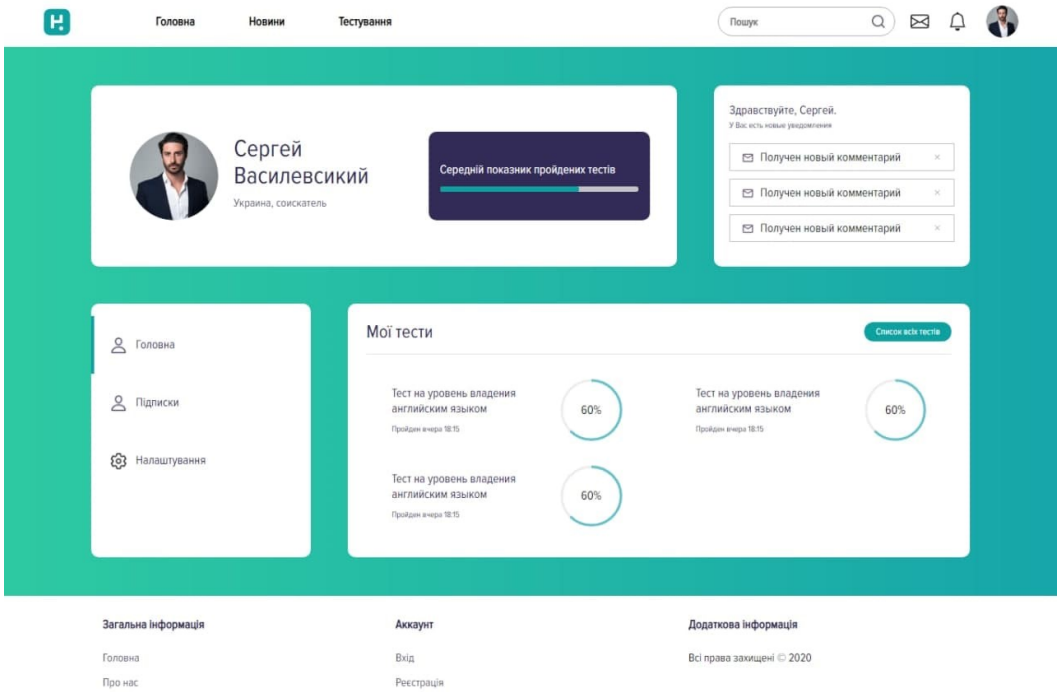

Рисунок 4.17 – Результат тестування на сторінці

Обравши пройдений тест є можливість переглянути деталі проходження тесту або видалити його (рис. 4.18).

Рисунок 4.18 – Детальний результат обраного тестування

В особистому кабінеті у Роботодавців є можливість створення свого нового тесту. Потрібно спочатку створити тест та задати обмеження часу на його виконання (рис. 4.19).

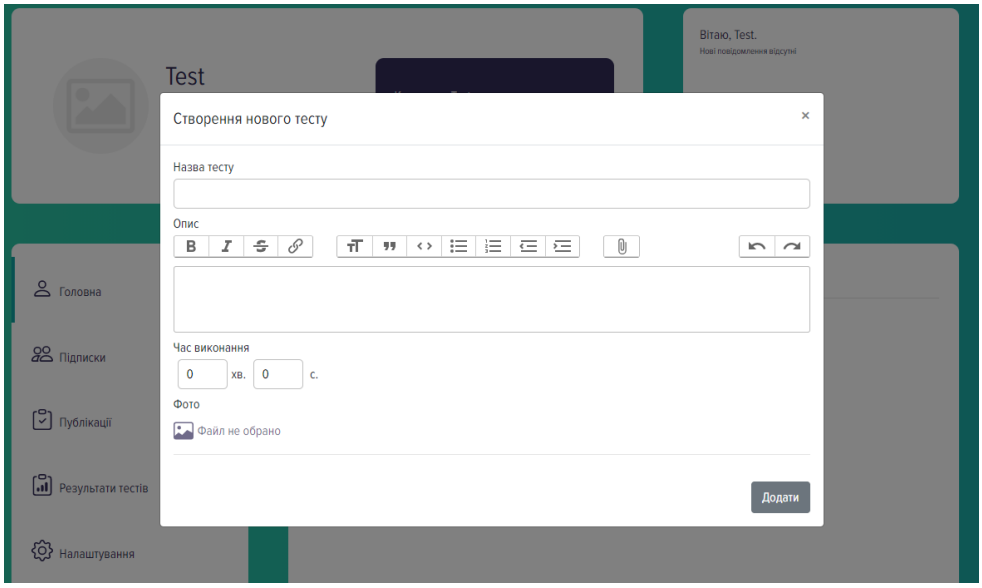

Рисунок 4.19 – Створення нового тесту

Після чого потрібно створити завдання для тесту (скласти запитання). Запитання у тесті можно як додавати, так і видаляти (рис. 4.20).

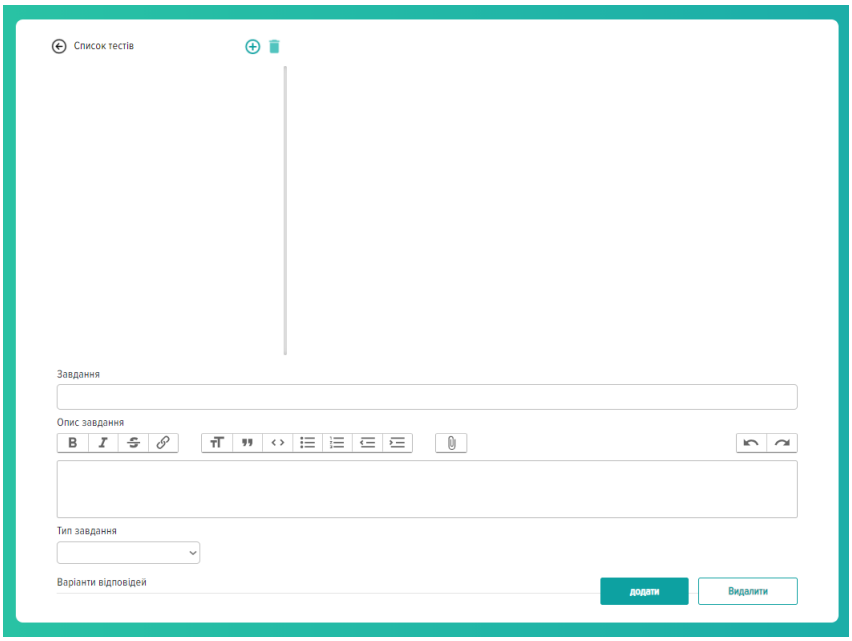

Рисунок 4.20 – Додавання запитань до тесту

В відповідях зі списку можуть бути використані ілюстрації (рис. 4.21).

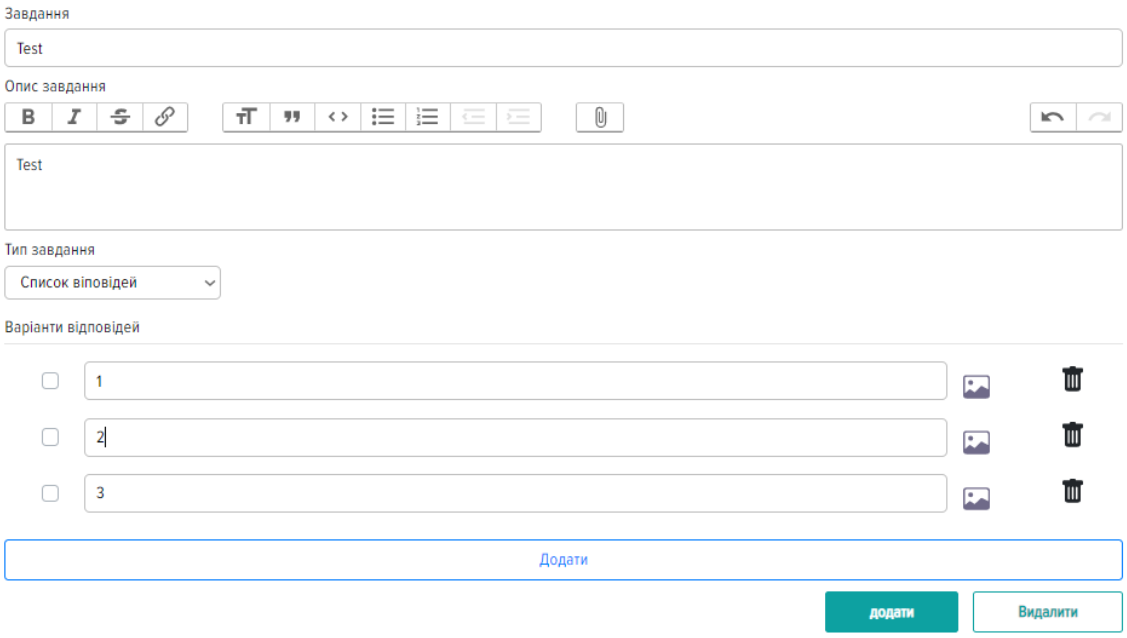

Рисунок 4.21 – Відповідь типу «Список відповідей»

Рисунок 4.22 – Відповідь типу «Текстове поле»

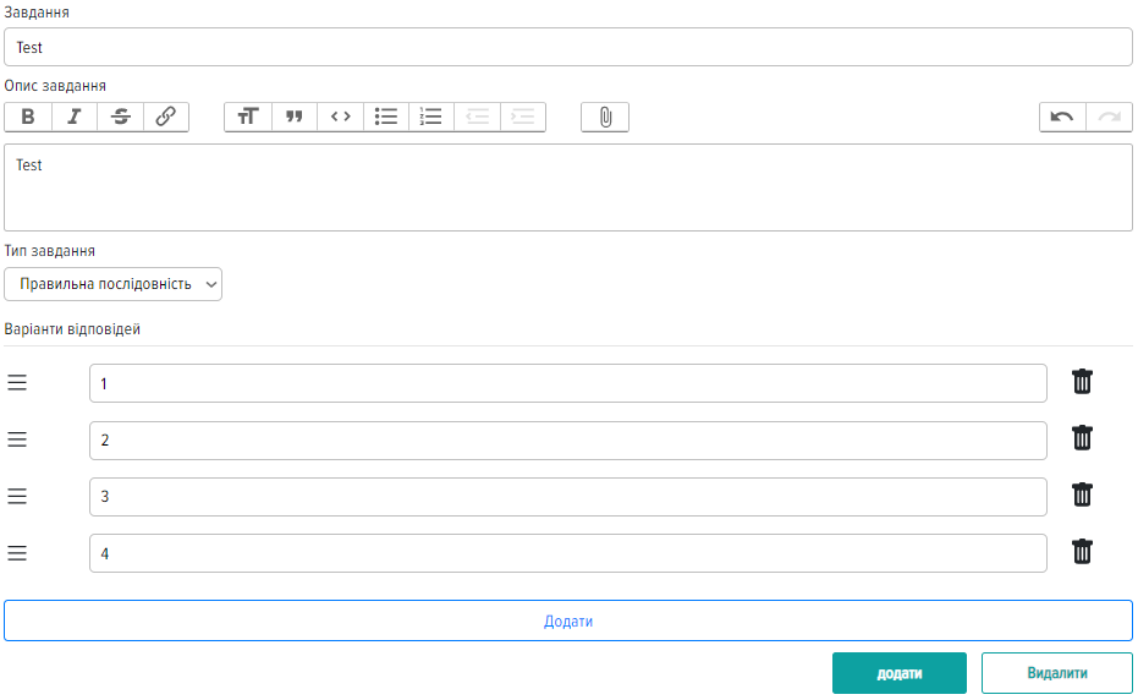

Рисунок 4.23 – Відповідь типу «Правильна послідовність»

Після створення тесту він буде відображатися на особистій сторінці роботодавця (рис. 4.24 – 4.25).

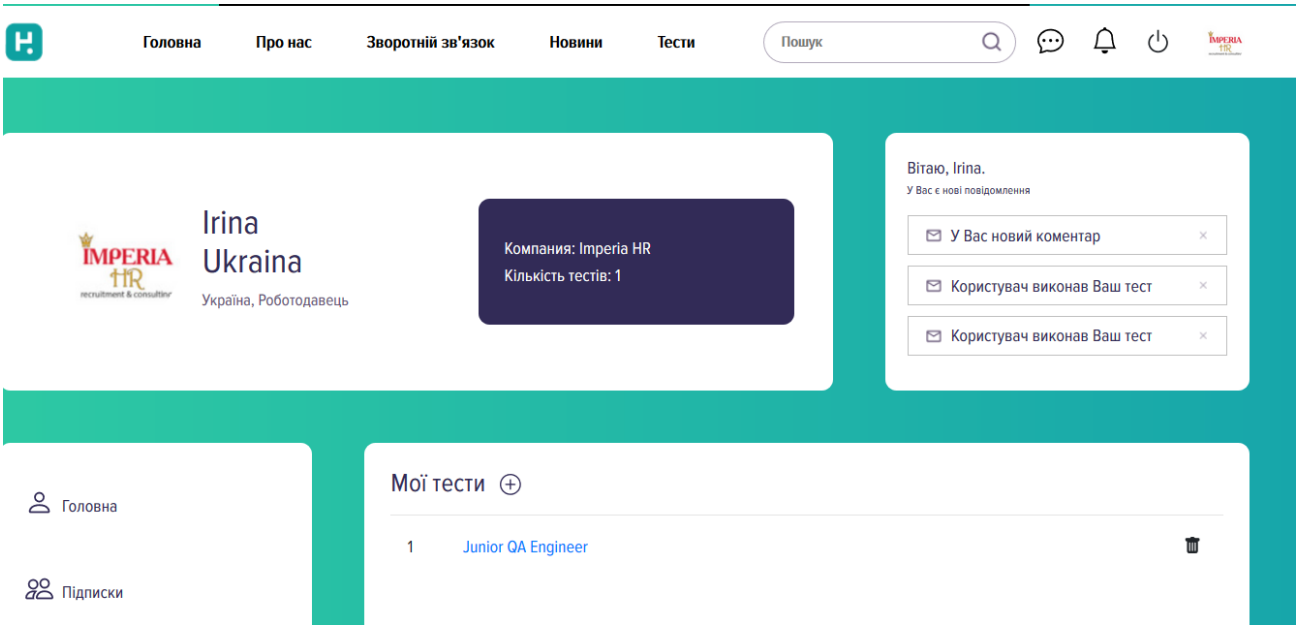

Рисунок 4.24 – Особиста сторінка роботодавця

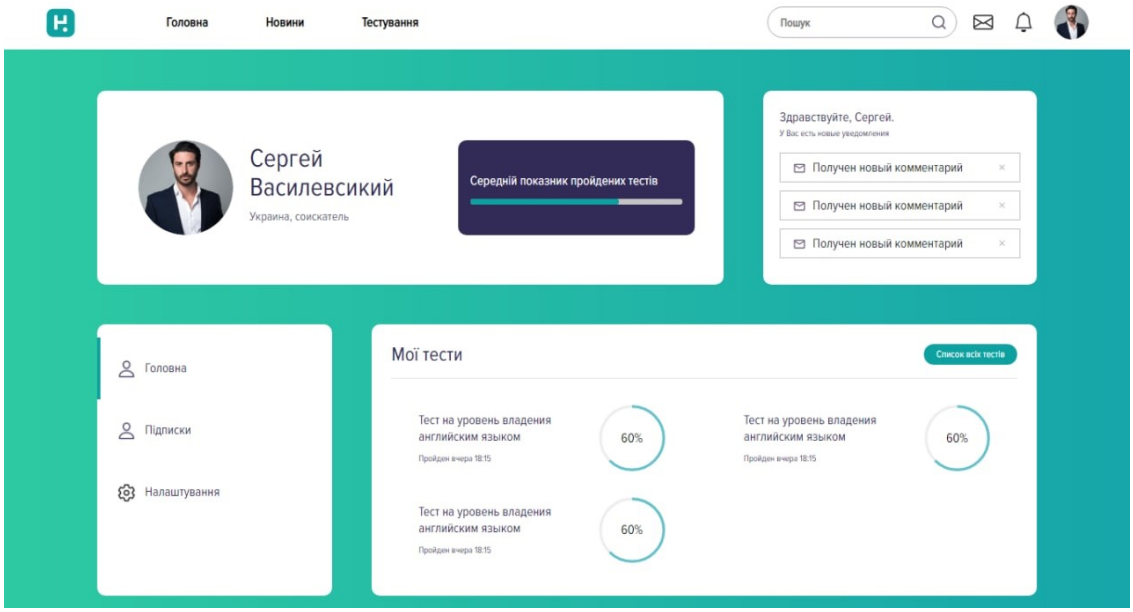

Рисунок 4.25 – Особисто сторінка претендента (головна)

На сайті реалізована функція «Підписатися». Щоб підписатися на інший аккаунт необхідно зайти на той акаунт і натиснути кнопку «Підписатися» (рис.4.26).

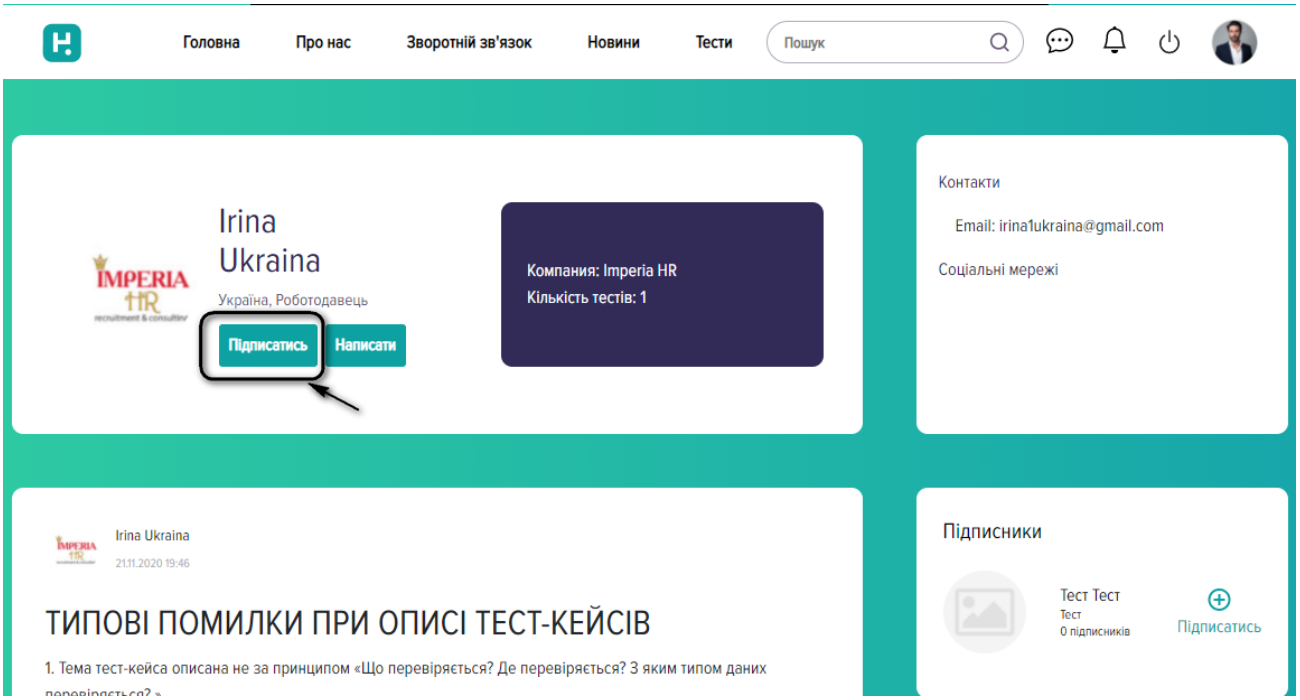

Рисунок 4.26 – Функція «Підписатися»

Акаунт, на який підписалися, отримає повідомлення, що на нього підписалися. В акаунті, який підписався будє відображено його підписки в меню «Мої підписки» (рис. 4.27).

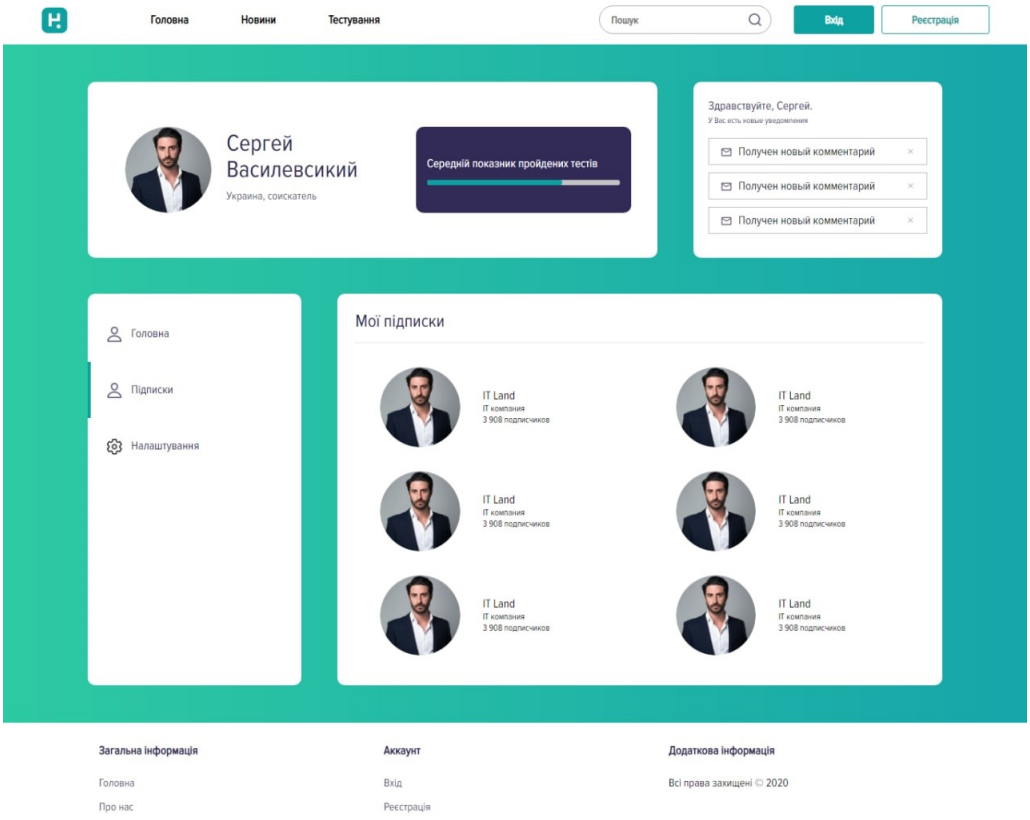

Рисунок 4.27 – Особисто сторінка користувача (підписки)

На сайті реалізована функція «Написати», завдяки якій можно спілкуватися між зареєстрованими акаунтами за допомогою чат-повідомлень в реальному часі (рис. 4.28).

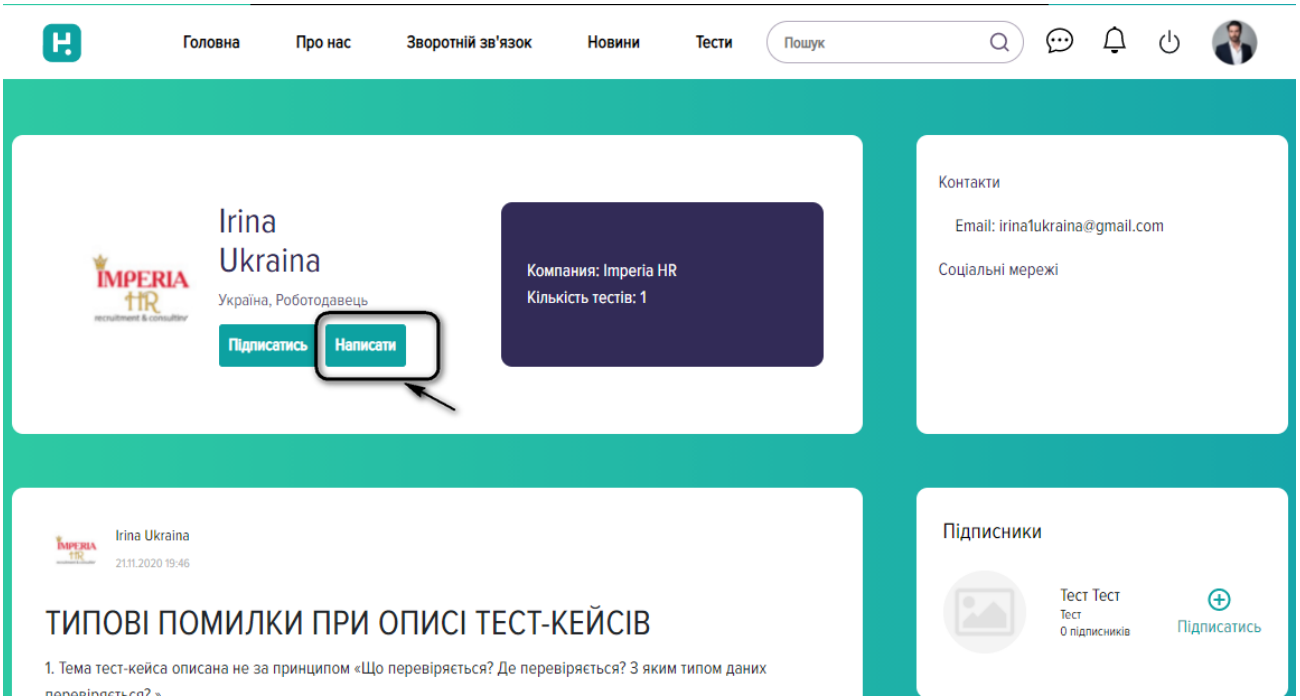

Рисунок 4.28 – Функція «чат-повідомлення в реальному часі»

Особиста сторінка має зручну навігацію. В пункті меню «Налаштування» існує можливість переглянути та змінити публічну та приватну інформацію (рис.4.29).

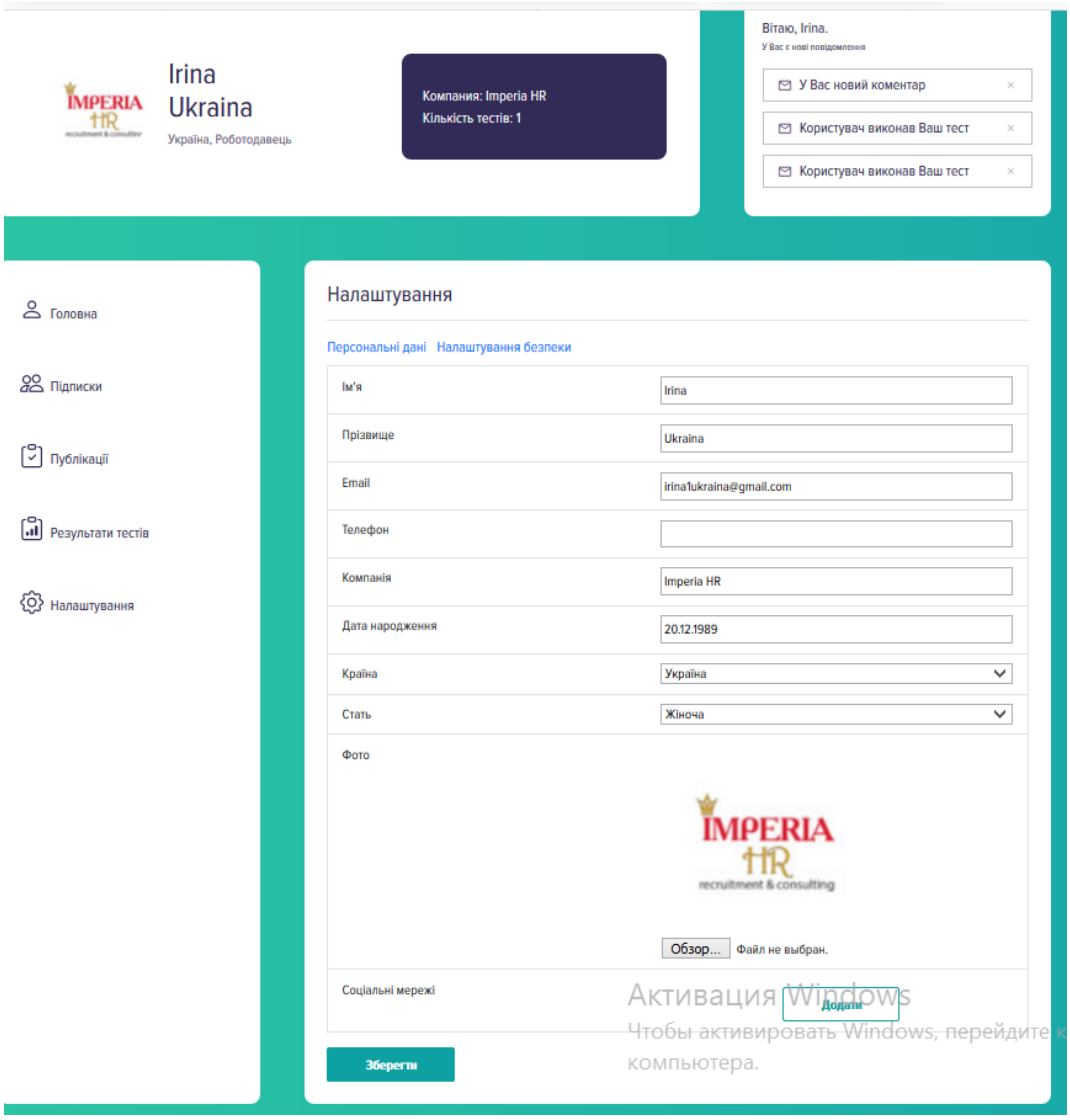

Рисунок 4.29 – Особисто сторінка користувача (налаштування)

Під новинами або статтями є можливість залишати коментарі (рис. 4.30).

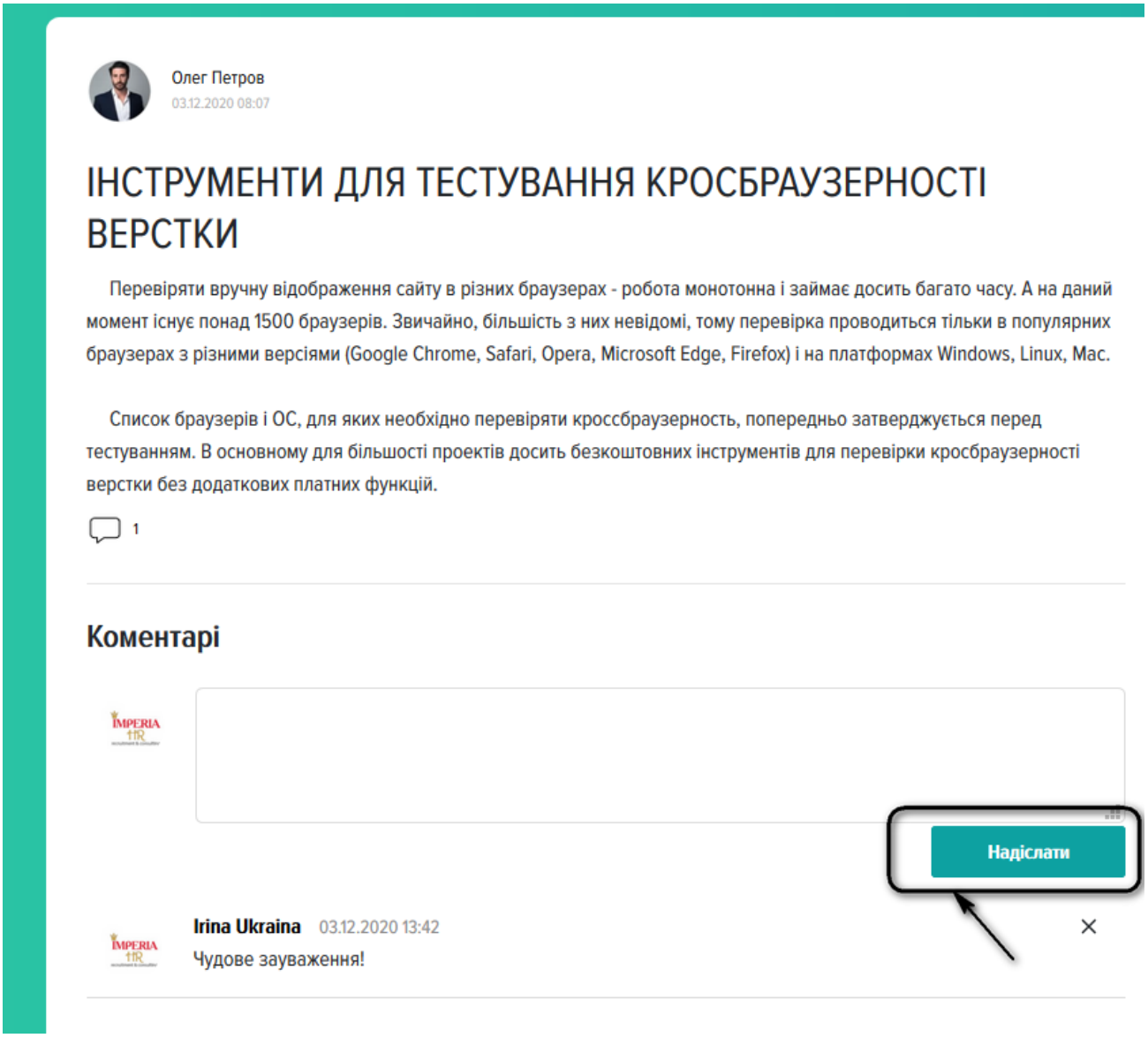

Рисунок 4.30 – Функція «Коментувати»

4.3 Адміністрування сайту

Адміністратор сайту має такі можливості як:

- Перегляд загальної інформації на сайті;
- Підтримка сайту;
- Блокування користувачів;
- Перевірка загального контенту.

Дані функції представлені на рисунках 4.31-4.37.

Рисунок 4.31 – Авторизація

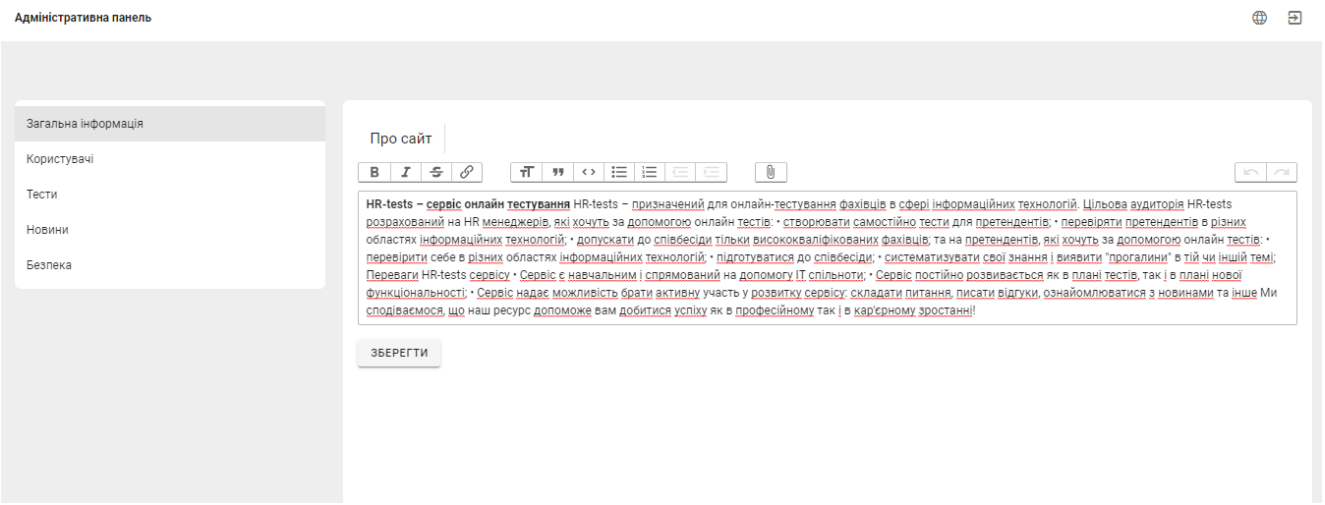

Рисунок 4.32 – Редагування загального контенту

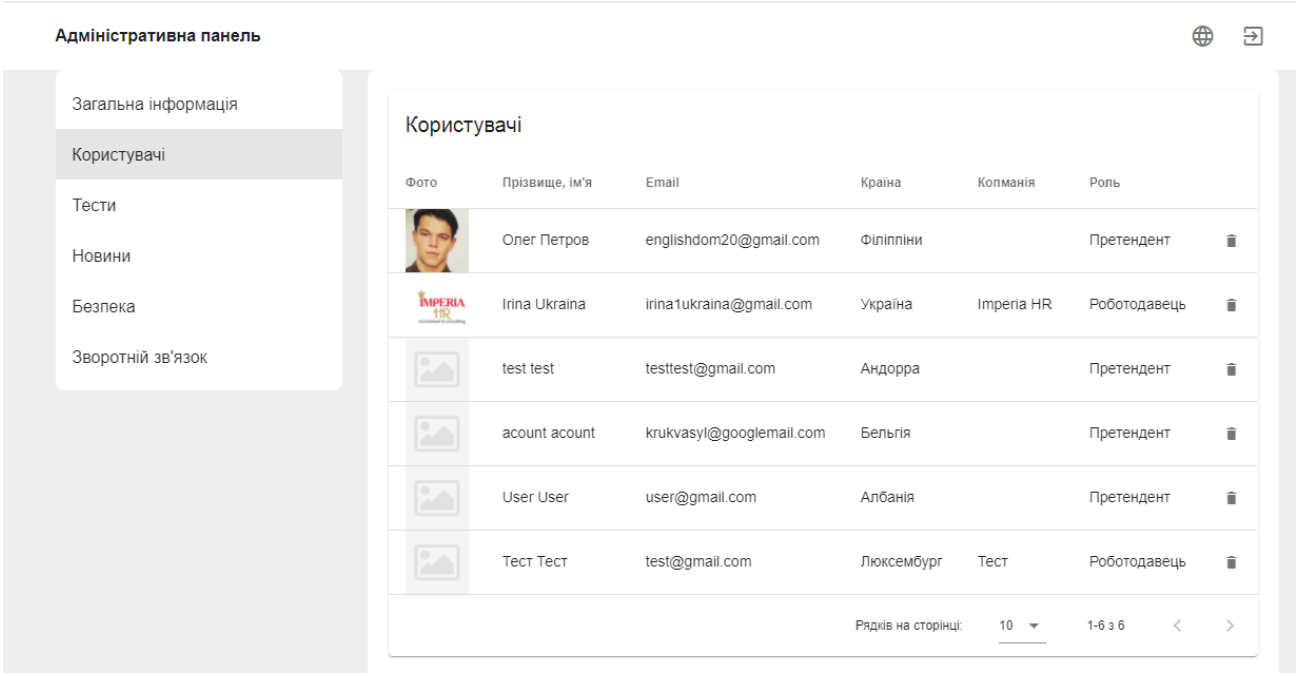

Рисунок 4.33 – Редагування контенту (користувачі)

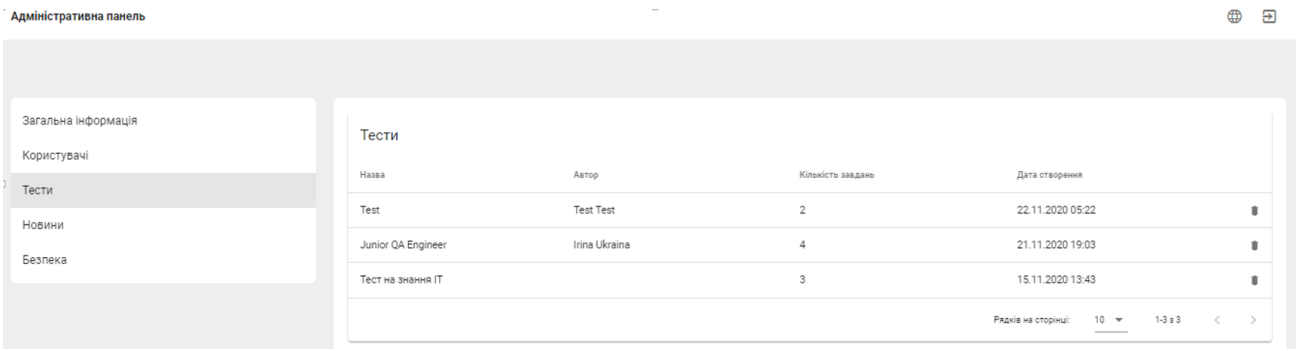

Рисунок 4.34 – Редагування контенту (тести)

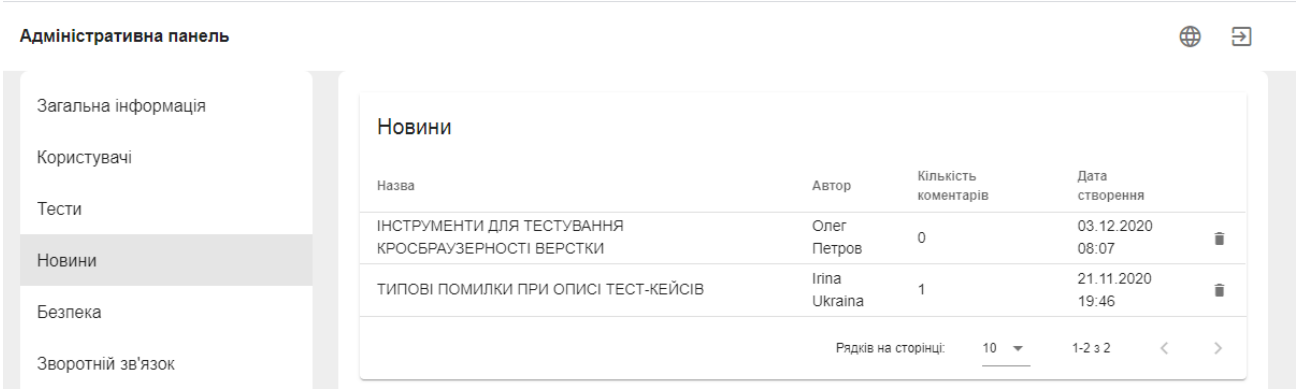

Рисунок 4.35 – Редагування контенту (новини)

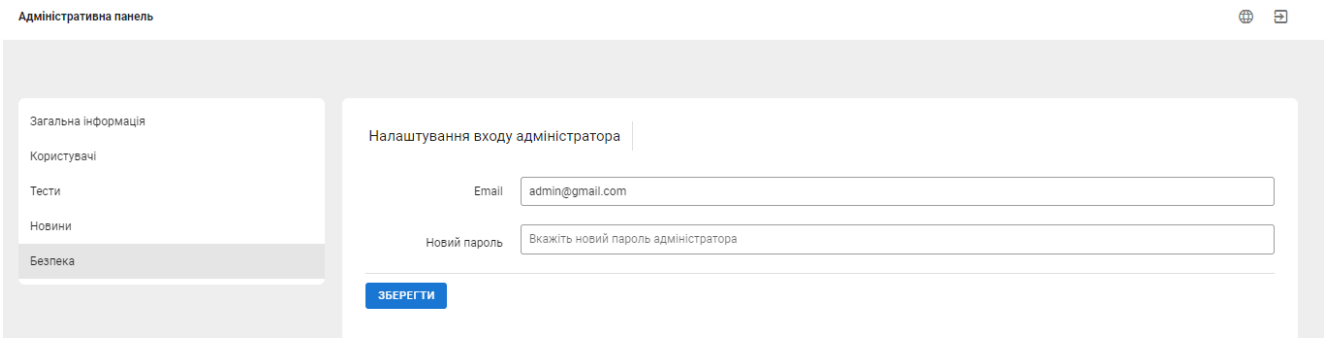

Рисунок 4.36 – Редагування контенту (безпека)

Інформаційна система оцінювання знань адаптована під мобільні пристрої (рис.4.37).

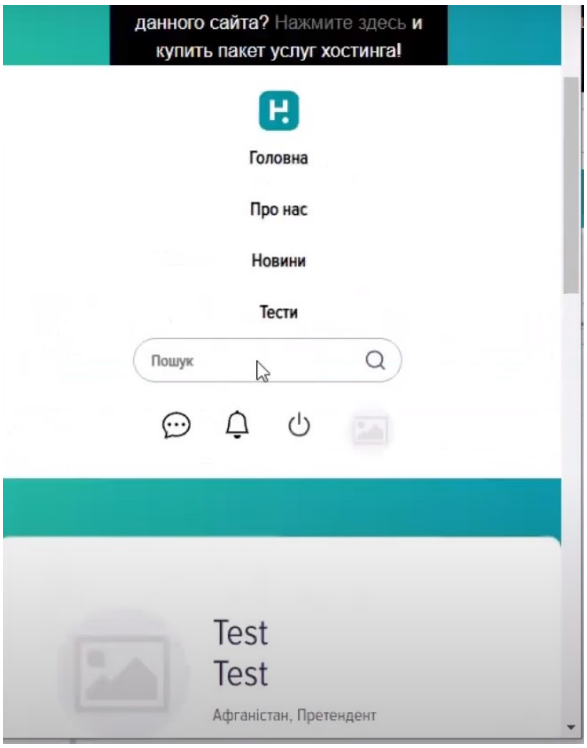

Рисунок 4.37 – Адаптація інформаційної системи під мобільні пристрої

ВИСНОВКИ

Нинішній рік, а також пандемія COVID-19, серйозно змінили багато речей, і в тому числі вимоги до функціонування ринку праці. Раптово з'явилися виклики і додаткові можливості для бізнесу та трудового життя загалом. І хоча процес тотальної цифровізації (з англ. digitalization) почався задовго до пандемії, тепер він набуває небачених масштабів. В умовах, що змінилися, бізнес почав активно готуватися до нової реальності.

В ході виконання дипломної роботи була розроблена інформаційна система оцінювання знань в області тестування програмного забезпечення.

Дана інформаційна система дозволяє створювати тестові завдання роботодавцями, проводити тестування претендентів, підбирати фахівців при працевлаштуванні та спілкуватися за допомогою чат-повідомлень в реальному часі.

Користуючись даною інформаційною системою, HR спеціалісти мають можливість аналізувати рівень знань при отриманні результатів тестів. Претенденти на вакантні посади мають можливість пройти тестування, отримати результати та фідбек від роботодавця.

Результатом проведеної дипломної роботи є створена інформаційна система оцінювання знань в області тестування програмного забезпечення.

Задачі щодо програмної розробки були вирішені за допомогою сучасних технологій, а саме фреймворків з відкритим кодом Vue.js та Laravel.

Об'єктом дослідження даної роботи є оцінювання знань претендентів на вакантні посади через їх тестування.

Предметом дослідження є інформаційна система для оцінювання знань в області тестування програмного забезпечення.

Практичне значення дипломної роботи полягає в розробленій інформаційній системі оцінювання знань, яка полегшує для роботодавців процес відбору претендентів для подальшої співбесіди на вакантні посади. Інформаційна система миттєво надає HR–менеджерам результати, чи має претендент необхідні знання на вакантну посаду, що збільшує швидкість знаходження потрібних співробітників.

Окрім оптимізації процесу найму створена інформаційна система оцінювання знань забезпечує покращення досвіду спілкування з кандидатами, порівняно з нетехнічними дискусіями, які занадто часто проводяться під час першої співбесіди.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Коронавірус змінив світ [Електронний ресурс] – Режим доступу до ресурсу: https://glavcom.ua/economics/finances/koronavirus-zminiv-svit-fahivtsi-nazvali-p-jat-prioritetiv-dlja-biznesu-majbutnogo-710477.html (дата звернення: 02.09.2020).

2. Працевлаштування в 2020: які труднощі чекають українців [Електронний ресурс] – Режим доступу до ресурсу: https://www.rbc.ua/rus/styler/trudoustroystvo-ukraine-2020-kakie-trudnosti-1584256591.html (дата звернення: 02.09.2020).

3. Труднощі пошуку: що заважає претендентам знаходити роботу мрії, в роботодавцям - потрібних кандидатів [Електронний ресурс] – Режим доступу до ресурсу: https://www.work.ua/ru/news/ukraine/816/ (дата звернення: 02.09.2020).

4. Стратегічне управління, управління портфелями, програмами та проектами. [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: http://pm.khpi.edu.ua/article/download/2413-3000.2016.1174.17/57227. (дата звернення: 25.10.2020).

5. Careers in Human Resource Management [Електронний ресурс]. – 2020. – https://www.shrm.org/membership/student-resources/pages/careersinhrm.aspx (дата звернення: 02.09.2020).

6. Aptitude test: What you can expect in an interview process [Електронний ресурс]. – 2017. – https://www.michaelpage.co.uk/advice/career-advice/job-interview-tips/aptitude-test-interview-process (дата звернення: 25.10.2020).

7. What Does a Human Resources Manager, Generalist, or Director Do? [Електронний ресурс]. – 2018. – https://www.thebalancecareers.com/what-does-a-human-resources-manager-do-1918551 (дата звернення: 25.10.2020).

8. How to Test Developers' Coding Skills Before Hiring [Електронний ресурс]. – 2018. – https://www.codingame.com/work/blog/tech-recruiting/test-developers-skills-before-hiring/ (дата звернення: 25.10.2020).

9. Interview Mocha [Електронний ресурс]. – 2020. – https://www.interviewmocha.com/pre-employment-testing/coding-tests (дата звернення: 25.10.2020).

10. Interview Mocha Review [Електронний ресурс]. – 2020. – https://reviews.financesonline.com/p/interview-mocha/ (дата звернення: 25.10.2020).

11. Aptitude Test Practice [Електронний ресурс]. – 2019. – https://www.assessmentday.co.uk (дата звернення: 25.10.2020).

12. Google Форм [Електронний ресурс]. – 2020. – https://www.google.com/intl/ru_ua/forms/about/ (дата звернення: 25.10.2020).

13. The Good and the Bad of Vue.js Framework Programming [Електронний ресурс]. – 2020. – https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/ (дата звернення: 25.10.2020).

14. Advantages and disadvantages of the laravel framework [Електронний ресурс]. – 2020. – https://www.popwebdesign.net/popart_blog/en/2020/03/advantages-and-disadvantages-of-the-laravel-framework/ (дата звернення: 25.10.2020).

15. Pros and Cons of Laravel [Електронний ресурс]. – 2020. – https://nimapinfotech.com/blog/pros-and-cons-of-laravel/ (дата звернення: 25.10.2020).

16. Батенко Л. П. Управління проектами: Навч. посібник. / Л. П. Батенко, О. А. Загородніх, В. Ліщинська. – Київ: КНЕУ, 2003. – 231 с. 62 Грей К.

17. Управление проектами: Практичекое руководство/Пер. с англ. / К. Грей, Э. Ларсон., 2003. – 528 с. 63

18. Локк Д. Основы Управления Проектами / Пер. с англ. / Д. Локк., 2004. – 253 с.

19. Что такое PDM-система [Електронний ресурс] – Режим доступу до ресурсу: http://pdm.in.ua/?page_id=142 (дата звернення: 02.09.2020)

20. Діаграма Ґанта [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Діаграма_Ґанта (дата звернення: 02.09.2020)

21. SMART [Електронний ресурс] // Вікіпедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SMART#cite_note-1 (дата звернення: 02.09.2020)

22. Оцінка життєздатності проекту [Електронний ресурс] – Режим доступу до ресурсу: https://pidruchniki.com/74414/ekonomika/otsinka_zhittyezdatnosti_proektu (дата звернення: 02.09.2020)

23. Воронин, А.В. Использование кластерного анализа для выбора локальных стратегий [Текст] / А.В. Воронин // Проблемы и перспективы управления экономикой и маркетингом в организации. – №1. – 2001.

24. Рідкокаша А.А., Голдер К.К. Основи систем штучного інтелекту. Навчальний посібник. Черкаси, "ВІДЛУННЯ – ПЛЮС", 2002. – 240 с.

25. Discovering Knowledge in Data: An Introduction to Data Mining, Jan 2005 – 190-201 с. 21. Chiang, C.L, (2003) Statistical methods of analysis, World Scientific. ISBN 981-238-310-7 – 274 с.

26. Bootstrap документація – [Електронний ресурс]. – Режим доступу: https://bootstrap-4.ru/docs/4.3.1/getting-started/introduction/ (дата звернення: 25.10.2020).

ДОДАТОК А. ПЛАНУВАННЯ РОБІТ

1 ІДЕНТИФІКАЦІЯ МЕТИ ІТ-ПРОЕКТУ

1.1. Загальна інформація про проект

У сучасному світі все більш актуальним стає питання пошуку роботи. Все частіше постає питання в чому ж камінь спотикання при пошуку роботи в області тестування програмного забезпечення і підходящих кандидатів. Виявилося, що думки роботодавців та шукачів багато в чому схожі, але є і протиріччя.

Так, наприклад, більшість претендентів стверджують, що головною проблемою при пошуку роботи є відсутність відповідних вакансій. У той час як роботодавці більшою мірою відзначають бажання потенційних кандидатів отримувати більше, ніж їм пропонують.

Виходить, ідеальне рішення проблем пошуку роботи та працівників - бути чесніше при складанні резюме і вакансій.

Дана інформаційна система – це платформа для HR та кандидатів, що знаходяться у пошуку роботи. Головна ідея сайту – перевірка кандидатів, що шукають роботу, на рівень знань для тієї чи іншої вакансії.

HR матиме можливість створювати тести за допомогою конструктора, робити публікації на сторінці та збирати статистику й результати з даних тестів.

1.2. Деталізація мети методом SMART

Для деталізації мети був використаний метод SMART, при якому відповівши на п'ять простих запитань, можна швидко розкрити поставлену мету ІТ-проекту та підвести до певного очікуваного результату від цього проекту.

Метод SMART є сучасним підходом до визначення цілей і постановки завдань [11]. Детальніше інформацію за кожним з пунктів розглянуто у таблиці А.1.1.

Таблиця А.1.1 – Конкретизація мети проекту за SMART методом

Тема	**Опис**
Specific (конкретна)	Розробка інформаційної системи, за допомогою якої HR спеціалісти матимуть можливість створювати тестування та аналізувати отриманні дані. Користувачі матимуть можливість пройти тестування, отримати результати та фідбек від роботодавця.
Measurable (вимірювана)	Прогрес та якість буде вимірюватися в кількості користувачах та сесії проведення користувача на сайті.
Achievable (досяжна)	Данна інформаційна система буде розроблятися з використанням HTML, CSS, JavaScript, Vue.js для розробки візуальної частини та PHP, MySQL, Laravel для функціональної.
Relevant (реалістична)	Для виконання проекту є всі необхідні технічні та програмні засоби.
Time-framed (обмежена у часі)	Для проекту визначений дедлайн, тому робота буде виконана вчасно, що підтверджується календарним планом.

1.3. Підготовка оціночного висновку

Інформаційна система оцінювання знань в області тестування програмного забезпечення буде ініціалізована по причині актуальності та релевантності обраної тематики. Для кандидатів пошуку роботи в області тестування програмного забезпечення – це перевірка знань з можливістю працевлаштування, а для роботодавців (HR-менеджерів) – тестування кандидатів на вакантні місця та швидкий аналіз отриманих результатів.

2 ПЛАНУВАННЯ ЗМІСТУ СТРУКТУРИ РОБІТ ІТ-ПРОЕКТУ

2.1. Планування змісту структури робіт ІТ-проекту (WBS)

Планування змісту структури робіт ІТ-проекту (WBS - Work Breakdown Structure) – являє собою графічне представлення проекту як пакету робіт в ієрархічному вигляді. WBS структура реалізується шляхом декомпозиції проекту на декілька рівнів[16].

Чітко визначена ієрархічна структура робіт (ІСР) може вплинути на успішність проекту. Невідповідність між ІСР, організаційною структурою і стилем управління проектного менеджера негативно впливає на ймовірність успішного завершення проекту. Існують різні підходи до побудови ІСР, незважаючи на те, що всі вони описують один і той самий проект.

Однак різні підходи до побудови ІСР вимагають різних організаційних структур і стилів управління під час реалізації проекту. Таким чином, розробник ІСР вже значно впливає на те, як проект повинен управлятися на самому ранньому етапі, іноді навіть не усвідомлюючи цього[17-18].

В даному випадку WBS структура має чотири основні фази:

- Ініціалізація
- Планування
- Реалізація
- Завершення

Загальна WBS структура представлена на рисунку А.2.1.

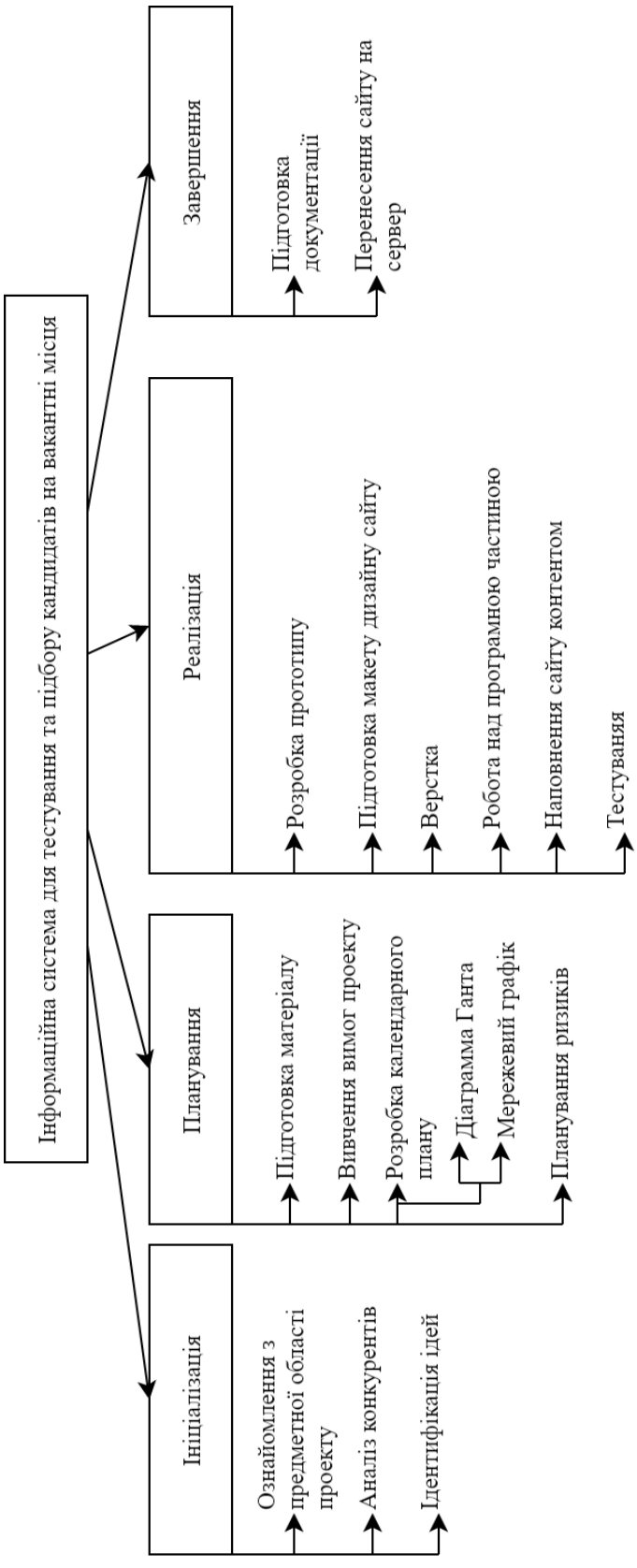

Рисунок А.2.1 – WBS структура проекту

2.2. Планування структури організації для впровадження готового проекту (OBS)

Структура розподілу організації або OBS (Organization Breakdown Structure) – це ієрархічна модель, що описує встановлені організаційні рамки для планування проектів, управління ресурсами, відстеження часу та витрат, розподілу витрат, звітування про доходи / прибуток та управління роботою [19].

Структура розподілу робіт (WBS) охоплює всі елементи проектів організовано. Розбиття великих, складних проектів на менші шматки проектів забезпечує кращу основу для організації та управління поточними та майбутніми проектами. WBS полегшує розподіл ресурсів, призначення завдань, вимірювання та контроль вартості проекту та виставлення рахунків. WBS використовується на початку проекту для визначення обсягу, визначення місць витрат і є відправною точкою для розробки планів проекту / діаграм Ганта[20].

Структура розподілу організації об’єднує подібні проектні заходи або «робочі пакети» та пов’язує їх зі структурою організації. OBS (також відома як Організаційна структурна структура) використовується для визначення відповідальності за управління проектами, звітування про витрати, виставлення рахунків, складання бюджету та контроль над проектами. OBS забезпечує організаційну, а не перспективну перспективу проекту. Ієрархічна структура OBS дозволяє агрегувати (зводити) інформацію про проект на вищі рівні. Коли визначені обов'язки проекту та призначена робота, OBS та WBS з'єднуються, забезпечуючи можливість потужної аналітики для вимірювання продуктивності проекту та робочої сили на дуже високому рівні (приклад продуктивності бізнес-підрозділу) або до деталей (приклад роботи користувача над завдання).

Загальна OBS структура представлена на рисунку А.2.2.

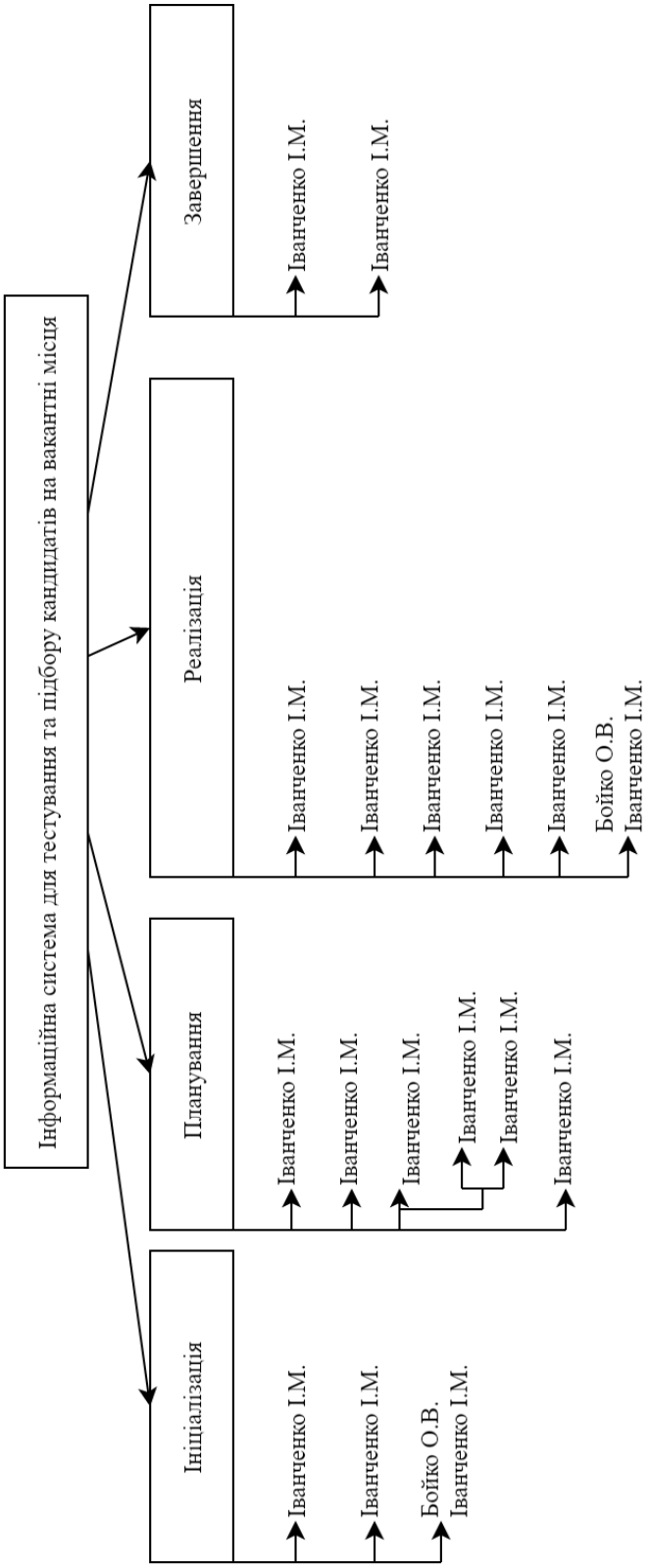

Рисунок А.2.2 – OBS структура проекту

3 ПОБУДОВА КАЛЕНДАРНОГО ГРАФІКУ ВИКОНАННЯ IT – ПРОЕКТУ

3.1. Побудова календарного графіку виконання IT-проекту

Базуючись Базуючись на сформованій інформації, а саме WBS та OBS, побудуємо календарний графік виконання IT-проекту.

Діаграма Ганта — діаграма, яка використовується для ілюстрації плану, графіка робіт за будь-яким проектом [20]. Діаграма Ганта є одним з засобів планування та управління проектами. Діаграма Ганта також являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі.

Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями. Діаграма може використовуватися для представлення поточного стану виконання робіт [20].

Розглянемо створену діаграму Ганта до заданої інформаційної системи (рис.А.3.1).

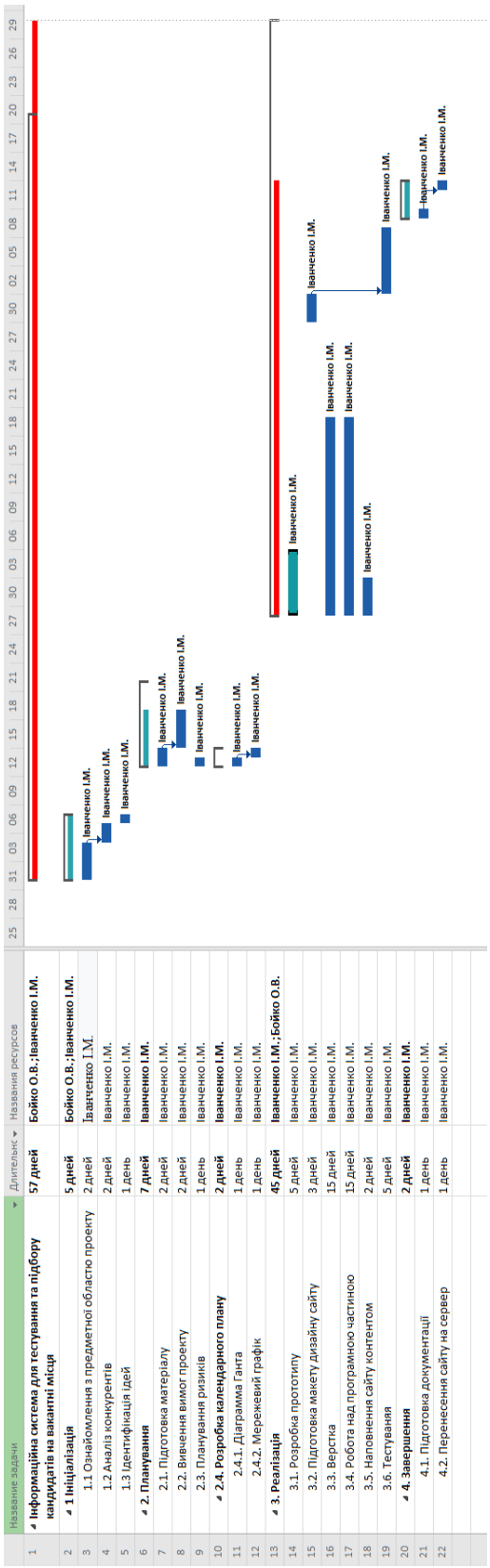

Рисунок А.3.1 – Діаграма Ганта

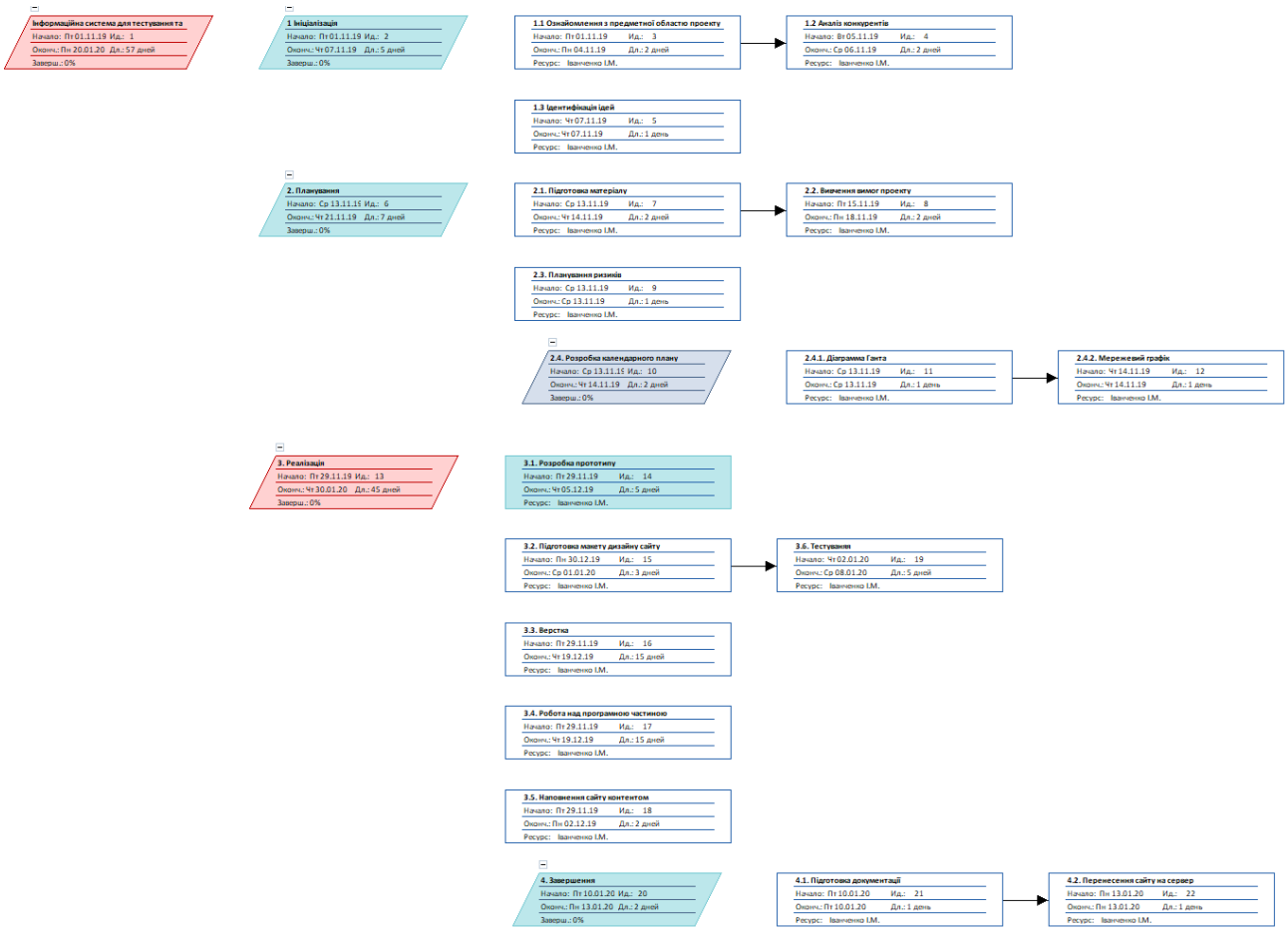

Рисунок А.3.2 – Мережевий графік проекту

3.2. Розрахунок бюджету IT-проекту

Бюджет продукту IT-проекту – це кошторис продукту проекту, розподілений в часі на основі календарного плану реалізації робіт або за окремими WBS елементами [21]. В основу розроблення бюджету продукту IT-проекту покладають перелік робіт з проекту та необхідні витрати на їх виконання. Бюджетну вартість кожної з робіт розраховують на підставі кількості використаного ресурсу, його вартості та тривалості використання. використання.

Розрахунок бюджету проекту:

загальні витрати за проект представлені на рисунку А.3.3.

Название задачи	Затраты
Інформаційна система для тестування та підбору кандидатів на вакантні місця	**8 700,00 ₴**
1 Ініціалізація	**0,00 ₴**
1.1 Ознайомлення з предметної областю проекту	0,00 ₴
1.2 Аналіз конкурентів	0,00 ₴
1.3 Ідентифікація ідей	0,00 ₴
2. Планування	**200,00 ₴**
2.1. Підготовка матеріалу	0,00 ₴
2.2. Вивчення вимог проекту	0,00 ₴
2.3. Планування ризиків	0,00 ₴
2.4. Розробка календарного плану	**200,00 ₴**
2.4.1. Діаграмма Ганта	100,00 ₴
2.4.2. Мережевий графік	100,00 ₴
3. Реалізація	**8 500,00 ₴**
3.1. Розробка прототипу	200,00 ₴
3.2. Підготовка макету дизайну сайту	800,00 ₴
3.3. Верстка	2 000,00 ₴
3.4. Робота над програмною частиною	3 000,00 ₴
3.5. Наповнення сайту контентом	300,00 ₴
3.6. Тестуваняя	1 500,00 ₴
4. Завершення	**700,00 ₴**
4.1. Підготовка документації	200,00 ₴
4.2. Перенесення сайту на сервер	500,00 ₴

Рисунок А.3.3 – Затрати за проект

4 ПЛАНУВАННЯ РИЗИКІВ ПРОЕКТУ

Ретельне і детальне планування управління ризиками спрямовано на підвищення вірогідності досягнення цілей проекту. Даний процес повинен бути завершений на ранній стадії планування [22].

Якісний аналіз ризиків включає розстановку пріоритетів для ідентифікованих ризиків, результати якої використовуються потім у ході кількісного аналізу ризиків і планування реагування на ризики.

У ході якісного оцінювання використовується нечислова шкала вірогідності, наприклад:

1 - дуже слабкий вплив;

2 - слабкий вплив;

3 - середній вплив;

4 - сильний вплив;

5 - дуже сильний вплив.

Оцінку ризиків проводять за допомогою оцінки вірогідності і наслідків. Один з можливих прикладів подібної матриці, в якій показані тільки оцінки наслідків, представлений у таблиці (табл. А.4.1).

Таблиця А.4.1 – Матриця оцінки наслідків

Ціль проекту	**Дуже слабкий вплив – 0, 05**	**Слабкий вплив - 0,01**	**Середній вплив - 0,2**	**Сильний вплив - 0,4**	**Дуже сильний вплив - 0,8**
Вартість	Несуттєве збільшення бюджету	Збільшення бюджету до 10%	Збільшення бюджету на 10-20%	Збільшення бюджету на 20-30%	Збільшення бюджету більше, ніж на 40%
Терміни	Несуттєве збільшення календарного плану	Порушення календарного плану не більше ніж на 5%	Порушення календарного плану на 5-10%	Порушення календарного плану на 10-20%	Порушення календарного плану більш ніж на 20%
Якість	Несуттєве зниження якості	Суттєве зниження якості	Зниження якості потребує узгодження з замовником	Зниження якості неприйнятне для замовника	Результат проекту повністю даремний

Існують три стратегії реагування на появу негативних ризиків, до яких відносяться:

– Ухилення від ризику передбачає змінення плану управління проектом таким чином, щоб виключити ризик, убезпечити цілі ІТ-проекту від наслідків ризику.

– Передача ризику має на увазі перекладення негативних наслідків на третю сторону. Передача ризику просто переносить відповідальність за його управління іншій стороні, але ризик при цьому не зникає. Передача ризику є найбільш ефективною у відношенні фінансових ризиків. Передача ризику практично завжди передбачає виплату премії за ризик стороні, яка прийняла ризик на себе. В якості інструментів передачі ризиків використовуються страховки, гарантійні зобов'язання і т.п. У ряді випадків витрати на ризики можуть перекладатися на покупця або продавця, що оговорюється у контракті.

– Зниження ризику передбачає зниження вірогідності наслідків негативної ризикованої події до прийнятних границь. Прийняття попереджувальних заходів зі зниження вірогідності настання ризику або його наслідків часто буває більш ефективним, ніж зусилля з усунення негативних наслідків, що здійснюються після настання події ризику.

ДОДАТОК Б. КОД РЕАЛІЗАЗІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2014_10_12_000000_create_users_table.php

Міграція створення таблиці користувачів.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('email')->unique();
            $table->string('password');
            $table->foreignId('account_role_id');
            $table->foreignId('country_id');
            $table->string('name');
            $table->string('surname');
            $table->boolean('sex');
            $table->string('date');
            $table->text('description')->nullable();
            $table->string('photo')->default('/img/no-image.png');
            $table->string('company')->nullable();
            $table->string('phone')->nullable();
            $table->rememberToken();
            $table->timestamps();
        });

        Schema::table('users', function (Blueprint $table) {
            $table->index('country_id');
            $table->foreign('country_id')->references('id')->on('country');
        });
```

```
        Schema::table('users', function (Blueprint $table) {
            $table->index('account_role_id');
            $table->foreign('account_role_id')->references('id')->on('account_role');
        });
    }
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

2020_11_09_160729_create_tests_table.php

Міграція створення таблиці тестів.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateTestsTable extends Migration
{
    public function up()
    {
        Schema::create('tests', function (Blueprint $table) {
            $table->id();
            $table->string('file')->nullable();
            $table->string('title');
            $table->text('description');
            $table->time('time', 0);
            $table->foreignId('account_id');
            $table->string('type')->default('tests');
            $table->timestamps();
        });

        Schema::table('tests', function (Blueprint $table) {
            $table->index('account_id');
            $table->foreign('account_id')->references('id')->on('users')->onDelete('cascade');
        });
```

```
    }
    public function down()
    {
        Schema::dropIfExists('tests');
    }
}
```

2020_11_09_161530_create_test_questions_table.php

Міграція створення таблиці завдань до тестів.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateTestQuestionsTable extends Migration
{
    public function up()
    {
        Schema::create('test_questions', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->text('description');
            $table->string('type');
            $table->text('answer');
            $table->foreignId('test_id');
            $table->boolean('ready')->default(0);
            $table->timestamps();
        });

        Schema::table('test_questions', function (Blueprint $table) {
            $table->index('test_id');
            $table->foreign('test_id')->references('id')->on('tests')->onDelete('cascade');
        });
    }
    public function down()
    {
        Schema::dropIfExists('test_questions');
    }
```

```
}
```

2020_11_09_161221_create_test_results_table.php

Міграція створення таблиці результатів пройдених тестів.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateTestResultsTable extends Migration
{
    public function up()
    {
        Schema::create('test_results', function (Blueprint $table) {
            $table->id();
            $table->foreignId('account_id');
            $table->foreignId('test_id');
            $table->integer('procent');
            $table->integer('count_true');
            $table->integer('count');
            $table->time('time', 0);
            $table->timestamps();
        });

        Schema::table('test_results', function (Blueprint $table) {
            $table->index('account_id');
            $table->foreign('account_id')->references('id')->on('users')->onDelete('cascade');
        });

        Schema::table('test_results', function (Blueprint $table) {
            $table->index('test_id');
            $table->foreign('test_id')->references('id')->on('tests')->onDelete('cascade');
        });
    }
    public function down()
    {
        Schema::dropIfExists('test_results');
```

```
    }
}
```

User.php

Модель таблиці користувачів.

```
<?php

namespace App\Models;

use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Laravel\Passport\HasApiTokens;

class User extends Authenticatable
{
    use Notifiable, HasApiTokens;

    protected $table = 'users';

    protected $fillable = [
        'email',
        'password',
        'account_role_id',
        'country_id',
        'name',
        'surname',
        'sex',
        'date',
        'description',
        'photo',
        'company',
        'phone'
    ];

    protected $hidden = [
        'password'
    ];

    function country() {
```

```
        return $this->belongsTo('App\Models\Country', 'country_id');
    }

    function role() {
        return $this->belongsTo('App\Models\AccountRole', 'account_role_id');
    }

    function news() {
        return $this->hasMany('App\Models\News', 'account_id');
    }

    function socials() {
        return $this->HasMany('App\Models\AccountSocial', 'account_id');
    }

    function tests() {
        return $this->HasMany('App\Models\Tests', 'account_id');
    }

    function testResults() {
        return $this->HasMany('App\Models\TestResult', 'account_id');
    }

    function followers() {
        return $this->HasMany('App\Models\Followers', 'account_follower_id');
    }

    function myFollowers() {
        return $this->HasMany('App\Models\Followers', 'account_id');
    }

    function notifications() {
        return $this->HasMany('App\Models\Notifications', 'account_id');
    }
}
```

Tests.php

Модель таблиці тестів.

```
<?php
```

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Tests extends Model
{
    use HasFactory;

    protected $table = 'tests';

    protected $fillable = [
        'file',
        'title',
        'description',
        'time',
        'account_id',
        'type'
    ];

    function answer() {
        return $this->hasMany('App\Models\TestQuestions', 'test_id');
    }

    function user() {
        return $this->belongsTo('App\Models\User', 'account_id');
    }

    function results() {
        return $this->hasMany('App\Models\TestResult', 'test_id');
    }

    function comments() {
        return $this->hasMany('App\Models\Comments', 'post_id');
    }
}
```

TestQuestions.php

Модель таблиці завдань до тестів.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class TestQuestions extends Model
{
    use HasFactory;

    protected $table = 'test_questions';

    protected $fillable = [
        'title',
        'description',
        'type',
        'answer',
        'test_id',
        'ready'
    ];
}
```

TestResult.php

Модель таблиці результатів пройдених тестів.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class TestResult extends Model
{
    use HasFactory;

    protected $table = 'test_results';

    protected $fillable = [
```

```
        'account_id',
        'test_id',
        'procent',
        'time',
        'count',
        'count_true'
    ];

    function test() {
        return $this->belongsTo('App\Models\Tests', 'test_id');
    }

    function user() {
        return $this->belongsTo('App\Models\User', 'account_id');
    }
}
```

AuthController.php

Контроллер авторизації користувачів.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;

use App\Models\User;

class AuthController extends Controller
{
    // register
    function register(Request $request) {
        $request->validate([
            'email' => 'required|string|email|unique:users',
            'password' => 'required|string'
        ]);
        $user = new User();
        $data = $request->all();
```

```
        $data["password"] = Hash::make($request->password);
        $user->create($data);
        $credentials = request(['email', 'password']);
        if(!Auth::attempt($credentials)) {
            return response()->json(['message' => 'Unauthorized'], 401);
        }
        $authUser = Auth::user();
        $tokenResult = $authUser->createToken('Personal Access Token');
        $token = $tokenResult->token;
        $token->save();
        return response()->json([
            'access_token' => 'Bearer '.$tokenResult->accessToken,
            'user' => $authUser
        ]);
    }

    // login
    function login(Request $request) {
        if(Auth::attempt(['email' => $request->email, 'password' => $request->password, 'account_role_id' => 1])
|| Auth::attempt(['email' => $request->email, 'password' => $request->password, 'account_role_id' => 2])) {
            $user = Auth::user();
            $tokenResult = $user->createToken('Personal Access Token');
            $token = $tokenResult->token;
            if($request->remember_me) {
                $token->expires_at = Carbon::now()->addWeeks(1);
            }
            $token->save();
            return response()->json([
                'access_token' => 'Bearer '.$tokenResult->accessToken,
                'user' => $user
            ]);
        } else {
            return response()->json(['message' => 'Unauthorized'], 401);
        }
    }

    // loginAdmin
    function loginAdmin(Request $request) {
```

```
        if(Auth::attempt(['email' => $request->email, 'password' => $request->password, 'account_role_id' => 3]))
{
            $user = $request->user();
            $tokenResult = $user->createToken('Personal Access Token');
            $token = $tokenResult->token;
            $token->save();
            return response()->json([
                'access_token' => 'Bearer '.$tokenResult->accessToken
            ]);
        } else {
            return response()->json(['message' => 'Unauthorized'], 401);
        }
    }
}
```

UserController.php

Контролер для роботи з користувачами.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Carbon\Carbon;

use App\Models\Country;
use App\Models\User;
use App\Models\Tests;
use App\Models\News;
use App\Models\Followers;
use App\Models\Comments;
use App\Models\Notifications;

class UserController extends Controller
{
    protected $fileStorage = "userfiles/";

    function delUser($id) {
        User::find($id)->delete();
```

```
    return response('ok', 200);
}

// users
function users() {
    $data = User::with('country', 'role')->where('account_role_id', '!=', 3)->get();
    return response()->json($data);
}

// search
function search(Request $request) {
    $news = News::select('title', 'description', 'type', 'id')
        ->where('title', 'like', "%".$request->search."%")
        ->orWhere('description', 'like', "%".$request->search."%");
    $test = Tests::select('title', 'description', 'type', 'id')
        ->where('title', 'like', "%".$request->search."%")
        ->orWhere('description', 'like', "%".$request->search."%");
    $result = $news->union($test)->get();
    return response()->json($result);
}

// updateUser
function updateUser(Request $request) {
    $id = Auth::id();
    $data = $request->all();
    if(isset($data['newPassword'])) {
        $user = Auth::user();
        if (Hash::check($data['oldPassword'], $user->password)) {
            $data["password"] = Hash::make($request->newPassword);
        } else {
            return response('error', 401);
        }
    }
    User::find($id)->update($data);
    return response()->json(User::find($id));
}

// country
function country() {
```

```
    $data = Country::get();
    return response()->json($data);
}

// cabinet
function cabinet() {
    $data = User::with(
        'country',
        'role',
        'tests',
        'testResults',
        'testResults.test',
        'testResults.test.user',
        'myFollowers.user.followers',
        'followers.userFollower.followers',
        'socials.social',
        'notifications',
        'news.user',
        'news.comments'
    )->find(Auth::id());
    $mean = 0;
    foreach ($data['testResults'] as $key => $item) {
        $mean += $item->procent;
        $item->date = Carbon::parse($item->created_at)->format('d.m.Y H:i');
    }
    foreach ($data['news'] as $key => $item) {
        $item->date = Carbon::parse($item->created_at)->format('d.m.Y H:i');
    }
    if($mean > 0) {
        $data['mean'] = $mean / count($data['testResults']);
    }
    return response()->json($data);
}

// userId
function userId($id) {
    $authId = Auth::id();
    $data = User::with(
        'country',
```

```
            'role',
            'news.user',
            'socials.social',
            'testResults',
            'myFollowers.user.followers',
            'followers.userFollower.followers',
            'socials',
            'news.user',
            'news.comments',
            'tests'
        )->find($id);
        $mean = 0;
        foreach ($data['news'] as $key => $item) {
            $item->date = Carbon::parse($item->created_at)->format('d.m.Y H:i');
        }
        foreach ($data['testResults'] as $key => $item) {
            $mean += $item->procent;
            $item->date = Carbon::parse($item->created_at)->format('d.m.Y H:i');
        }
        if($mean > 0) {
            $data['mean'] = $mean / count($data['testResults']);
        }
        return response()->json($data);
    }
    // usersResults
    function usersResults() {
        $data = Tests::with('results.user')->where('account_id', Auth::id())->get();
        foreach ($data as $key => $item) {
            foreach ($item['results'] as $k => $v) {
                $v->date = Carbon::parse($v->created_at)->format('d.m.Y H:i');
            }
        }
        return response()->json($data);
    }

    // postFollower
    function postFollower($id) {
        $model = new Followers();
        if(!Followers::where('account_id', Auth::id())->where('account_follower_id', $id)->exists()) {
```

```
            $model->create([
                "account_id" => Auth::id(),
                "account_follower_id" => $id
            ]);
        }
        return response('ok', 200);
    }
    // postFollower
    function delFollower($id) {
        Followers::where('account_id', Auth::id())->where('account_follower_id', $id)->delete();
        return response('ok', 200);
    }

    // comments
    function postComment(Request $request, $id) {
        $model = new Comments();
        $response = $model->create([
            "account_id" => Auth::id(),
            "post_id" => $id,
            "text" => $request->comment
        ]);
        if($request->type == "news") {
            $post = News::find($id);
            $userId = $post['account_id'];
            $title = $post['title'];
            $notification = new Notifications();
            $textNotification = "Користувач <a href=\"/user/" . Auth::id() . "\">" . Auth::user()['name'] . " " .
Auth::user()['surname'] . "</a> залишив коментар до Вашої публікації <a href=\"/news/" . $id . "\">" . $title .
"</a>";
            $notification->create([
                "title" => "У Вас новий коментар",
                "account_id" => $userId,
                "text" => $textNotification
            ]);
        }
        if($request->type == "test") {
            $post = Tests::find($id);
            $userId = $post['account_id'];
            $title = $post['title'];
```

```
            $notification = new Notifications();
            $textNotification = "Користувач <a href=\"/user/" . Auth::id() . "\">" . Auth::user()['name'] . " " .
Auth::user()['surname'] . "</a> залишив коментар до Вашого тесту <a href=\"/news/" . $id . "\">" . $title .
"</a>";
            $notification->create([
                "title" => "У Вас новий коментар",
                "account_id" => $userId,
                "text" => $textNotification
            ]);
        }
        return response()->json($response);
    }
    function delComment($id) {
        Comments::find($id)->delete();
        return response('ok', 200);
    }

    function updateNotification($id) {
        Notifications::where('id', $id)->update([
            'watch' => true
        ]);
        return response('ok', 200);
    }

    // img Account
    function img(Request $request) {
        if(isset($request['photo'])) {
            $arr = [];
            if($request['photo']) {
                $file = uniqid().'_photo_min.png';
                $uploadfile = $this->fileStorage . $request['id'] . '/' . $file;

                $img = str_replace('', '', $request['photo']);
                $img = str_replace(' ', '+', $img);
                $fileData = base64_decode($img);
                file_put_contents(public_path().'/'.$uploadfile, $fileData);

                $arr['status'] = 'success';
                $arr['path_mini'] = 'http://'.$_SERVER['HTTP_HOST'].'/'.$uploadfile;
```

```
            $arr['file_mini'] = $file;
        }
    }
    else {
        if(!file_exists("userfiles/".$request['id'])) {
            mkdir($this->fileStorage.$request['id']);
        }
        $uploadfile = $this->fileStorage. $request['id'] . '/' . uniqid().'_photo_original.png';
        $arr = array();
        if (move_uploaded_file($_FILES['file']['tmp_name'], public_path().'/'.$uploadfile)) {
            $arr['status'] = 'success';
            $arr['path_max'] = 'http://'.$_SERVER['HTTP_HOST'].'/'.$uploadfile;
            $arr['file_max'] = $_FILES['file']['name'];
        } else {
          $arr['status'] = 'fail';
        }
    }
    header('Content-type: application/json');
    return response()->json($arr);
  }
}
```

TestController.php

Котроллер для роботи с тестами.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Carbon\Carbon;

use App\Models\Tests;
use App\Models\TestQuestions;
use App\Models\TestResult;
use App\Models\Notifications;

class TestController extends Controller
{
```

```
protected $fileStorage = "/userfiles/";

// get
function get() {
    $data = Tests::with('user', 'comments.user', 'answer')->orderBy('created_at', 'desc')->get();
    foreach ($data as $key => $item) {
        $item->date = Carbon::parse($item->created_at)->format('d.m.Y H:i');
    }
    return response()->json($data);
}
// getId
function getId($id) {
    $data = Tests::with('user', 'answer', 'comments.user')->find($id);
    $data['date'] = Carbon::parse($data['created_at'])->format('d.m.Y H:i');
    foreach ($data['answer'] as $key => $value) {
        $value['answer'] = json_decode($value['answer']);
    }
    return response()->json($data);
}

function getId2($id) {
    $data = Tests::with('answer', 'comments.user')->find($id);
    $userData = Tests::with('answer')->find($id);
    $data['date'] = Carbon::parse($data['created_at'])->format('d.m.Y H:i');
    foreach ($data['answer'] as $key => $value) {
        $value['answer'] = json_decode($value['answer']);
    }
    foreach ($userData['answer'] as $key => $value) {
        $value['answer'] = json_decode($value['answer']);
    }
    return response()->json([
        'data' => $data,
        'userData' => $userData['answer']
    ]);
}
// post
function post(Request $request) {
    $model = new Tests();
    $data = $request->all();
```

```
    $data['account_id'] = Auth::id();
    $data['time'] = $request->hours . ":" . $request->minutes . ":00";
    if(isset($request['file']) && $request['file'] != "null") {
        $puth = $this->fileStorage.Auth::id();
        $name =  $puth."/".uniqid().'.'.$request['file']->getClientOriginalExtension();
        $request['file']->move(public_path().$puth, $name);
        $data['file'] = $name;
    }
    $response = $model->create($data);
    return response()->json($response);
}
// update
function updateTest(Request $request, $id) {
    $model = Tests::find($id);
    $data = $request->all();
    $data['time'] = "00:" . $request->hours . ":" . $request->minutes;
    if(isset($request['file']) && $request['file'] != "null") {
        $puth = $this->fileStorage.Auth::id();
        $name = $puth."/".uniqid().'.'.$request['file']->getClientOriginalExtension();
        $request['file']->move(public_path().$puth, $name);
        $data['file'] = $name;
    }
    $model->update($data);
    return response()->json($model);
}
// delTest
function delTest($id) {
    $model = Tests::find($id)->delete();
    return response('ok', 200);
}
// postQuestion
function postQuestion(Request $request, $id) {
    $model = new TestQuestions();
    $data = $request->all();
    $data['test_id'] = $id;
    $data['answer'] = json_encode($request[$request->type.'_answer']);
    $response = $model->create($data);
    $response['answer'] = json_decode($response['answer']);
    return response()->json($response);
```

```
    }
    // updateQuestion
    function updateQuestion(Request $request, $id) {
        $model = TestQuestions::find($request->id);
        $data = $request->all();
        $data['answer'] = json_encode($request[$request->type.'_answer']);
        $model->update($data);
        $model['answer'] = json_decode($model['answer']);
        return response()->json($model);
    }
    // deleteQuestion
    function deleteQuestion($id) {
        $model = TestQuestions::find($id)->delete();
        return response('ok', 200);
    }
    // saveResult
    function saveResult(Request $request, $id) {
        $model = new TestResult();
        $data = $request->all();
        $data['test_id'] = $id;
        $data['account_id'] = Auth::id();
        $response = $model->create($data);
        $response['date'] = Carbon::parse($response->created_at)->format('H:i:s');

        $test = Tests::find($id);
        $notifications = new Notifications();
        $textMessage = "Користувач <a href=\"/user/" . Auth::id() . "\">" . Auth::user()['name'] . " " .
Auth::user()['surname'] . "</a> виконав Ваш тест <a href=\"/news/" . $id . "\">" . $test['title'] . "</a> з
результатом " . $request->procent . "%.";
        $notifications->create([
            'account_id' => $test['account_id'],
            'title' => "Користувач виконав Ваш тест",
            'text' => $textMessage
        ]);
        return response()->json($response);
    }
}
```

ChatController.php

Контроллер роботи чату.

```
<?php

namespace App\Http\Controllers;

use App\Events\MessageSentEvent;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Carbon\Carbon;
use App\Models\Chats;
use App\Models\Messages;

class ChatController extends Controller
{
   function postChat(Request $request) {
      $model1 = Chats::where("account_id_1", $request->account_id)->where("account_id_2", Auth::id());
      $model2 = Chats::where("account_id_1", Auth::id())->where("account_id_2", $request->account_id);
      if($model1->exists() || $model2->exists()) {
         $message = new Messages();
         $id1 = $model1->first()['id'];
         $id2 = $model2->first()['id'];
         $message->create([
            "text" => $request->message,
            "account_id" => $request->account_id,
            "chat_id" => $model1->exists() ? $id1 : $id2
         ]);
      } else {
         $chat = new Chats();
         $response = $chat->create([
            "account_id_1" => Auth::id(),
            "account_id_2" => $request->account_id,
         ]);
         $message = new Messages();
         $message->create([
            "text" => $request->message,
            "account_id" => Auth::id(),
            "chat_id" => $response['id']
         ]);
      }
   }
```

```
    function getChats() {
        $chats = Chats::with('messages.user')->where("account_id_1", Auth::id())->orWhere("account_id_2",
Auth::id())->get();
        foreach ($chats as $key => $chat) {
            $chat->date = Carbon::parse($chat->created_at)->format('d.m.Y H:i');
        }
        return response()->json($chats);
    }

    function postMessages(Request $request, $chat_id) {
        $user = Auth::user();

        $message = new Messages();
        $response = $message->create([
            "text" => $request->message,
            "account_id" => Auth::id(),
            "chat_id" => $chat_id
        ]);

        broadcast(new MessageSentEvent($response, $user, $chat_id))->toOthers();

        return response()->json([
            'message' => $response,
            'account' => $user
        ]);
    }

    function getMessages($chat_id) {
        $result = [];
        $data = Messages::with('user')->where('chat_id', $chat_id)->get();
        foreach ($data as $key => $chat) {
            array_push($result, [
                'message' => $chat,
                'account' => $chat['user']
            ]);
        }
        return response()->json($result);
    }
```

```
}
```

MessageSentEvent.php

Прослуховувач подів для роботи чату в реальному часі.

```
<?php

namespace App\Events;

use App\Models\Messages;
use App\Models\User;
use Illuminate\Broadcasting\Channel;
use Illuminate\Broadcasting\InteractsWithSockets;
use Illuminate\Broadcasting\PresenceChannel;
use Illuminate\Broadcasting\PrivateChannel;
use Illuminate\Contracts\Broadcasting\ShouldBroadcast;
use Illuminate\Foundation\Events\Dispatchable;
use Illuminate\Queue\SerializesModels;
use Carbon\Carbon;

class MessageSentEvent implements ShouldBroadcast
{
    use Dispatchable, InteractsWithSockets, SerializesModels;

    public $message;
    public $account;
    public $chat_id;

    public function __construct(Messages $message, User $user, $chat_id)
    {
        $this->message = $message;
        $this->account = $user;
        $this->chat_id = $chat_id;
    }

    public function broadcastOn()
    {
        return new Channel('chat-' . $this->chat_id);
    }
}
```

/cabinet/ Index.vue

Компонент кабінету користувача.

```
<template>
  <b-container>
    <b-modal v-model="modal" id="modal-center" centered title="Створення нового тесту"  hide-footer>
      <div class="modal-input">
        <div class="head-title mb-1">
          Назва тесту
        </div>
        <input
          v-model="test.title"
          type="text"
          class="w-100"
          name="title"
          v-validate="'required'"
          :class="errors.has('title') ? 'input-has-error' : ''"
        >
        <div class="head-title mt-2 mb-1">
          Опис
        </div>
        <textarea
          v-model="test.description"
          class="w-100"
          name="description"
          v-validate="'required'"
          :class="errors.has('description') ? 'input-has-error' : ''"
        ></textarea>
        <div class="head-title mt-2 mb-1">
          Час виконання
        </div>
        <div class="time">
          <input
            v-model="test.hours"
            type="number"
            name="hours"
            v-validate="'required'"
            :class="errors.has('hours') ? 'input-has-error' : ''"
          >год. <input
```

```
                    v-model="test.minutes"
                    type="number"
                    name="minutes"
                    v-validate="'required'"
                    :class="errors.has('minutes') ? 'input-has-error' : ""
                  > хв.
                </div>
                <div class="head-title mt-2 mb-1">
                  Фото
                </div>
                <label for="file" class="file">
                  <b-icon class="icon" icon="image" font-scale="1.5"></b-icon> {{ fileName ? fileName : "Файл не обрано" }}
                </label>
                <b-form-file
                  @change="previewFile($event)"
                  ref="file"
                  id="file"
                  style="display:none"
                  v-model="test.file"
                ></b-form-file>
              </div>
              <hr>
              <div class="text-right">
                <b-button class="mt-3" @click="postTest">Додати</b-button>
              </div>
            </b-modal>
            <b-row>
              <b-col md="8" lg="8" xl="8" xs="12" class="user-block block mr-5">
                <User :user="user"></User>
              </b-col>
              <b-col class="block">
                <Messages :notifications="user.notifications" @update="fetchData()"></Messages>
              </b-col>
            </b-row>
            <b-row class="mt-5">
              <b-col md="3" lg="3" xl="3" xs="12" class="block mr-5 px-0 menu">
                <Menu></Menu>
              </b-col>
```

```
        <b-col class="block p-4">
          <div class="title-page">Мої тести
            <b-icon v-if="$store.getters.authUser.account_role_id == 1" @click="modal = true" font-
scale="0.8" class="ml-2" icon="plus-circle"></b-icon>
            <router-link
              v-if="$store.getters.authUser.account_role_id == 2"
              class="all-tests"
              to="/cabinet/result-tests"
            >Список всіх тестів</router-link>
          </div>
          <hr>
          <div v-if="$store.getters.authUser.account_role_id == 1">
            <div class="text-center my-4" v-if="user.tests.length == 0">
              Тести відсутні
            </div>
            <b-row cols="12" v-for="(item, index) in user.tests" :key="index" class="mb-3">
              <b-col class="text-center" cols="1">
                {{ index + 1 }}
              </b-col>
              <b-col>
                <router-link :to="'/cabinet/edit-test/'+item.id">{{ item.title }}</router-link>
              </b-col>
              <b-col cols="1">
                <b-icon @click="deleteTest(item)" icon="trash-fill"></b-icon>
              </b-col>
            </b-row>
          </div>
          <b-row v-if="$store.getters.authUser.account_role_id == 2">
            <b-col md="6" lg="6" xl="6" xs="12" v-for="(item, index) in user.test_results" :key="index"
class="px-5 py-3">
              <ItemTestResult :item="item"></ItemTestResult>
            </b-col>
          </b-row>
        </b-col>
      </b-row>
    </b-container>
</template>
<script>
import ItemTestResult from '../../../components/site/cabinet/ItemTestResult';
```

```
import User from '../../../components/site/cabinet/User';
import Messages from '../../../components/site/cabinet/Messages';
import Menu from '../../../components/site/cabinet/Menu';
export default {
  components: {
    User,
    Messages,
    Menu,
    ItemTestResult
  },
  data() {
    return {
      fileName: null,
      modal: false,
      test: {
        title: "",
        description: "",
        hours: 0,
        minutes: 0,
        file: null
      },
      user: {
        name: "",
        surname: "",
        photo: "",
        country: {
          title: ""
        },
        role: {
          title: ""
        },
        tests: [],
        test_results: [],
        notifications: []
      }
    }
  },
  created() {
    this.fetchData();
```

```
},
methods: {
    previewFile(event) {
        this.fileName = event.target.files[0].name;
    },
    fetchData() {
        axios.get('/api/cabinet')
        .then((response) => {
            this.user = response.data;
        })
    },
    postTest() {
        this.$validator.validateAll().then((result) => {
            if (!result) {
                return;
            } else {
                let formData = new FormData();
                formData.append('title', this.test.title);
                formData.append('description', this.test.description);
                formData.append('hours', this.test.hours);
                formData.append('minutes', this.test.minutes);
                formData.append('file', this.test.file);
                axios.post('/api/test', formData, {
                    headers: {
                        'Content-Type': 'multipart/form-data'
                    }
                })
                .then((response) => {
                    this.modal = false;
                    this.user.tests.push(response.data);
                    this.$validator.reset();
                    this.$router.push("/cabinet/edit-test/"+response.data.id);
                })
            }
        })
    },
    deleteTest(item) {
        console.log(item)
    }
```

```
    }
}
</script>
<style lang="css" scoped>
    @media screen and (max-width: 600px) {
        .user-block {
            margin: 0 0 30px 0 !important;
        }
        .menu {
            margin: 0 0 30px 0 !important;
        }
    }
    .modal-input .time input {
        width: 60px;
        margin: 0 5px;
    }
    .modal-input input, .modal-input textarea {
        border: 1px solid silver;
        border-radius: 5px;
        padding: 5px 10px;
    }
    a.all-tests {
        float: right;
        font-size: 12px;
        color: #ffffff;
        background: #0DA1A1;
        border-radius: 31px;
        padding: 0 15px;
    }
</style>
```

Test.vue

Компонент проходження тесту.

```
<template>
    <b-container class="block px-5 py-4">
        <b-modal v-model="resultModal" no-close-on-backdrop id="modal-tall" title="Ваш результат">
            <template #modal-header>
                <h5>Ваш результат</h5>
            </template>
```

```
<b-row class="px-4 mb-3">
  <b-col cols="3">
    <div id="circleProgress" class="progressbar-js-circle rounded">
      <svg class="round-block" viewBox="0 0 100 100">
        <path d="M 50,50 m 0,-48 a 48,48 0 1 1 0,96 a 48,48 0 1 1 0,-96" stroke="#eee" stroke-
width="4" fill-opacity="0"></path>
        <path class="round" d="M 50,50 m 0,-48 a 48,48 0 1 1 0,96 a 48,48 0 1 1 0,-96"
stroke="rgb(98,194,205)" stroke-width="4" fill-opacity="0" :style="'stroke-dashoffset: '+
progress(result.procent)"></path>
      </svg>
      <div class="progressbar-text">{{ result.procent }}%</div>
    </div>
  </b-col>
  <b-col class="description">
    <div>
      <div class="title">Оцінка</div>
      <div class="date">Кількість правельних відповідей <br> {{ result.count_true }}/{{
result.count }}</div>
    </div>
  </b-col>
</b-row>
<div class="px-4">
  <div class="head-title">
    Назва тесту
  </div>
  <p>
    {{ data.title }}
  </p>
  <div class="head-title">
    Дата виконання тесту
  </div>
  <p>
    {{ result.date }}
  </p>
  <div class="head-title">
    Час вокинання тесту
  </div>
  <p>
    {{ result.time }}
```

```
            </p>
            <div class="head-title">
              Максимальний час вокинання тесту
            </div>
            <p>
              {{ data.time }}
            </p>
          </div>
          <template #modal-footer>
            <b-button size="sm" variant="success" @click="ok()">Готово</b-button>
          </template>
        </b-modal>
        <b-row>
          <b-col cols="4">
            <ul class="left-menu">
              <li v-for="(item, index) in data.answer" :key="index" @click="openQuestions(index)">
                Питання {{ index + 1 }} <span v-if="item.ready" class="answered">Пройдено</span>
              </li>
            </ul>
            <b-row>
              <b-col>Відповідей</b-col>
              <b-col class="text-right pr-5">{{ data.answer.filter(item => item.ready).length }}/{{
data.answer.length }}</b-col>
            </b-row>
          </b-col>
          <b-col class="px-4">
            <div class="timer">
              <img src="/img/ion_alarm-outline.png"> {{ theTime }} хв.
            </div>
            <div v-if="answerId != null" class="mb-5">
              <div class="head-title">
                {{ userUnswers[answerId].title }}
              </div>
              <p>
                {{ userUnswers[answerId].description }}
              </p>
              <div v-if="userUnswers[answerId].type == 'input'">
                <input
                  class="w-100"
```

```
                    v-model="userUnswers[answerId].answer"
                    type="text"
                    placeholder="Ваша відповідь"
                  >
                </div>
                <div v-if="userUnswers[answerId].type == 'list'">
                  <b-row class="answers mb-2" v-for="(item, index) in data.answer[answerId].answer"
:key="index">
                    <b-col cols="1">
                      <b-form-radio
                        @change="test(userUnswers[answerId].answer, index)"
                        :name="'answer_'+item.id"
                        class="type-account-item"
                        size="lg"
                      ></b-form-radio>
                    </b-col>
                    <b-col class="answer-text">
                      <div>
                        <img v-if="item.variant_file" :src="item.variant_file" class="w-50 mb-1"><br v-
if="item.variant_file">
                        {{ item.variant_text }}
                      </div>
                    </b-col>
                  </b-row>
                </div>
                <div v-if="userUnswers[answerId].type == 'sequence'">
                  <draggable v-model="userUnswers[answerId].answer">
                    <b-row v-for="(item, index) in userUnswers[answerId].answer" :key="index" class="answer-
text mt-3">
                      <b-col cols="1">
                        <b-icon icon="list" font-scale="1.5" style="vertical-align: -0.3em;"></b-icon>
                      </b-col>
                      <b-col class="px-1">
                        {{ item.variant_text }}
                      </b-col>
                    </b-row>
                  </draggable>
                </div>
              </div>
```

```
            <div class="battuns-test">
                <button v-if="data.answer.length != data.answer.filter(item => item.ready).length ||
data.answer.filter(item => item.ready).length == 1" class="active" @click="next">Далі</button>
                <button :class="data.answer.length == data.answer.filter(item => item.ready).length ? 'active' : ""
@click="finish">Завершити</button>
            </div>
          </b-col>
        </b-row>
    </b-container>
</template>
<script>
import draggable from "vuedraggable";
export default {
    data() {
        return {
            resultModal: false,
            timer: {
                deadline: 0,
                hours: 0,
                minutes: 0,
                seconds: 0
            },
            answerId: null,
            start: false,
            userUnswers: [],
            data: {
                title: "",
                description: "",
                time: "00:00:00",
                file: null,
                answer: [],
            },
            result: {
                count_true: 0,
                count: 0,
                procent: 0,
                time: "00:00:00",
                date: ""
            }
```

```
    }
},
components: {
    draggable
},
created() {
    this.fetchData();
},
computed: {
    theTime() {
        if(this.start) {
            var time = setInterval(() => {
                if(this.timer.deadline > 0) {
                    this.timer.deadline = this.timer.deadline - 1;
                    this.timer.hours = Math.floor(this.timer.deadline / 60 / 60);
                    this.timer.minutes = Math.floor(this.timer.deadline / 60) - (this.timer.hours * 60);
                    this.timer.seconds = this.timer.deadline % 60;
                    clearInterval(time);
                    if(this.timer.deadline == 0) {
                        this.finish();
                    }
                }
            }, 1000);
        }
        return this.timer.hours + ":" + this.timer.minutes + ":" + this.timer.seconds;
    }
},
methods: {
    test(answers, index) {
        for (let i = 0; i < answers.length; i++) {
            if(index == i) {
                answers[i].right = true
            } else {
                answers[i].right = false
            }
        }
    },
    ok() {
        this.$router.push({ path: '/cabinet' });
```

```
    },
    progress(procent) {
      return 314 - (314 / 100 * procent);
    },
    next() {
      this.data.answer[this.answerId].ready = 1;
      let index = this.data.answer.find(item => !item.ready);
      if(this.data.answer.indexOf(index) != -1) {
        this.answerId = this.data.answer.indexOf(index);
      }
    },
    finish() {
      this.data.answer.map((item, index) => {
        if(JSON.stringify(this.userUnswers[index].answer) == JSON.stringify(item.answer)) {
          this.result.count_true++;
        }
      });
      this.result.count = this.data.answer.length;
      this.result.procent = ((this.result.count_true / this.result.count) * 100).toFixed(0);
      this.result.time = this.theTime;
      axios.post('/api/result/'+this.$route.params.id, this.result)
      .then((response) => {
        this.result.date = response.data.date;
        this.resultModal = true;
      })
    },
    openQuestions(index) {
      this.answerId = index;
    },
    fetchData() {
      axios.get('/api/start-test/'+this.$route.params.id)
      .then((response) => {
        this.userUnswers = response.data.userData;
        this.data = response.data.data;
        this.timer.deadline = (Number(this.data.time.split(":")[0]) * 3600) +
(Number(this.data.time.split(":")[1]) * 60) + Number(this.data.time.split(":")[2]);
        this.userUnswers = this.userUnswers.map((item, index) => {
          if(item.type == 'input') {
            item.answer = "";
```

```
                }
                if(item.type == 'list') {
                    item.answer.map(i => {
                        i.right = false;
                        return i;
                    })
                }
                if(item.type == 'sequence') {
                    item.answer.sort(() => Math.random() - 0.5);
                }
                return item;
            });
            this.showMsgBoxOne();
        })
    },
    showMsgBoxOne() {
        var text = "Тест \"" + this.data.title + "\". Час виконання тесту " + this.data.time + ". Кількість завдань " + this.data.answer.length + ".";
        this.boxTwo = ''
        this.$bvModal.msgBoxConfirm(text, {
        title: 'Почати тест?',
        size: 'sm',
        buttonSize: 'sm',
        okTitle: 'Почати',
        cancelTitle: 'Назад',
        footerClass: 'p-2',
        noCloseOnBackdrop: true,
        hideHeaderClose: false,
        centered: true
        })
        .then(value => {
            if(value) {
                this.answerId = 0;
                this.start = true;
            } else {
                this.$router.go(-1)
            }
        })
    }
```

```
    },
}
</script>
<style lang="css" scoped>
    .description {
        color: #322B57;
        display: flex;
        align-items: center;
    }
    .title {
        font-size: 16px;
        line-height: 20px;
        margin-bottom: 5px;
    }
    .date {
        font-size: 14px;
        line-height: 14px;
    }
    .progressbar-text {
        position: absolute;
        left: 50%;
        top: 50%;
        padding: 0px;
        margin: 0px;
        transform: translate(-50%, -50%);
        color: rgb(0, 0, 0);
        font-size: 1rem;
    }
    .round-block {
        display: block;
        width: 100%;
    }
    .round-block .round {
        stroke-dasharray: 314, 314;
    }
    input, textarea, select {
        border: 1px solid silver;
        border-radius: 5px;
        padding: 5px 10px;
```

```
    }
    .answers {
        display: flex;
        align-items: center;
    }
    .answer-text {
        padding: 10px 15px;
        border: 1px solid silver;
        border-radius: 5px;
        display: flex;
        align-items: center;
    }
    .left-menu {
        min-height: 400px;
    }
    .left-menu span {
        margin-right: 25px;
    }
    .left-menu span.answered {
        float: right;
    }
    .timer {
        text-align: right;
        color: #2ECAA3;
        font-size: 18px;
        padding: 10px 0;
    }
    .timer img {
        margin-right: 5px;
    }
    .battuns-test {
        position: absolute;
        bottom: 0;
        right: 0;
    }
</style>
```

ResultTests.vue

Компонент з результатами тестів.

```
<template>
  <b-container class="block px-5 py-4">
    <b-row>
      <b-col md="4" lg="4" xl="4" xs="12">
        <div class="back">
          <router-link to="/cabinet"><img src="/img/back.png"> В меню</router-link>
        </div>
        <ul class="left-menu">
          <li v-for="(item, index) in results" :key="index" @click="openTest(index, item)">
            {{ item.test.title }}
          </li>
        </ul>
      </b-col>
      <b-col class="px-4">
        <div v-if="testId != null">
          <div class="head-title">
            Назва
          </div>
          <p>
            {{ resultItem.test.title }}
          </p>
          <div class="head-title">
            Опис
          </div>
          <p>
            {{ resultItem.test.description }}
          </p>
          <div class="head-title">
            Час виконання
          </div>
          <p>
            {{ resultItem.time }}
          </p>
          <div class="head-title">
            Кількість питань
          </div>
          <p>
            {{ resultItem.count }}
          </p>
```

```
                <div class="head-title">
                    Дата проходження
                </div>
                <p>
                    {{ resultItem.date }}
                </p>
                <div class="head-title">
                    Автор тесту
                </div>
                <p>
                    {{ resultItem.test.user.name }} {{ resultItem.test.user.surname }}
                </p>
                <div class="head-title">
                    Оцінки
                </div>
                <table>
                    <tr>
                        <td width="200px">Правельних відповідей</td>
                        <td>{{ resultItem.count_true }}</td>
                    </tr>
                    <tr>
                        <td>Не правельних відповідей</td>
                        <td>{{ resultItem.count - resultItem.count_true }}</td>
                    </tr>
                    <tr>
                        <td>Результат</td>
                        <td>{{ resultItem.procent }}%</td>
                    </tr>
                </table>
            </div>
        </b-col>
    </b-row>
</b-container>
</template>
<script>
export default {
    data() {
        return {
            testId: null,
```

```
            resultItem: {
                procent: 0,
                time: 0,
                count_true: 0,
                count: 0,
                date: "",
                test: {
                    title: "",
                    description: "",
                    time: ""
                }

            },
            results: []
        }
    },
    created() {
        this.fetchData();
    },
    methods: {
        openTest(index, item) {
            this.testId = index;
            this.resultItem = Object.assign(this.resultItem, item);
        },
        fetchData() {
            axios.get('/api/cabinet')
            .then((response) => {
                this.results = response.data.test_results;
            })
        }
    }
}
</script>
<style lang="css" scoped>
    @media screen and (max-width: 600px) {
        .left-menu {
            min-height: auto !important;
        }
    }
```

```
    .left-menu {
        min-height: 400px;
    }
    .back {
        display: flex;
        align-items: center;
    }
    .back a {
        color: #000000;
        font-size: 16px;
    }
    .back img {
        margin-right: 5px;
    }
</style>
```

Chat.vue

Компонент чату.

```
<template>
    <b-container class="block px-5 py-4">
        <b-row>
            <b-col md="4" lg="4" xl="4" xs="12">
                <div class="back">
                    <router-link to="/cabinet"><img src="/img/back.png"> В меню</router-link>
                </div>
                <ul class="left-menu">
                    <li :class="$route.params.id == item.id ? 'active' : "" v-for="(item, index) in chats" :key="index"
@click="openMessage(item.id)">
                        {{ item.messages[item.messages.length - 1].user.name }} {{
item.messages[item.messages.length - 1].user.surname }}: {{ item.messages[item.messages.length - 1].text }}
                    </li>
                </ul>
            </b-col>
            <b-col class="px-4">
                <div class="chat mt-3 pr-1 mb-2">
                    <div class="d-flex justify-content-center" v-if="preloader">
                        <b-spinner style="width: 2rem; height: 2rem;" label="Large Spinner"></b-spinner>
                    </div>
                    <div v-if="!preloader && messages == 0" class="text-center">
```

```
                Діалоги відсутні
            </div>
            <div v-for="(item, index) in messages" :key="index" :class="item.message.account_id ==
$store.getters.authUser.id ? 'alert alert-primary' : 'alert alert-secondary'">
                <div class="photo">
                    <img class="avatar" :src="item.account.photo">
                </div>
                <div class="text">
                    {{ item.message.text }}
                </div>
                <div class="date">
                    {{ item.message.created_at.substring(0, 10) }} {{ item.message.created_at.substring(11, 16)
}}
                </div>
            </div>
        </div>
        <b-row>
            <b-col cols="9">
                <b-form-input v-model="newMessage"></b-form-input>
            </b-col>
            <b-col cols="3">
                <button class="active" @click="sendMessage">Надіслати</button>
            </b-col>
        </b-row>
    </b-col>
  </b-row>
 </b-container>
</template>
<script>
export default {
  data() {
    return {
      preloader: true,
      messages: [],
      newMessage: '',
      chats: []
    }
  },
  created() {
```

```
    this.fetchData();
    this.getChats();
    this.fetchMessages();
    Echo.channel('chat-'+this.$route.params.id)
        .listen('MessageSentEvent', (e) => {
            this.messages.push(e);
        });
},
methods: {
    fetchData() {
        axios.get('/api/cabinet')
        .then((response) => {
            this.user = response.data;
        })
    },
    getChats() {
       axios.get('/api/chats')
        .then((response) => {
            this.chats = response.data;
        })
    },
    openMessage(id) {
        this.$router.push({path: '/chat/'+id});
    },
    fetchMessages() {
        axios.get('/api/message/'+this.$route.params.id)
        .then(response => {
            this.preloader = false;
            this.messages = response.data;
        }).catch(() => {
            this.preloader = false;
        });
    },

    addMessage(message) {
        axios.post('/api/message/'+this.$route.params.id, {
            message
        }).then(response => {
            this.messages.push(response.data);
```

```
            });
        },

        sendMessage() {
            this.addMessage(this.newMessage);
            this.newMessage = '';
        }
    }
}
</script>
<style lang="css" scoped>
    @media screen and (max-width: 600px) {
        .left-menu {
            min-height: auto !important;
        }
    }
    .chat {
        overflow-y: scroll;
    }
    .chat::-webkit-scrollbar {
        width: 5px;
    }
    .chat::-webkit-scrollbar-track {
        box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.3);
        border-radius: 10px;
    }
    .chat::-webkit-scrollbar-thumb {
        background-color: #000000;
        border-radius: 10px;
    }
    .left-menu {
        min-height: 400px;
    }
    .back {
        display: flex;
        align-items: center;
    }
    .back a {
        color: #000000;
```

```
    font-size: 16px;
  }
  .back img {
    margin-right: 5px;
  }
  .chat {
    height: 400px;
    overflow: auto;
  }
  .alert {
    float: left;
    width: 100%;
  }
  .alert .photo {
    width: 10%;
    height: 100%;
    float: left;
  }
  .alert .text {
    width: 90%;
    float: left;
  }
  .alert .date {
    width: 90%;
    float: right;
    text-align: right;
  }
  .alert .photo img {
    width: 40px;
    border-radius: 20px;
  }
</style>
```

User.vue

Компонент сторінки користувача.

```
<template>
  <b-container>
    <b-row>
      <b-col md="8" lg="8" xl="8" xs="12" class="user-block block mr-5">
```

```
            <User :user="user" @update="fetchData()"></User>
          </b-col>
          <b-col class="block">
            <div class="contacts">
              <div class="title">
                Контакти
              </div>
              <div class="contact">
                Email: {{ user.email }}
              </div>
              <div class="contact" v-if="user.phone">
                Телефон: {{ user.phone }}
              </div>
              <div class="title">
                Соціальні мережі
              </div>
              <div class="contact" v-for="(item, index) in user.socials" :key="index">
                <a :href="item.social.link"><img :src="item.social.logo" class="mr-2"> {{ item.social.title
}}</a>
              </div>
            </div>
          </b-col>
        </b-row>
        <b-row class="mt-5">
          <b-col md="8" lg="8" xl="8" xs="12" class="user-block mr-5 p-0">
            <div v-for="(item, index) in user.news" :key="index">
              <news-item :item="item"></news-item>
            </div>
          </b-col>
          <b-col class="p-0">
            <div class="block p-4">
              <div class="title-block">Підписники</div>
              <user-item-short
                v-for="(item, index) in user.followers"
                :key="index"
                :item="item"
                @update="fetchData()"
              ></user-item-short>
            </div>
```

```
            </b-col>
        </b-row>
    </b-container>
</template>
<script>
import ItemTestResult from '../../components/site/cabinet/ItemTestResult';
import UserItemShort from '../../components/site/UserItemShort';
import User from '../../components/site/cabinet/User';
import NewsItem from '../../components/site/NewsItem';
export default {
    components: {
        User,
        ItemTestResult,
        NewsItem,
        UserItemShort
    },
    data() {
        return {
            user: {
                name: "",
                surname: "",
                photo: "",
                email: "",
                phone: "",
                country: {
                    title: ""
                },
                role: {
                    title: ""
                },
                tests: [],
                socials: [],
                news: [],
                followers: [],
                my_followers: []
            },
        }
    },
    created() {
```

```
        this.fetchData();
    },
    methods: {
        fetchData() {
            axios.get('/api/user/'+this.$route.params.id)
            .then((response) => {
                console.log(response.data)
                this.user = response.data;
            })
        }
    },
    watch: {
        $route(to, from) {
            this.fetchData();
        }
    }
}
</script>
<style lang="css" scoped>
    @media screen and (max-width: 600px) {
        .subscribe-block {
            margin-top: 0px !important;
        }
        .user-block {
            margin: 0 0 30px 0 !important;
        }
    }
    .title-block {
        color: #000000;
        font-size: 20px;
        margin-bottom: 20px;
    }
    .contacts {
        padding: 20px 5px;
    }
    .contacts .title {
        color: #322B57;
        font-size: 16px;
        margin: 10px 0;
```

```
    }
    .contacts .contact {
        padding: 5px 15px;
        display: flex;
        align-items: center;
    }
</style>
```