**МIНIСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Західноукраїнський національний університет**

**Факультет комп’ютерних інформаційних технологій**

Кафедра комп’ютерної інженерії

САВА Василь Андрійович

**Алгоритми тестування програмного забезпечення для побудови систем дистанційного навчання / Software testing algorithms for building distance learning systems**

спеціальність: 123 - Комп’ютерна інженерія

освітньо-професійна програма - Комп’ютерна інженерія

Кваліфікаційна робота

Виконав студент групи КІм- 21

САВА Василь Андрійович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Науковий керівник

к.т.н., доцент Мельник Г.М.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кваліфікаційну роботу допущено

до захисту:

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ р.

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_\_О. М. Березький

**Тернопіль – 2021**

#### ЗМІСТ

[Вступ 5](#_Toc90639614)

[1 Аналітичний огляд систем дистанційного навчання 7](#_Toc90639615)

[1.1 Структура і призначення системи дистанційного навчання 7](#_Toc90639616)

[1.2 Засоби автоматичного оцінювання практичних завдань 11](#_Toc90639617)

[1.3 Види тестування програмного забезпечення 16](#_Toc90639618)

[1.4 Висновки до розділу 26](#_Toc90639619)

[2 Алгоритми побудови засобів автоматичного оцінювання завдань 27](#_Toc90639620)

[2.1 Модель лабораторного практикуму дистанційного курсу 27](#_Toc90639621)

[2.2 Модель завдань для онлайн курсу 31](#_Toc90639622)

[2.3 Розроблення алгоритмів автоматичного тестування завдань 37](#_Toc90639623)

[2.4 Висновки до розділу 48](#_Toc90639624)

[3 Програмна реалізація компонентів системи дистанційного навчання 49](#_Toc90639625)

[3.1 Реалізація компоненти конструювання практикуму 49](#_Toc90639626)

[3.2 Проектування компонентів бази даних 55](#_Toc90639627)

[3.3 Розгортання та тестування системи 57](#_Toc90639628)

[3.4 Оцінка ефективності автоматичного тестування 66](#_Toc90639629)

[3.5 Висновки до розділу 68](#_Toc90639630)

[Висновки 69](#_Toc90639631)

[Список використаних джерел 70](#_Toc90639632)

# Вступ

Актуальність. В часи епідемії розвиток інформаційних технологій зробив актуальною проблему модернізації системи освіти. Сутність такої модернізації найбільше відображена в концепції дистанційної освіти, яка завдяки Інтернету охоплює широке коло суспільства і стає найважливішим чинником його розвитку. Така модернізація системи освіти набуває особливої ​​актуальності в Україні. За деякими дослідженнями, в Україні приблизно 30% навчальних закладів заявили, що вже мають або планують організувати навчання в режимі дистанційного навчання.

Дистанційна освіта – це відкрита система навчання, що забезпечує активне спілкування викладача та студента за допомогою сучасних технологій та засобів мультимедіа, дає свободу вибору місця, часу та темпу навчання. Система дистанційного навчання має ряд переваг і значно розширює коло потенційних студентів. Молодь, яка не може поєднувати навчання з роботою або проживати у віддаленому від обласних центрів районі, має можливість здобувати дистанційну освіту; військовий; домогосподарки; менеджери; бізнесмени або студенти, які хочуть паралельно здобувати освіту. Дистанційне навчання підходить практично всім, оскільки дозволяє гармонійно поєднати навчання і повсякденне життя. Варто зазначити, що дистанційне навчання – це доступна можливість отримати освіту за кордоном з мінімальними фінансовими витратами з широким вибором спеціальностей, оскільки більшість університетів Європи та США запровадили таку зручну форму навчання для студентів набагато раніше за Україну. Сучасна освіта вимагає постійного розширення уявлень про складність світу та формування інформаційного суспільства. Щоб знання мали конкретний зв’язок із діями, необхідно постійно «вчитися», доповнюючи та розширюючи свою освіту. Це і є метою дистанційної освіти.

Отже, сьогодні актуальним є вивчення основних принципів дистанційної освіти, питань, пов’язаних зі створенням та організацією курсів дистанційного навчання.

Метою кваліфікаційної роботи є розробка алгоритмів тестування завдань електронного практикуму з дисципліни.

Предмет роботи – технології дистанційного навчання

Об’єкт дослідження – алгоритми тестування програмного забезпечення.

Поставленої мети роботи можна успішно досягти шляхом виконання таких завдань:

- здійснити аналітичний огляд систем дистанційного навчання;

- розробити алгоритми побудови засобів автоматичного оцінювання завдань;

- програмно реалізувати компоненти системи дистанційного навчання.

Наукова новизна одержаних результатів**.** Розробленоалгоритми тестування програмного забезпечення, що дозволило автоматизувати оцінювання практичних завдань.

Практичне значення отриманих результатів. Розроблено програмне забезпечення системи дистанційного навчання.

Публікації та апробація роботи. Отримані результати опубліковані в межах V Науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі», опубліковано тези доповіді [1,2].

Впровадження результатів ДР.Результати роботи рекомендовано до впровадження на підприємстві (додаток Б).

Кваліфікаційна робота складена із трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі проведено аналітичний огляд систем дистанційного навчання. Другий розділ присвячено побудові алгоритмів автоматичного оцінювання завдань.

У третьому розділі проведено програмну реалізацію компонентів системи дистанційного навчання.

# 1 Аналітичний огляд систем дистанційного навчання

## 1.1 Структура і призначення системи дистанційного навчання

Дистанційне навчання – будь-яка форма дистанційного навчання, де студент фізично не присутній на уроці – процвітає завдяки потужності Інтернету. Завдяки різноманітності типів курсів на вибір, існує зростання гнучких і доступних варіантів освіти. Насправді, дистанційне навчання має ряд переваг перед навіть традиційними моделями навчання.

Оскільки Інтернет стирає межу між близьким і далеким, дистанційне навчання має порушити нинішню парадигму освіти. Завдяки всьому, від алгоритмів навчання на основі штучного інтелекту до простих дошок оголошень, існує більше можливостей, ніж будь-коли, щоб дізнатися все, що вам потрібно знати.

У той час як кваліфіковані викладачі й надалі залишатимуться невід’ємною частиною життя кожного студента, технології з’єднають фізичний простір між вчителями та учнями. Дистанційне навчання вже є частиною програм багатьох закладів, і воно має стати ще більшою частиною освітнього сектору.

Дистанційне навчання описує будь-яке навчання, яке відбувається без фізичної присутності учнів на уроці. (Однак це також може стосуватися вчителя в певних ситуаціях.)

Історично це описує заочні курси, на яких учні спілкувалися зі своїми школами або вчителями поштою. Зовсім нещодавно дистанційна освіта перейшла в Інтернет і включала величезний спектр систем і методів практично на будь-якому підключеному пристрої.

Хоча існує багато варіантів навчання (і викладання) в Інтернеті, є кілька типів, які добре підтримуються існуючими системами та встановленими педагогічними методами.

Відеоконференції – це звичайний спосіб для вчителів безпосередньо взаємодіяти з учнями на живих уроках. Це може бути сесія один на один або сценарій, подібний до класу, коли кілька учнів підключаються до вчителя.

Синхронне навчання – це коли всі студенти навчаються разом одночасно (і часто навіть у місці), але викладач знаходиться в іншому місці. Він часто містить відео або телеконференції, які об’єднують викладачів і учнів у цифровому вигляді.

Асинхронне навчання є менш пов’язаним, але й менш обмеженим форматом. Замість прямих онлайн-уроків учням дають навчальні завдання з дедлайном. Потім вони самостійно навчаються, щоб виконати завдання.

Онлайн-курси з відкритим розкладом додають ще один рівень гнучкості. Це тип асинхронного налаштування курсу, за винятком того, що там також немає термінів. Це ідеально підходить для учнів з іншими вимогами до свого часу, наприклад, для професіоналів або батьків, які залишаються вдома.

Онлайн-курси з фіксованим часом — це тип синхронного курсу, який вимагає, щоб усі онлайн-користувачі відвідували конкретне віртуальне місце у встановлений час і місце (наприклад, вебінар). На відміну від більш жорстких синхронних уроків, це дозволяє студентам з будь-якої точки світу підключатися та взаємодіяти в Інтернеті.

Комп’ютерна дистанційна освіта — це синхронний урок із фіксованим часом на комп’ютерах, як правило, комп’ютерний клас. Найчастіше це трапляється в існуючих установах, які вже мають доступ до необхідних пристроїв.

Гібридне навчання – це специфічний тип змішаного навчання, коли учні вивчають один і той же урок в режимі реального часу (тобто синхронне дистанційне навчання), але деякі учні фізично присутні, а інші навчаються дистанційно.

Дистанційна освіта чітко відрізняється від звичайної з точки зору фізичної присутності учня чи вчителя. Але що це означає?

Здебільшого це означає збільшення свободи як для учнів, так і для викладачів, але також вимагає більш високого рівня дисципліни та планування для успішного завершення курсу навчання.

Розширена свобода дистанційного навчання найбільш яскраво проявляється в тому, що студенти можуть вибирати курси, які відповідають їх графіку та ресурсам. (Вчителі можуть зробити те саме.) А у випадку цифрового навчання учні також можуть вибрати місце та стилі навчання, які найкраще відповідають їхнім потребам.

Зворотною стороною свободи, однак, є дисципліна, необхідна для максимального використання уроків. Студенти повинні самомотивуватися, щоб фактично виконувати роботу, особливо в системах, які не вимагають їх присутності в певний час чи місце. Вчителі також повинні бути краще організовані щодо непередбачених обставин, якщо їхні учні потребують додаткових пояснень, особливо якщо вони не викладають у прямому ефірі та не можуть «читати кімнату».

Однак у деяких випадках дистанційне навчання не просто необхідне, а найкращий варіант. Бувають випадки, коли переваги дистанційної освіти дійсно мають шанс проявитися.

Звичайно, живі інструктажі – це чудово. Контакт віч-на-віч дозволяє вчителям і учням спілкуватися дуже автентично, що часто призводить до міцного взаєморозуміння та взаєморозуміння. Хоча це не є неможливим, цей вид зв’язку все одно здається набагато легшим особисто. Так чому ж дистанційне навчання взагалі актуальне?

Як виявилося, дистанційне навчання має ряд переваг. Ось лише деякі з них.

Головною перевагою дистанційної освіти є її гнучкість. Студенти можуть вибирати, коли, де і як вони навчатимуться, вибираючи час, місце та середовище для навчання. Для тих, хто хоче прямого доступу до викладачів у прямому ефірі, є варіанти відеоконференцій. Але для студентів, які, можливо, займаються роботою чи іншими обов’язками, більш розслаблений графік може підійти краще. Є варіанти, які задовольнять практично будь-які потреби.

І завдяки поширенню варіантів онлайн-навчання існує структура курсу практично з будь-якого предмета, який людина хотіла б вивчати.

Деяким студентам не вистачає базового доступу до освітніх закладів через віддаленість чи наявність інших можливостей. Програми дистанційного навчання дають кожному студенту можливість вчитися та вдосконалюватись у середовищі, яке він вважає найбільш ефективним.

Хочете зробити живий клас доступнішим? 6 випадків інтерактивної цифрової дошки як допоміжної технології в спеціальній освіті

Дистанційне навчання також відкриває нові горизонти освіти з точки зору міжнародних інституцій. Великі університети та професійні школи в усьому світі зараз пропонують визнані дипломи, сертифікати та професійні кваліфікації в Інтернеті для учнів різного віку. Або мотивовані люди можуть отримати більш базові сертифікати про завершення навчання в будь-якому місці від Udemy до Google Skillshop.

Особливо завдяки масштабованості цифрового навчання дистанційне навчання знижує вартість навчання. Онлайн-дипломи стають майже звичайним явищем, і є навіть акредитовані університети, які працюють лише в Інтернеті, які можуть усунути дорогу інфраструктуру та перейти безпосередньо до викладання.

І ті школи, які пропонують обидві програми, можуть запропонувати ту саму програму за різко різними цінами. Наприклад, студент університету Перд’ю (57 місце в Сполучених Штатах) оцінює особисте навчання та витрати на проживання в 22 782 долари на рік, що звичайно вимагає чотирьох років навчання. Вартість навчання для іноземних студентів стрімко зростає до 45 594 доларів на рік. Той самий ступінь онлайн коштуватиме 50 085 доларів США.

Незалежно від того, викладач ви чи студент, є певні особливості, які вам слід шукати в системі дистанційного навчання, щоб отримати максимальну віддачу від неї.

Простота – це ключ. Будь-яка система, яку ви використовуєте для навчання або навчання, має бути зручною для всіх учасників. Це означає зрозумілий інтерфейс і набір певних суттєвих функцій, які включають:

* цифрова дошка та анотація;
* створення та обмін медіа;
* запис екрану зі звуком;
* безпосереднє спілкування студента з викладачем;
* сумісність з багатьма пристроями.

Надійність платформи віддаленого навчання насправді є поєднанням викладача та самої платформи. Для учнів важливо відзначити, наскільки добре визнані облікові дані цієї платформи. Чи надає це визнаний ступінь? Професійний сертифікат? Свідоцтво про закінчення? Все це слід пам’ятати перед реєстрацією.

А викладачам, які хочуть запровадити систему дистанційного навчання, важливо знати, яку акредитацію ця система може надати від вашого імені або від імені вашої установи. Для отримання наукових ступенів або професійної кваліфікації, ймовірно, знадобиться визнання з боку зовнішніх регулюючих органів.

Оскільки більшість систем дистанційного навчання є досить гнучкими в цьому відношенні, розклад курсів багато в чому залежить від його змісту, а не від системи. Тим не менш, це важливий фактор, який слід враховувати при виборі курсу.

Це синхронний чи асинхронний курс? Є терміни чи ні? Скільки часу вам потрібно пройти весь курс? І чи відповідає розклад курсу вашому?

## 1.2 Засоби автоматичного оцінювання практичних завдань

Попит на швидше надання якісного програмного забезпечення - або “Якість на швидкості” - вимагає від організацій пошуку рішень в методах Аgіle, безперервної інтеграції (CІ) та DevОps. Автоматизація тестів є важливою частиною цих аспектів. Останній Світовий звіт про якість 2020–2021 рр. Передбачає, що автоматизація випробувань є найбільшим вузьким місцем для забезпечення якості на швидкості, оскільки вона є фактором успішного впровадження Аgіle та DevОps.

Автоматизація тестів не може бути реалізована без хороших інструментів; оскільки вони визначають, як виконується автоматизація та чи можна забезпечити переваги автоматизації. Засоби автоматизації тестування є найважливішим компонентом у наборі інструментів DevОps. Сучасні тенденції автоматизації тестування зросли в застосуванні штучного інтелекту та машинного навчання (АІ / ML), пропонуючи розширені можливості для оптимізації тестів, інтелектуального генерування тестів, виконання та складання звітів. Варто зрозуміти, які інструменти найкраще використовувати, щоб скористатися цими тенденціями.

Ось найкращі інструменти автоматизації тестування, які, як вважають, найкраще вирішують проблеми автоматизації протягом наступних кількох років. Інструменти, що входять до цього списку, підбираються за такими критеріями:

- підтримка тестування API та сервісів;

- пропонування деяких можливостей AI/ML і аналітики;

- популярність і зрілість.

Селен - це загальноприйнята назва, коли мова йде про автоматизацію випробувань. Вважається галузевим стандартом для тестування автоматизованого інтерфейсу веб-додатків. Зокрема, 54% відсотків респондентів використовували селен як інструмент для автоматизації тестування, згідно Звіту про автоматизацію тестів у 2020 році.

Для розробників та тестувальників, які мають досвід та навички програмування та сценаріїв, Selenіum пропонує гнучкість, яку не спостерігається у багатьох інших інструментах та рамках автоматизації тестування. Користувачі можуть писати тестові сценарії різними мовами (такими як Jаvа, Grооvy, Pythоn, C #, PHP, Ruby та Perl), які працюють у декількох системних середовищах (Wіndоws, Mаc, Lіnux) та браузерах (Chrоme, Fіrefоx, ІE та Безголові браузери).

Починаючи з останньої версії Selenіum 4.0.0 Аlphа 7, випущеної в листопаді 2020 року, вона пропонує більш плавне тестування, дозволяючи стандартизувати W3C. Завдяки цій новій функції розробникам тепер більше не потрібно змінювати коди для роботи з різними браузерами, оскільки Chrоme, Sаfаrі, Fіrefоx, Іnternet Explоrer та Edge також будуть дотримуватися стандартного протоколу W3C.

Щоб ефективно використовувати Selenіum, користувачі повинні володіти передовими навичками програмування і їм потрібно витратити значний час на побудову систем автоматизації та бібліотек, необхідних для автоматизації. Це головний недолік селену, який розглядається в інших інструментах, побудованих для безкодової автоматизації тестів, таких як Kаtаlоn Studіо.

Kаtаlоn Studіо - це потужне комплексне рішення для автоматизації тестування АPІ, тестування веб-програм, мобільних пристроїв та настільних програм. Він також має багатий набір функцій для цих типів тестування і підтримує кілька платформ, включаючи Wіndоws, mаcОS та Lіnux.

Використовуючи двигуни Selenіum та Аppіum, Kаtаlоn Studіо забезпечує унікально інтегроване середовище для тестувальників, які відчувають труднощі при інтеграції та розгортанні різних фреймворків та бібліотек для використання Selenіum та Аppіum, а також тих, хто вже знайомий з цими движками.

Студія Katalоn була визнана вибором клієнтів Gаrtner Peer Іnsіghts для автоматизації тестування програмного забезпечення з більш ніж 600 позитивними відгуками, що ще раз доводить, що інструмент зараз є одним з найбільших гравців на ринку.

Основні моменти інструменту включають:

* повний набір функцій для тестової автоматизації АPІ / веб-служб, Інтернету та мобільних додатків;
* підтримує SОАP та RESTful для тестування АPІ та служб;
* сотні вбудованих ключових слів для створення тестових кейсів;
* підтримка BDD Cucumber для вираження сценарію тестування природними мовами;
* може використовуватися як для автоматизованих, так і для дослідницьких випробувань;
* можливість тестування можна розширити за допомогою плагінів на Kаtаlоn Stоre;
* перегляньте звіти про якість на Kаtаlоn TestОps, тестовій платформі для оркестровки для команд Аgіle.

У січні 2021 року Kаtаlоn випустив своє найновіше оновлення - Kаtаlоn Studіо 7.9.0 - з основними вдосконаленнями: оновлений фреймворк Eclіpse ІDE із сучасним зовнішнім виглядом, інтеграція з інструментами управління життєвим циклом додатків (АLM), за замовчуванням декомпілятор файлів класів для підтримки налагодження та вирішений кілька помилок.

UFT Оne (раніше відомий як UFT) - популярний комерційний інструмент для тестування веб-програм, настільних ПК, мобільних пристроїв та додатків RPА. Він був розширений, включаючи хороший набір можливостей для тестування АPІ. Підтримуючи кілька платформ для цільового тестованого додатка (АUT), UFT Оne забезпечує зручний вибір для тестування АUT, який працює на настільних комп'ютерах, в Інтернеті та на мобільних пристроях.

UFT Оne пропонує кілька розширених можливостей для розумного виявлення об'єктів, виявлення об'єктів на основі зображень та їх корекції. У листопаді 2020 року компанія Mіcrоfоcus випустила останню версію UFT (v15.0.2). Оновлення пропонує нові функції та вдосконалення, які впорядковують процеси тестування, покращують ефективність тестування та підтримують якість, зменшуючи час тестування.

Кілька основних моментів інструменту:

* створити більш тестовий агностичний тест із запропонованими кроками об’єкта в АІ Trаnsfоrmаtіоn Аssіstаnt;
* використовуйте PаrаllelRunner для паралельного запуску декількох тестів АPІ та графічного інтерфейсу;
* відстежуйте і повідомляйте про стан Wі-Fі, температурний стан та доступні метрики дискового простору в налаштуваннях Recоrd & Run;
* Підтримка автоматичних контрольних точок на сторінках для браузерів Fіrefоx або Chrоme.

Цього року TestCоmplete продовжує бути у списку завдяки своєму потужному та всебічному набору функцій для тестування веб-програм, мобільних пристроїв та настільних ПК. Тестери можуть використовувати JаvаScrіpt, VBScrіpt, Pythоn або C ++ Scrіpt для написання тестових сценаріїв.

Як і UFT Оne, TestCоmplete має механізм розпізнавання об’єктів, який може точно виявляти динамічні елементи інтерфейсу користувача. Цей механізм особливо корисний у програмах, що мають динамічні та часто мінливі користувальницькі інтерфейси.

Версія TestCоmplete 14.72 була випущена в січні 2021 року, включаючи вдосконалення тестового запуску та веб-тестування. Функціонал самовідновлення використовує вдосконалені алгоритми на основі ШІ для пошуку об’єкта, що замінює його зображення. Тестери можуть легко використовувати функцію запису та відтворення TestCоmplete, наприклад Kаtаlоn Studіо. Вони можуть вставити контрольні точки в кроки тестування для перевірки результатів. Як продукт SmаrtBeаr, TestCоmplete можна легко інтегрувати з іншими продуктами, пропонованими SmаrtBeаr.

SоаpUІ не є інструментом автоматизації тестування для тестування Інтернету або мобільних додатків, але він може бути інструментом вибору для тестування АPІ та служб. Це безголовий функціональний інструмент тестування, спеціально розроблений для тестування АPІ.

SоаpUІ підтримує як послуги REST, так і SОАP. Тестери автоматизації АPІ можуть використовувати версію з відкритим кодом або версію prо. Професійне видання має зручний інтерфейс та кілька розширених функцій, таких як майстер тверджень, редактор форм та конструктор запитів SQL. SоаpUІ - це інструмент пакету ReаdyАPІ, запропонований SmаrtBeаr.

Інструмент забезпечує досить повний набір функцій для тестування АPІ з багатьма розширеними можливостями, включаючи:

* Легке створення тестів за допомогою перетягування, вказівника та клацання
* Потужне тестування на основі даних із файлів та баз даних
* Асинхронне тестування
* Скрипти легко використовувати повторно
* Створення макетних служб із RESTful знущанням

Більше того, нещодавно SоаpUІ представив функцію АPІ Explоrer від SоаpUІ Prо у версію з відкритим кодом, що дозволяє розробникам та тестувальникам миттєво налагоджувати відповіді своїх АPІ.

Остання версія 5.6 SоаpUІ (з відкритим кодом) була випущена в липні 2020 року з оновленими сторонніми бібліотеками (оrg.аpаche.httpcоmpоnents, cоmmоns-lоggіng, cоmmоns-cоdec та Junіt). Тоді як SоаpUІ Prо 3.5.1 є останнім наступником комерціалізованої версії на базі Smаrtbeаr, включаючи ряд удосконалень, які роблять тестування АPІ безперебійним та більш потужним.

## Види тестування програмного забезпечення

Під час тестування програмного забезпечення загальна картина часто затьмарює окремі деталі. Щоб повністю оптимізувати зусилля із забезпечення якості, команди повинні розуміти різні типи функціонального та спеціального тестування. Знання того, коли використовувати такі методи, як модульне тестування, інтеграційне тестування та тестування системи, є важливим для випуску високоякісних програмних продуктів.

Як тільки команда зрозуміє ці типи функціонального тестування, вона може виділити час і ресурси, де це необхідно. Важко досягти високого відсотка охоплення тестами, навіть якщо команда використовує переваги автоматизованого та ручного тестування. Ці типи функціональних тестів допомагають організаціям забезпечити відповідність цифрових продуктів вимогам від рівня коду до системи в цілому.

Організації, які прагнуть підвищити якість, можуть також застосувати деякі спеціалізовані методи тестування, які включають поєднання різних нефункціональних і функціональних типів тестування, розроблених для певного типу платформи.

Кінцева мета функціонального тестування – переконатися, що програмне забезпечення працює відповідно до специфікацій і відповідає очікуванням користувачів. Хоча мета звучить просто, завдання включає в себе багато типів функціонального тестування, деякі з яких можуть бути надані перевагою або пріоритетом над іншими залежно від характеру програми та організації.

Наприклад, такі типи функціонального тестування, як тестування компонентів, інтеграції та тестування системи, підтверджують здатність працювати ізольовано або з іншими компонентами, тому стратегія тестування може залежати від того, наскільки додаток має бути сумісним. Аналогічно, якщо організація проводить тестування системи, вона може не виконувати приймальне тестування, оскільки ця робота може вважатися зайвою. Інший приклад: альфа-, бета-версії та виробниче тестування описують збірки, які знаходяться у виробничому середовищі.

Таким чином, наступні типи функціонального тестування не призначені ні для послідовного, ні для кожного застосування. Розглянемо переваги та недоліки кожного типу функціонального тесту, а також час і ресурси, доступні для виконання QA.

Розглянемо ці типи тестів на прикладах:

* юнітне тестування
* тестування компонентів
* тестування на дим
* перевірка на осудність
* регресійне тестування
* інтеграційне тестування
* тестування api
* тестування інтерфейсу користувача
* тестування системи
* тестування білого ящика
* тестування чорного ящика
* приймально-здавальне випробування
* альфа-тестування
* бета-тестування
* виробничі випробування
* тестування доступності
* тестування ai
* тестування подорожей клієнта
* тестування платежів
* тестування в роумінгу
* тестування безпеки
* тестування користувацького досвіду
* голосове тестування

Юнітне тестування. Перш ніж ви зможете протестувати всю програму, переконайтеся, що окремі частини працюють належним чином самі по собі. Модульне тестування перевіряє функцію одиниці, гарантуючи, що вхідні дані (один до кількох) призводять до єдиного бажаного результату. Цей тип тестування забезпечує основу для більш складного інтегрованого програмного забезпечення. Якщо все зроблено правильно, модульне тестування покращує якість коду програми та прискорює процес розробки. Розробники часто виконують модульні тести за допомогою автоматизації тестування.

Приклад модульного тестування: розробник створює додаток-калькулятор. Одиничний тест перевірить, чи може користувач ввести два числа та отримати точну суму. Окремі модульні тести підтвердять інші функції калькулятора, такі як віднімання, множення та ділення.

Тестування компонентів. Тестування компонентів також називається модульним тестуванням, яке перевіряє окремі частини програми. Подібно до модульного тестування, тестування компонентів оцінює частину програмного забезпечення ізольовано від більш широкої системи. Різниця між модульним тестуванням і тестуванням компонентів полягає в тому, що перше виконується розробниками у форматі білого ящика для перевірки виконання програмних модулів, а останнє виконується тестувальниками у форматі чорного ящика для перевірки окремих об’єктів або частин програми. програмне забезпечення. Якщо інші компоненти програмного забезпечення покладаються на компонент, який тестується, спеціаліст з контролю якості може використовувати заглушку та драйвер для імітації взаємодії між цими залежними компонентами.

Приклад тестування компонентів: мобільний додаток для банківських послуг включає можливість запланувати зустріч з банківським фахівцем. Заглушка надає змодельований профіль користувача, а водій надає змодельований розклад доступних зустрічей. У цьому прикладі функціонального тестування середній компонент — той, що тестується — визначає місцезнаходження користувача за допомогою GPS і відображає місцеві банківські центри, з яких він може вибрати. Тестуючи цей компонент ізольовано, тестер може переконатися, що служба геолокації працює правильно та відображає точний список найближчих місць.

Тестування на дим. Димове тестування, тип приймального тестування, забезпечує первинну перевірку стабільності нової збірки програмного забезпечення та її критичної функціональності. Якщо тести на дим пройдуть, конструкція може пройти подальші випробування. Димове тестування, яке також називають тестуванням перевірки збірки, часто перевіряє, чи відповідає нова чи критична функціональність своїй цілі. Якщо тести не проходять, як кажуть, «де дим, там і вогонь», і потрібна додаткова робота розробника.

Приклад тестування на дим: веб-додаток для страхової компанії додає сторінку стану претензій. Тестувальники застосовували б тести на дим, щоб переконатися, що існуюча збірка працює на фундаментальному рівні, наприклад, чи може користувач успішно ввійти в систему, перейти на сторінку стану претензій і отримати статус конкретної претензії без збоїв або збоїв програми.

Перевірка на осудність. Тип регресійного тестування, професіонали з контролю якості виконують тестування на працездатність нових версій стабільних збірок, щоб перевірити нову функціональність або виправлення помилок. Хоча це схоже на димове тестування тим, що обидва забезпечують перевірку, чи готова збірка до додаткового тестування, тестування на працездатність не має сценаріїв і спеціально націлено на область, яка зазнала змін коду.

Приклад перевірки здоров’я: веб-сторінка постачальника послуг телемедицини повертає помилку 404 для своєї сторінки психічного здоров’я. Розробники виправляють проблему, а потім передають збірку для тестування. Фахівець з контролю якості виконує перевірку працездатності, щоб визначити, чи працюють основні функції та навігація для цієї конкретної сторінки належним чином.

Регресійне тестування. Те, що функціональні тести проходять один раз, не означає, що вони пройдуть завжди. Коли розробники фіксують новий код або змінюють функцію, ви запускаєте регресійні тести, щоб переконатися, що програмне забезпечення все ще працює належним чином. Регресійне тестування допомагає підтримувати стабільний продукт під час внесення в нього змін. Регресійні тести часто автоматизовані.

Приклад регресійного тестування: роздрібний продавець одягу додає можливість оплачувати бонусними балами клієнтів у своєму мобільному додатку. Тестувальники можуть виконувати регресійні тести інших існуючих функцій, таких як можливість оплати кредитними та подарунковими картками, щоб переконатися, що всі форми оплати працюють правильно.

Інтеграційне тестування. Інтеграційне тестування часто виконується разом із модульним тестуванням. За допомогою інтеграційного тестування спеціалісти з контролю якості перевіряють, що окремі модулі коду належним чином працюють разом як група. Багато сучасних додатків працюють на мікросервісах, автономних додатках, які призначені для вирішення певного завдання. Ці мікросервіси повинні мати можливість спілкуватися один з одним, інакше програма не працюватиме належним чином. За допомогою інтеграційного тестування тестувальники забезпечують безперебійну роботу цих компонентів і їх взаємодію.

Приклад тестування інтеграції: компанія кредитної картки містить сторінку, на якій клієнт може подати запит на збільшення кредиту, що є окремою базою коду від функції входу. Тестувальники можуть виконувати інтеграційні тести, щоб переконатися, що система запам’ятає користувача після того, як вони перейдуть на сторінку збільшення кредиту, і знову після успішного запиту.

Тестування API. Інтерфейси прикладного програмування з’єднують різні програми або системи, і вони стають популярнішими, оскільки споживачі очікують, що програми взаємодіють. За допомогою тестування API тестувальники підтверджують, що з’єднання та відповіді API функціонують належним чином, включаючи те, як вони обробляють дані та дозволи користувачів.

Приклад тестування API: сайт бронювання подорожей може отримати дані про ціни з бази даних авіакомпанії через API. За допомогою тестування API спеціалісти з контролю якості можуть переконатися, що правильний тип даних повертається в місцевій валюті та реагує на зміни дати та розташування.

Тестування інтерфейсу користувача. За допомогою тестування інтерфейсу користувача професіонали QA взаємодіють з графічним інтерфейсом програмного забезпечення. Це включає тестування елементів керування інтерфейсу користувача, таких як кнопки, меню та введення тексту, щоб переконатися, що потік досвіду та вибрані функції є оптимальними для роботи користувача.

Приклад тестування інтерфейсу користувача: виробник носимих пристроїв створює мобільний додаток для налаштування та обслуговування продукту. У рамках тестування інтерфейсу користувача команда переконається, що необхідні поля функціонують належним чином, зображення відображаються правильно, а інформація про технічне обслуговування з’являється на інформаційній панелі програми після використання.

Тестування системи. За допомогою тестування системи фахівці з контролю якості тестують програмне забезпечення повністю, як цілісний продукт. За допомогою такого типу функціонального тестування тестувальники перевіряють повний і інтегрований пакет програмного забезпечення, щоб переконатися, що він відповідає вимогам. За потреби тестувальники можуть надати відгук про функціональність та продуктивність програми чи веб-сайту без попереднього знання того, як вони були запрограмовані. Це допомагає командам розробляти тестові приклади для подальшого використання. Тестування системи також називають наскрізним тестуванням.

Приклад тестування системи. Виробник автомобілів виробляє розважальну систему для автомобіля, яка надає користувачам функціональні можливості для голосового керування, GPS, відеопрогравача, підключення Bluetooth, підключення мобільних телефонів, підтримки сенсорного екрана та клімат-контролю. Тестувальники оцінюють всі ці функції окремо, але вони також повинні протестувати їх як цілісну систему, щоб забезпечити сумісність і хороший досвід роботи з користувачем.

Тестування білого ящика. Коли внутрішня інфраструктура, код і дизайн програмного забезпечення видимі для розробника або тестувальника, це відноситься до тестування білого ящика. Цей підхід включає в себе різні типи функціонального тестування, включаючи модульне, інтеграційне та системне тестування. У підході до тестування «білого ящика» організація перевіряє кілька аспектів програмного забезпечення, наприклад, попередньо визначені вхідні дані та очікувані результати, а також гілки рішень, цикли та оператори в коді.

Приклад тестування білого ящика. У цьому прикладі функціонального тестування розглянемо наскрізний тест для клієнта, який додає платіжну інформацію до програми роздрібного продавця. Розробники та тестувальники проводитимуть тести у форматі білого ящика, щоб переконатися, що конфіденційні дані, такі як номер кредитної картки, зберігаються у спосіб, що відповідає стандарту PCI. Тести білого ящика також можуть гарантувати, що інформація про закупівлю надходить до алгоритму машинного навчання для генерування прогнозів, покупка правильно генерує винагородні бали, а система інвентаризації віднімає товари з кількості запасів.

Тестування чорного ящика. На відміну від тестування білого ящика, тестування чорного ящика включає тестування на системі, де внутрішній код, шляхи та інфраструктура не видно. Таким чином, тестувальники використовують цей метод для перевірки очікуваних результатів щодо конкретних вхідних даних. Будь-який момент, коли фахівець з контролю якості не вивчає код перед тестуванням, можна вважати чорним ящиком. За допомогою тестування «чорного ящика» організація може протестувати програмне забезпечення так само, як клієнти. Тестування чорного ящика охоплює різноманітні типи нефункціональних і функціональних тестів, залежно від мети тесту.

Приклад тестування чорної скриньки: на платформі потокового телебачення тестувальник перемикає функцію пошуку та виконує пошук конкретного актора. Потім тестувальник перевіряє, що функція пошуку повертає логічні (очікувані) результати, включаючи телевізійні шоу, в яких з’являвся актор, або пропоновані назви, подібні до відомих робіт цього актора.

Приймально-здавальне випробування. Мета приймального тестування полягає лише в тому, щоб переконатися, що кінцевий користувач може досягти цілей, встановлених у бізнес-вимогах. Замість того, щоб зосередитися на функціональності конкретних функцій, тестування на приймання включає в себе перевірку повної роботи програми та наскрізного досвіду. Приймальне тестування користувача (UAT) і бета-тестування, підмножини приймального тестування, залучають кінцевих користувачів до проведення аналізу готового продукту. Звідти організація може оцінити цей відгук і внести зміни.

Приклад придатного тестування: компанія, що займається програмним забезпеченням, випускає продукт, який дозволяє користувачам керувати великими даними. Після випуску нової версії програмного забезпечення група найбільш значущих користувачів цієї компанії проводить тестування користувача, щоб визначити, чи відповідає нова версія їхнім основним потребам і як продукт можна покращити.

Альфа-тестування. Інша частина приймального тестування, альфа-тестування, використовує внутрішніх членів команди для оцінки продукту. Ці члени команди повинні знати про проект, але не брати безпосередню участь у його розробці чи тестуванні. Якщо деякі збірки все ще можуть бути дещо нестабільними, альфа-тестування забезпечує безпосередню підгрупу тестувальників для усунення основних помилок до того, як програмне забезпечення побачать зовнішні користувачі.

Приклад альфа-тестування. У цьому прикладі функціонального тестування постачальник ігор казино випускає нову версію свого додатка, яка включає відеопокер. Організація складає багатофункціональну групу внутрішніх користувачів, які перевіряють, чи правильно працює додаток на їхніх пристроях і як можна покращити користувацький досвід.

Бета-тестування. Після того, як внутрішня команда тестує продукт і виправляє помилки, відбувається бета-тестування з вибраною групою кінцевих користувачів. Бета-тестування служить плавним запуском, що дає змогу отримати відгуки від реальних користувачів, які раніше не знають програму. Бета-тестування дає змогу збирати відгуки від неупереджених користувачів, які можуть взаємодіяти з продуктом інакше, ніж ви планували, можливо, виявляючи критичні невідомі помилки перед випуском для широкої бази користувачів.

Приклад бета-тестування: мережа ресторанів випускає нову мобільну систему замовлення та самовивезення. Перш ніж компанія випустить функціональність для всіх користувачів мобільних додатків, вона тестує додаток з невеликою кількістю відданих клієнтів і надає їм винагороду за участь.

Виробничі випробування. Щойно продукт стає загальнодоступним, він перебуває в робочому середовищі, де будь-який користувач може взаємодіяти з ним будь-яким чином — ви більше не можете контролювати все, від середовища тестування до кількості людей, які використовують продукт. Виробниче тестування є частиною безперервного тестування та тестування перемикання вправо, яке намагається якнайшвидше виявити та сортувати дефекти, про які повідомляють користувачі. Випробовуючи у виробництві, організація може тестувати за межами сценарних тестових випадків у різноманітному середовищі. За допомогою виробничого тестування організація може підтвердити функціональність і стабільність продукту.

Приклад виробничого тестування: виробник фітнес-обладнання може відстежувати дефекти, про які повідомляють користувачі, і показники пристрою, щоб переконатися, що його підключені до Інтернету бігові доріжки, еліптичні тренажери та тренажери для підйому по сходах працюють належним чином — під час доставки та постійно.

Тестування доступності. Ваші цифрові продукти повинні працювати для всіх користувачів, а не лише для певної групи користувачів. Це робить тестування доступності обов’язковим для всіх організацій. Компанії, які дотримуються федеральних стандартів доступності, відкривають додаткові канали доходу та зменшують ризик судових позовів.

Тестування AI. Багато роботи — і даних — витрачається на навчання алгоритмів машинного навчання. Але це не зупиняється на досягнутому. Тестування штучного інтелекту за різними критеріями гарантує, що системи AI та ML функціонують належним чином.

Тестування подорожей клієнта. Майже кожна компанія зв’язується зі споживачами кількома каналами (наприклад, мобільний, веб-сайт, соціальні мережі, в магазині). Таким чином, тестування клієнтського шляху або омніканальне тестування є важливим для наскрізного досвіду. Тестування клієнтської подорожі використовує реальних людей для перевірки досвіду з одного каналу на інший, щоб забезпечити стабільну цифрову якість та безперервність досвіду.

Тестування платежів. Для будь-якої транзакційної системи тестування платежів має вирішальне значення для успіху. За допомогою тестування платежів користувачі підтверджують вибрані способи оплати, такі як кредитні картки, дебетові картки, мобільні гаманці та платформи P2P. Тестування платежів гарантує, що валюта може безперешкодно переходити від споживача до бізнесу, а транзакція може бути оброблена.

Тестування в роумінгу. Коли ви застосовуєте вищезгадані типи функціонального тестування в одному місці, пам’ятайте, що немає гарантії, що програма буде — або повинна — виконувати те саме в іншому місці. У сучасному світі, який стає все більш мобільним, розробники та тестувальники повинні переконатися, що пристрої добре працюють за кордоном і в різних мережевих умовах.

Тестування безпеки. Усі програми та веб-сайти вимагають надійного захисту, щоб підтримувати довіру споживачів та захистити дані та інтелектуальну власність. Існують численні автоматичні сканування та оцінки, які кожна компанія повинна виконувати в рамках тестування безпеки, але активні елементи тестування також повинні залучати до тестування людську сторону.

Тестування користувацького досвіду. На відміну від прийнятного тестування користувачами, тестування користувацького досвіду враховує думку користувачів. Важливо мати функціональні цифрові продукти, але також важливо мати продукти, якими клієнти можуть легко користуватися — і, в ідеалі, їм подобається користуватися.

Голосове тестування. У міру розвитку пристроїв розвиваються й механізми, за допомогою яких користувачі взаємодіють із ними. Голосові програми вносять новий рівень складності в тестування програмного забезпечення через численні змінні, пов’язані з людським голосом. Голосове тестування залучає реальних людей для перевірки введених даних на різних мовах, діалектах, емоційних станах тощо.

## 1.4 Висновки до розділу

В даному розділі було розглянуто важливість дистанційного навчання, засоби автоматичного оцінювання практичних завдань та види тестування ПЗ.

Дистанційне навчання все ще має великий невикористаний потенціал, щоб охопити студентів там, де вони є, і по-новому поєднати викладачів і учнів. Від підвищеної гнучкості до нових стилів навчання, здається, що майбутнє навчання буде таким же різноманітним за часом і місцем, як і мислення.

# 2 Алгоритми побудови засобів автоматичного оцінювання завдань

## 2.1 Модель лабораторного практикуму дистанційного курсу

Електронне оцінювання – це нова ідея в області дидактики, яка має на меті забезпечити автоматичне оцінювання іспитів за допомогою штучного інтелекту та комп’ютерних систем розпізнавання, особливо OCR та розуміння зображень. Програмне забезпечення для електронного оцінювання – це система, яка дає змогу оцінювати екзаменаційні бланки відповідей екзаменатором на екрані комп’ютера, а не читати паперові документи. У нашій попередній роботі ми розробляли вимоги до такої системи. Перевагою системи електронного оцінювання є краща організація екзаменаційної сесії, оскільки вона виконує сегментацію бланків відповідей на окремі завдання, розподіляє завдання між обраними екзаменаторами, які спеціалізуються у своїй області (наприклад, фізика), оцінює завдання та надсилає їх назад до екзаменаційної комісії та перевіряє оцінку, збирає статистичні дані та представляє результати.

Електронне оцінювання може покращити якість іспиту шляхом:

* забезпечення повної анонімності оцінювання завдань;
* придушити ефект ореолу – спосіб вирішення одним учнем не впливає на оцінку наступних завдань;
* постійне поширення потенційних помилок при оцінюванні – завдання для оцінювання екзаменаторам призначаються випадковим чином.

Завдання, які фігурують в екзаменаційних роботах, можуть бути сформульовані як «закриті» або «відкриті». Закрите завдання — це тест з одним або кількома відповідями, або воно складається із заповнення пропусків конкретними словами чи цифрами. Одна з неправильних відповідей, представлених як вибір у тесті з кількома варіантами, відома як відволікаючий, а правильна відповідь називається верстрактором.

Комп’ютерна оцінка такого завдання нескладна. У випадку електронної форми це просто порівняння результату з ключовою відповіддю. У випадку паперового бланка відсканований аркуш обробляється для розпізнавання певних позначок (фігур «X» або «V»), які учень поставив на бланку.

Набагато складніше автоматичне оцінювання відкритої задачі, оскільки набір правильних рішень може бути нескінченним. Крім того, неповні або неточні рішення мають оцінюватися зниженим балом.

Більшість алгоритмів розпізнавання математичних формул працюють у два кроки:

1. виявлення окремих символів шляхом сегментації зображення та індексації об'єктів із застосуванням методів розпізнавання образів;

2. структурний аналіз зв'язків (наприклад, просторових) між виділеними сегментами.

Для розпізнавання деяких символів використовується класичний підхід. Спочатку виконується сегментація, і класифікатор присвоює символи окремим об’єктам. Для скорочення кроків обробки використовується метод прихованих марківських моделей. Структурний аналіз виразу (і виправлення у разі помилок розпізнавання) можливий за допомогою застосування двовимірної контекстно-вільної граматики.

Тип відкритого завдання, який ми враховуємо в цій роботі, — це накреслення графіка заданої математичної функції. Цей тип завдання часто зустрічається на випускних іспитах з математики в середній школі або зрілості. Учень повинен намалювати графік у друкованій системі координат. Оскільки графік малюється вручну, вчитель перевіряє, чи перетинає намальована крива деякі характерні точки решітки. У разі складного графіка (кілька кривих або ліній) крива може бути переривчастою, і учень повинен позначити закінчення відрізків.

Для сегментації відсканованих екзаменаційних листів ми використовуємо метод відповідності шаблону, який використовує алгоритм крос-кореляції. Як шаблон запиту був використаний невеликий фрагмент зображення, яке з’являється у верхній частині області кожного завдання).

Алгоритм виконується за такими кроками:

1. Знайти на зображенні аркуша (1...*M*, 1...*N*) усі локальні максимуми для узгодження шаблону. Задайте координати точок як (x1, y1) ... (xk, yk).

2. Для кожного локального максимуму (xi, yi) знайти інший локальний максимум (*x*j, *yj*), для якого xj>xi.

Якщо точка знайдена, це означає, що область зображення, що містить завдання, досягає значення xj. Якщо ні, це означає, що (xi, yi) знаходиться в правому стовпці, а область досягає правого краю зображення (M).

3. Для кожного місця розташування локального максимуму (xi, yi) шукайте найближчий локальний максимум (xj, yj), для якого yj>yi та |xj-xi|<eps, eps>0 . Якщо точка знайдена, область досягає значення yj. Якщо ні – (xi, yi) є останнім завданням на сторінці, а область досягає нижнього краю зображення (N).

Для групи студентів було дано два завдання. Накреслити графік лінійної функції, заданої як y = 2x. Оцінка за завдання була 0 або 1. Намалюйте графік функції квадрата, заданої формулою y = x2 – 4x + 4. Розв’язання оцінили від 0 до 2.

Алгоритм використовує три зображення для кожного оцінювання:

I1 – зображення порожньої системи координат

I2 – скановане зображення модельного рішення (вчитель малює графік на екзаменаційному аркуші)

I3 – відскановане зображення рішення студента

Етапи алгоритму:

1. За допомогою методу взаємної кореляції знайти найкраще відповідність зображень I1 та I2. Отримаємо Ir = I1 – I2. Використовуючи математичні морфологічні фільтри, видалити спотворення (наприклад, фрагменти системи координат, що залишилися від недосконалої геометрії сканера) з Ir і залиште лише криву графіка . Використовуйте алгоритм витончення, щоб отримати лінію товщиною 1 піксель.

2. Виконайте таку ж обробку для зображень I3 та I1. Отримайте зображення. Позначте St як зв'язаний компонент у зображенні It, що містить криву зображення.

3. Для Ir зробити розширення (елементом структури є коло r = 10 px). Призначте його як Ird і Srd як зв’язаний компонент зображення.

4. Обчислити, скільки пікселів контрольної діаграми Srd відповідає пікселям тестової діаграми St.

Може статися, що кількість пікселів P2 дуже мала (учень намалював лише кілька крапок, а не весь графік). Ми порівнюємо кількість пікселів у тестовому та еталонному розширених графіках.

Якщо умова не виконується, то графік недійсний (ставимо від’ємне число). У всіх інших випадках P знаходиться в діапазоні (0, 1) і описує рівень відповідності між еталонними графіками та графіками запиту.

Оцінка функції квадрата є дещо складнішою, оскільки учень не може намалювати криву так само точно, як пряму. Під час реального іспиту він отримує 1 бал за обчислення екстремуму параболи і другий бал, якщо графік перетинає принаймні три точки «ящика». У випадку функції y = x2 + 4x + 4 ми маємо мінімум у (2, 0), а точки ящика – (1, 1), (3, 1), (0, 4), (4, 4) .

Алгоритм складається з таких кроків:

1. З еталонного графіка витягнити зображення, що містить екстремум параболи з околицями 0,5. Його відповідність графіку запиту оцінюється за допомогою алгоритму градації лінійної функції.

2. Перевірте всі 4 точки ящика. Для кожної точки картки отримайте зображення. За допомогою алгоритму взаємної кореляції знайдити найкращі збіги з графіком запиту.

Якщо крайня і принаймні 3 точки ящика намальовані правильно (рівень відповідності перевищує експериментально встановлений рівень), робота оцінюється як 2 бали та 1 бал лише за крайню.

Модель нашого практикуму передбачатиме собою систему, яка буде мати завдання, які студент має виконати, термінал, на якому студент буде виконувати завдання, а також систему оцінювання, яка буде перевіряти виконані завдання студента.

## 2.2 Модель завдань для онлайн курсу

В якості завдань для онлайн курсу ми виберемо завдання по BАSH-скриптах, які будуть автоматично перевірятися. Для цього будуть створені завдання по командам, які представлені нижче.

Найбільш часто використовувані команди:

ls

ls, ймовірно, перша команда, яку кожен користувач Linux ввів у своєму терміналі. Він дозволяє вам перерахувати вміст потрібного каталогу (поточний каталог за замовчуванням), включаючи файли та інші вкладені каталоги.

Він має багато параметрів, тому було б добре отримати допомогу за допомогою прапорця --help. Цей прапор повертає всі прапори, які ви можете використовувати з ls.

alias

Команда alias дозволяє визначити тимчасові псевдоніми в сеансі оболонки. Створюючи псевдонім, ви вказуєте своїй оболонці замінити слово серією команд.

Якщо ви хочете перерахувати всі псевдоніми, які у вас є в сеансі оболонки, ви можете запустити команду alias без аргументів.

unalias

Як випливає з назви, команда unalias має на меті видалити псевдонім із вже визначених псевдонімів.

pwd

Команда pwd означає «друк робочого каталогу», і вона виводить абсолютний шлях до каталогу, у якому ви перебуваєте. Наприклад, якщо ваше ім'я користувача — «vasyl», а ви перебуваєте в каталозі «Документи», його абсолютний шлях буде таким: /home/vasyl/Документи.

cd

Команда cd дуже популярна, поряд з ls. Він відноситься до «каталогу зміни» і, як випливає з його назви, перемикає вас до каталогу, до якого ви намагаєтеся отримати доступ.

cp

Копіювати файли та папки безпосередньо в термінал Linux настільки легко, що іноді це може замінити звичайні файлові менеджери.

Щоб використовувати команду cp, просто ввести її разом із вихідними та цільовими файлами.

rm

Ви можете використовувати команду rm для видалення файлів і каталогів. Однак будьте обережні під час його використання, оскільки відновити файли, видалені таким чином, дуже важко (але не неможливо).

mv

Ви використовуєте команду mv для переміщення (або перейменування) файлів і каталогів у вашій файловій системі.

Щоб використовувати цю команду, потрібно ввести її назву разом із вихідними та цільовими файлами.

mkdir

Щоб створити папки в оболонці, ви використовуєте команду mkdir. Просто вказати назву нової папки, переконатися, що вона не існує, і готово до роботи.

man

Іншою важливою командою Linux є man. Він відображає сторінку посібника будь-якої іншої команди (якщо вона є).

touch

Команда touch дозволяє оновлювати час доступу та модифікації вказаних файлів.

chmod

Команда chmod дозволяє швидко змінити режим файлу (дозволи). З нею доступно багато варіантів.

Основні дозволи, які може мати файл:

r (читати)

w (писати)

x (виконати)

Один із найпоширеніших випадків використання chmod — зробити файл виконуваним користувачем.

./

Можливо, нотація ./ не є командою сама по собі, але її варто згадати в цьому списку. Це дозволяє вашій оболонці запускати виконуваний файл із будь-яким інтерпретатором, встановленим у вашій системі, безпосередньо з терміналу. Більше немає подвійного клацання файлу в графічному файловому менеджері!

Наприклад, за допомогою цієї команди ви можете запустити сценарій Python або програму, доступну лише у форматі .run, наприклад XAMPP. Запускаючи виконуваний файл, переконайтеся, що він має дозволи на виконання (x), які ви можете змінити за допомогою команди chmod.

exit

Команда exit робить саме те, що передбачає її назва: за допомогою неї ви можете завершити сеанс оболонки і, у більшості випадків, автоматично закрити термінал, який ви використовуєте.

sudo

Ця команда розшифровується як «суперкористувач робити», і вона дозволяє вам діяти як суперкористувач або користувач root під час виконання певної команди. Таким чином Linux захищає себе та запобігає користувачам від випадкової зміни файлової системи машини або встановлення невідповідних пакетів.

Sudo звичайно використовується для встановлення програмного забезпечення або редагування файлів за межами домашнього каталогу користувача

shutdown

Як ви можете здогадатися, команда вимкнення дозволяє вимкнути вашу машину. Однак його також можна використовувати для зупинки та перезавантаження.

htop

htop — це інтерактивний засіб перегляду процесів, який дозволяє керувати ресурсами вашої машини безпосередньо з терміналу.

unzip

Команда unzip дозволяє витягти вміст файлу .zip з терміналу. Знову ж таки, цей пакунок може бути не встановлений за замовчуванням, тому переконайтеся, що ви встановили його за допомогою менеджера пакетів.

apt, yum, pacman

Незалежно від того, який дистрибутив Linux ви використовуєте, швидше за все, ви використовуєте менеджери пакетів для встановлення, оновлення та видалення програмного забезпечення, яке ви використовуєте щодня.

Ви можете отримати доступ до цих менеджерів пакетів через командний рядок, і ви будете використовувати той чи інший залежно від дистрибутиву, на якому працює ваша машина.

echo

Команда echo відображає певний текст у терміналі.

cat

Cat, скорочення від «concatenate», дозволяє створювати, переглядати та об’єднувати файли безпосередньо з терміналу. В основному він використовується для попереднього перегляду файлу без відкриття графічного текстового редактора.

ps

За допомогою ps ви можете подивитися на процеси, які виконує ваша поточна сесія оболонки. Він друкує корисну інформацію про програми, які ви запускаєте, як-от ідентифікатор процесу, TTY (TeleTYpewriter), час і назву команди.

kill

Це дратує, коли програма не відповідає, і ви не можете закрити її жодним чином. На щастя, команда kill вирішує цю проблему.

Простіше кажучи, kill надсилає TERM або сигнал знищення процесу, який припиняє його.

Ви можете припинити процеси, ввівши PID (ідентифікатор процесів) або двійкове ім’я програми.

ping

ping — найпопулярніша утиліта мережевого терміналу, що використовується для перевірки підключення до мережі. ping має масу опцій, але в більшості випадків ви будете використовувати його для запиту домену або IP-адреси.

vim

vim — це безкоштовний термінальний текстовий редактор з відкритим вихідним кодом, який використовується з 90-х років. Це дозволяє редагувати звичайні текстові файли, використовуючи ефективні комбінації клавіш.

Деякі люди вважають його складним у використанні — вихід із Vim — одне з найпопулярніших питань StackOverflow — але як тільки ви звикнете до нього, він стане вашим найкращим союзником у командному рядку.

Для генерації тестових завдань ми використаємо один із алгоритмів генерування тестових завдань.

Алгоритм генерування завдань на основі параметризації

Метод генерування тестових завдань, заснований на параметризації, дозволяє проводити багаторазове тестування, щоразу отримуючи різні зразки завдань.

Перед програмною реалізацією завдань виконується математичне моделювання кожного з них. Визначаються діапазони, в яких необхідно генерувати значення параметрів, що входять до завдання. Нехай A — набір завдань, згенерованих у вибірці. Тоді елементи цієї множини ai можна представити у вигляді функціональної залежності від варіанту породження умови задачі di, множини значень параметрів, що входять до задачі P, зв'язків між цими параметрами E і множини правильні відповіді R. У математичних записах це можна записати так:



де *n* — кількість завдань у вибірці.

Слід зазначити, що значення функції ai визначені лише для таких множин Pi, Ei, для яких множина R не буде порожньою і не матиме нескінченної кількості елементів. Іншими словами, створена задача повинна мати скінченну кількість розв’язків. Таким чином, завданням розробника є створення такого алгоритму генерування задач, який би максимізував кількість можливих значень аргументів di і найбільшу кількість різних варіантів генерування множини Pi.

Аргументи E і R не впливають на збільшення кількості варіантів генерування завдань. При достатньо великій потужності множини P можна досягти результату, коли кількість можливих варіантів генерування завдання становитиме тисячі або навіть десятки тисяч. Це набагато більше, ніж, наприклад, у випадку використання фасетних таблиць. Такий підхід до створення завдань гарантує, що кожен учень отримає унікальний набір завдань. Ймовірність отримати той самий набір невелика і нею можна знехтувати.

Класичний підхід до генерації обчислювальних тестових завдань

Цей метод заснований на використанні двох логічно незалежних елементів: математичного алгоритму розв’язування задачі на основі заданих вхідних даних (розв’язувача) та генератора вхідних даних. Пристрій розв’язки являє собою функцію зі змінним набором вхідних параметрів залежно від конкретного завдання. Генератор реалізує автоматичне формування «прийнятних» вхідних даних на основі певних правил. Ці правила описують зв'язки між вхідними даними. Генератор повертає набір вхідних даних, а пристрій розв’язки повертає набір відповідних результатів.

Дані, згенеровані вищеописаним методом, надходять на вхід генератора запитань разом із шаблоном тексту завдання та структурою (шаблоном) формату XML, якщо він передбачений. Генератор запитань вставляє згенеровані вхідні дані в текст завдання, а отриманий результат – у набір варіантів відповідей. Також такий підхід передбачає реалізацію алгоритму формування неправильних відповідей, якщо цього вимагає специфіка тестового завдання. Невірні відповіді, а також вхідні дані також можуть бути вибрані випадковим чином з бази даних.

З точки зору доцільності, при виборі завдань для автоматизації в першу чергу необхідно сформувати такі завдання, розробка яких «вручну» займає багато часу та зусиль. Основні характеристики наступних завдань:

- Необхідність створення картини. Часто візуалізація дерев, графіків чи інших структур вимагає навичок роботи з певними програмними інструментами і займає деякий час.

- Великий набір вхідних даних. Іноді знайти потрібну інформацію буває непросто.

- Необхідність створення великої кількості різноманітних наборів вхідних даних. За допомогою комп’ютерних засобів можна генерувати максимум унікальних комбінацій вхідних даних, що дозволяє отримати набагато більше різноманітних завдань, ніж при створенні їх «вручну».

— Підрахунок результату займає багато часу. Отримання рішення на комп’ютері відбувається практично миттєво, що забезпечує значну економію часу викладача.

Для описаних вище характеристик підходять багато задач, зокрема задачі з деревами, графіками і т. д. Автоматичне генерування таких завдань значно полегшить роботу вчителя за рахунок економії часу та відсутності потреби в установці додаткового програмного забезпечення.

## 2.3 Розроблення алгоритмів автоматичного тестування завдань

У індустрії програмного забезпечення автоматизація відіграє важливу роль для підвищення ефективності тестування організації тестування програмного забезпечення. Щоб забезпечити задоволеність покриттям тестуванням і, таким чином, знизити ризик, проекти використовують більше людей для ручного тестування або використовують інструменти чи методи автоматизації для підвищення рівня автоматизації тестування. Вибір вибору залежить від скорочення часу проектного циклу або скорочення часу на тестування. Далі розглядаються кроки автоматизації програмного забезпечення та пояснюється алгоритм вирішення проблем, що виникають при автоматизації управління системою різних сервісів, таких як DHCP, DNS, брандмауер тощо..

Досить поширеним є те, що багатьох тестувальників автоматизації кидають на роботу з автоматизації, не маючи належних рекомендацій щодо процесу тестування автоматизації. Вони просто «занурюються» і починають автоматизувати тестові випадки, не замислюючись про будь-який процес чи стратегію. Звичайно такий підхід призводить до ситуації, яка говорить: «Ми багато інвестували в автоматизацію та кількість тестувальників, виділених на повний робочий день для автоматизованого тестування. Після X місяців ми все ще не бачимо жодного покращення часу виконання тестового циклу. Насправді, тестування, здається, займає більше часу! ”. Ми повинні мати чітке розуміння нашого проекту та процесу автоматизації, який ми збираємося впровадити. Також ми повинні мати належну стратегію тестування у виборі фреймворка та виконання. Деякі з факторів, які необхідно враховувати під час визначення стратегії тестування:

* автоматизація тестування не є побічною роботою, але це повний робочий день.
* розглядайте діяльність з розробки тестів і тестову структуру як абсолютно окремі сутності.
* використовуйте незалежну від програми фреймворк.
* тестовий каркас повинен легко розширюватися та підтримуватися
* дизайн тесту або словник стратегії повинен бути незалежним від фреймворка.
* дизайн або стратегія тестування повинні позбавити більшість тестувальників складнощі тестової рамки.

На рисунок 2.1 показані різні етапи процесу автоматизації тестування. Звичайно керівник/менеджер з автоматизації відповідатиме за вибір інструменту, розробку інструментів та діяльність з обслуговування каркаса. Інженер-тестувальник відповідатиме за створення, розгортання та виконання сценаріїв

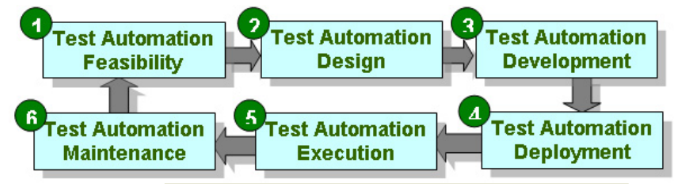


Рисунок 2.1 - Етапи автоматизації

Техніко-економічне обґрунтування

Цей етап вирішить, чи потрібно нам автоматизувати проект чи ні. Збій на цьому етапі може мати більший вплив на виконання проекту. На рисунок 2.2 показаний процес техніко-економічного обґрунтування. Нижче наведено деякі з факторів, які враховуються на етапі виконання.

* Вплив на проект: Пріоритет автоматизації відповідно до пріоритетів проекту та його виконання
* Вибір тестового прикладу:

1. Повторюваний тест, який потрібно виконувати на кількох збірках

2. Часто використовувані функції

3. Тести, які виконуються на кількох різних платформах

4. Тести, які займають багато зусиль і часу під час ручного тестування

* Доступність Framework: Вивчаємо різні існуючі інструменти та оцінюємо, чи можемо ми використовувати їх у наших налаштуваннях шляхом налаштування

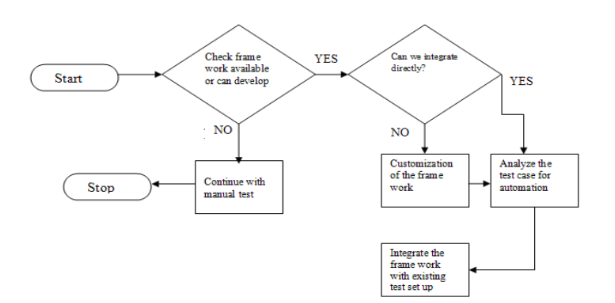


Рисунок 2.2. Процес техніко-економічного обґрунтування

На рисунку 2.3 показано, як ми розглядаємо тестові шаблони та автоматизацію тестування на різних етапах тестування програмного забезпечення.

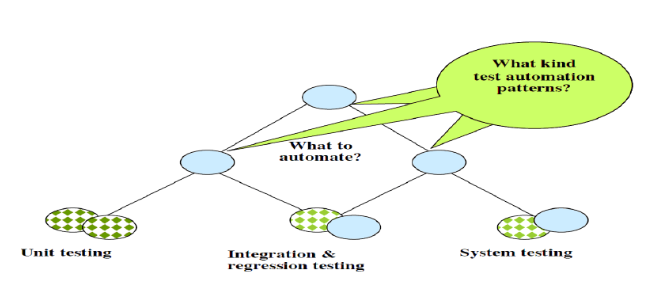


Рисунок 2.3. Завдання з автоматизації

Проектування автоматизації

Хороший дизайн тестування автоматизації розповість, як буде перевірена конкретна функція або функція. Розробник тестів розгляне такі факти:

1. Що перевіряється і як тест налаштований?

2. Які вхідні дані використовуються і звідки вони надходять?

3. Що перевіряється і де очікувані результати?

4. Які речі потрібно надрукувати?

5. Як ви знаєте, що тест пройшов чи не пройшов?

6. Зробити вихід простим і добре відформатованим. І т.д.

Розробка та розгортання автоматизації

В основному нам потрібно розробити два типи елементів на етапі розробки

* скрипти для виконання
* фреймовий робочий код (або налаштування існуючого коду)

Деякі проекти, цілі модулі можуть бути недоступними, у такому випадку нам може знадобитися розробити заглушки та драйвери для моделювання модуля. У нашій структурі розробник тестів відповідає за створення тестових наборів (тестових сценаріїв), налаштування інструменту та налаштування середовища для тестового стенду. Крім того, після розробки сценарію та коду нам потрібно регулярно перевіряти їх у інструменті керування конфігурацією та створити належну збірку для виконання. Тестувальники мають дозвіл на те, щоб приймати те саме для тестування.

Виконання та обслуговування автоматизації

На етапі тестування ми будемо виконувати тестові випадки вручну або автоматизовано. Для автоматизації відбір тестових випадків здійснюється за допомогою стратегії автоматизації. Звичайно всі тести регресії будуть переміщені в автоматизоване середовище, щоб уникнути повторного виконання вручну. В автоматизованому середовищі тестер може вибрати/скасувати вибір тестових випадків, які він хоче виконати на певній платформі. Під час виконання ми можемо дати два різні варіанти. У першому варіанті, тобто зупинка при невдачі, виконання тесту припиняється, якщо будь-який із вибраних тестів не вдасться. Але у другому варіанті, продовжити у разі відмови, дозволить виконати всі вибрані пакети, навіть якщо один пакет не вийде. Наприкінці виконання ми можемо побачити результати та журнали кожного пакета окремо. Це дасть детальний статус виконання.

Проблема автоматизації управління системою та алгоритмічне рішення

Під час автоматизації програмного забезпечення для керування системою (наприклад: брандмауери) команда автоматизації стикається з багатьма проблемами та проблемами, як-от налаштування інструменту автоматизації, зв’язок між клієнтом і сервером, доступ до вхідних файлів із сервера Unix до вікон Windows тощо. Більшість проблем стосуються технічної компетенції, крім адміністрування різних системних служб. Для вирішення проблем системних сервісів реалізовано новий алгоритм. Проблема та рішення описані таким чином.

Проблема: Керування системою означає адміністрування різноманітних системних служб, таких як DHCP, DNS, брандмауер тощо. Усі ці системні служби використовують деякі стандартизовані файли конфігурації для адміністратора, щоб налаштувати їх відповідно до своїх вимог. Зміна цих файлів вручну є виснажливим завданням. Люди використовують ПЗ для керування цими послугами. Дані, що містяться в цих файлах, відрізняються за форматом, вмістом тощо залежно від системної служби, яку вони використовують. Деякі дані можуть мати кілька випадків. Коли ми намагаємося маніпулювати даними в цьому файлі за допомогою програмного забезпечення, стає важко знайти дані, які ми шукаємо.

Рішення: Рішення полягає в тому, що система хотіла б мати деяке значення індексу/хеш-значення, пов'язане з файлом конфігурації, щоб дані, якими ми хочемо маніпулювати за допомогою програмного забезпечення, могли покладатися на значення індексу для кожного разу ідентифікувати ті самі дані (Індексація бази даних концепція).

Техніко-економічний аналіз: на етапі аналізу доцільності команда досліджувала різні варіанти вирішення проблеми. Але жодне з наявних рішень не змогло вирішити проблему. Тому команда вирішила розробити нове рішення, врахувавши вплив проекту під час впровадження нового рішення. Команда прийшла до висновку, що в результаті аналізу доцільності новий дизайн не повинен впливати на існуючі налаштування тестування, а графік впровадження відповідатиме, не впливаючи на основні віхи.

Розробка та реалізація рішення: у нас є файл конфігурації (CONF), і тепер нам потрібні унікальні значення хеш-індексів, пов’язані з даними, що зберігаються в ньому. Як і в базі даних, ми також хочемо, щоб хеш-значення були постійними. Дані у файлі потрібно розділити, щоб мати пов’язані з ними хеш-значення. Давайте розділимо файл CONF на FIELDS, де поле може бути одним із цих:

* Однобайтове символьне поле.
* Набір символів, що закінчуються пробілом/табуляцією (роздільник), називається полем слова
* Набір слів, що закінчуються новим рядком/кареткою (розділювач), називається рядковим полем
* Набір рядків, що закінчуються двома послідовними переведенням рядка/повернення каретки ( роздільник) називається полем абзацу.

Унікальні хеш-значення можуть бути згенеровані за допомогою будь-якого алгоритму генерації випадкових чисел. Зберігання та збереження хеш-значень – це проблема, яку ми хочемо вирішити тут. Давайте зробимо копію файлу CONF. Створити інший файл для зберігання хеш-значень. Файл CONF логічно розділений на послідовні ПОЛЯ, і файл хеш-ключів зберігатиме значення хеш-ключа послідовно, таким чином, що повторювані поля CONF-файлу будуть зіставлені один в один із значеннями хеш-ключа у файлі хеш-ключа. Якщо у нас є роздільник, що розбиває повторювані поля CONF-файлу, у нас буде такий самий роздільник для розділення хеш-значень у файлі хеш-ключа. Якщо хтось змінить файл CONF вручну або за допомогою програмного забезпечення, буде змінено лише оригінальний файл конфігурації. Дублікат призначений для відстеження змін, які відбулися у файлі CONF після останніх змін. Модифікація у файлі CONF може бути створенням, видаленням або зміною FIELD. Та сама зміна не знайшла б відображення в дублікаті, і, отже, ми можемо визначити, що була зміна і де вона відбулася. Після визначення змін дублікат файлу CONF буде оновлено щодо файлу CONF, а файл хеш-ключа також буде відповідно оновлено.

Для реалізації рішення використовуються в основному три функції:

1. Ініціалізація середовища: ця функція візьме шлях до файлу CONF як вхідну інформацію та зробить копію файлу CONF. Це також створить файл хеш-ключа, який міститиме хеш-значення для полів у файлі DUP. На рисунок 2.4 показано блок-схему для ініціалізації

2. Відстеження змін: Основна мета цієї функції — відстежувати зміни, що відбуваються у файлі CONF, і, отже, оновлювати хеш-значення. Операція оновлення може бути операцією створення, видалення або зміни. Під час операції створення нове поле буде вставлено в файл конфігурації. Зміна буде відслідковуватися щодо дубліката файлу, щоб знайти розташування поля. Відповідно до цього поля новий унікальний хеш-ключ буде вставлений у файл хеш-ключа, а потім дублікат файлу буде оновлено до його оригіналу. Подібним чином буде оброблено операцію видалення. Для операції модифікації, оскільки вона змінює вміст поля, зміна хеш-значень не буде виконана. На рисунку2.5 показана блок-схема змін треку

3. Функція підтримки: функція підтримки допоможе отримати хеш-значення для поля або розташування у файлі CONF, а також отримати поле або розташування для хеш-значення з хеш-файл.

Нижче наведені блок-схеми показують, як працює реалізація.

Нижче наведено деякі переваги автоматизації тестування, які ми отримали в нашому прикладі

* Високий охоплення регресійним тестуванням без додавання додаткових ресурсів
* Підвищення швидкості виходу продукту на ринок за рахунок скорочення часу на тестування
* Покращення продуктивності
* Детальні журнали тестування
* Запуск сценаріїв на кількох платформах
* Швидкий, надійний, комплексний і багаторазовий
* Підвищення економічної ефективності

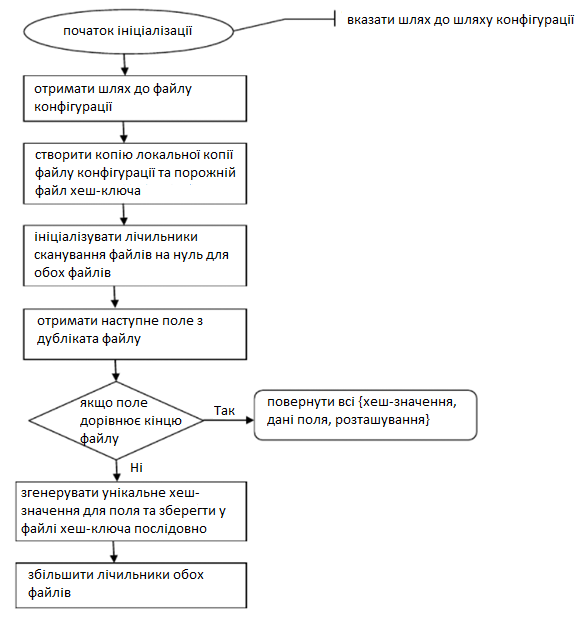


Рисунок 2.4 – Ініціалізація

Фактори успіху автоматизації тестування

Успіх автоматизації тестування залежить від прямих і непрямих факторів організації тестування.

Прямі фактори успіху:

* процес тестування;
* керування тестуванням ;
* розмежування тестового об’єкта ;
* визначення тестових даних;
* тестові дані та визначення тестових даних;
* тестова інфраструктура та середовище;
* інструменти тестування;
* співробітники.

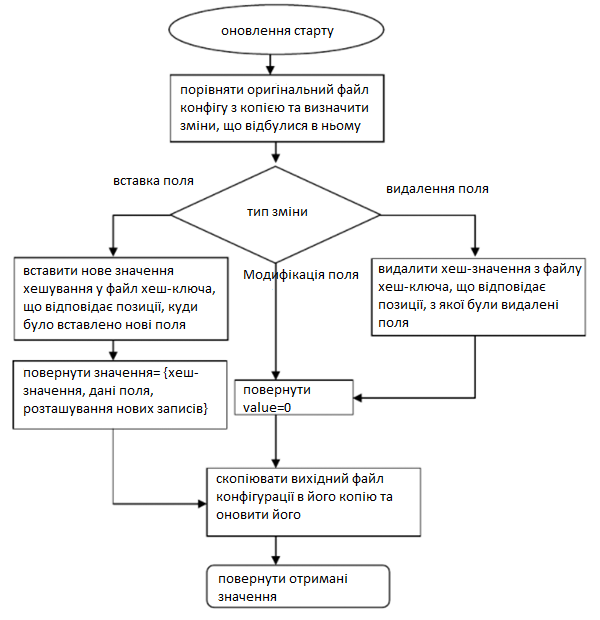


Рисунок 2.5 Відстеження змін

Непрямі фактори успіху:

* управління конфігурацією
* управління змінами
* управління дефектами
* випуск управління
* управління вимогами.

При створенні автоматизованих систем тестування необхідно вирішити наступні проблеми:

1. Вибір методу створення тестових завдань (автоматична генерація тестів або введення тестів вручну).

2. Метод інтерпретації результатів.

Вибір способу створення тестових завдань:

1. Імпортуйте файл певного формату (наприклад, XML) з тестовими завданнями. Цей метод реалізовано в системі Model.

2. Заповнення форми вчителем з ручним введенням тексту завдання та варіантів відповідей. Наприклад, система TestWizard 1.7.

3. Автоматичне формування тестових завдань на основі вбудованих математичних алгоритмів, аналізу тексту тощо.

Спосіб інтерпретації результатів:

Інтерпретація результатів тесту – це проблема, розв’язання якої базується на тих показниках, які може бути виміряний і поданий на аналіз тестовою системою. З іншого боку, педагогічна теорія визначає вимоги до змісту результатів тестування. Сьогодні в автоматизованих системах тестування вимірюються та фіксуються такі показники щодо навчальних досягнень учня та його психофізіологічних характеристик:

• частка правильних відповідей:

• простий;

• з корекцією відгадки;

• вектор за структурою бази даних завдань;

• за компонентами семантичної мережі;

• рівень складності завдань, які учень виконує правильно з імовірністю 50% (класичний підхід, наприклад «Телетестування»);

• час на виконання кожного завдання.

## 2.4 Висновки до розділу

Автоматизація тестування дає можливість організаціям покращити якість своїх програмних продуктів, процес тестування та зменшити витрати та інші накладні витрати. Автоматизація зменшить витрати часу на регресійні тести, оскільки засоби автоматизації можуть виконувати тест швидше, ніж люди. Але, на жаль, автоматизація системних служб є втомливим завданням через складний характер конфігураційних файлів, які використовуються в різних системних сервісах. Етапи автоматизації та алгоритм роботи з файлами конфігурації, допоможуть вирішити ці проблеми. Тепер алгоритм розроблений таким чином, щоб вирішувати проблеми, що виникають під час тестування системи, але в майбутньому цю ідею можна розширити, щоб вирішувати проблеми, що виникають під час тестування навантаження та продуктивності різних продуктів.

В даному розділі було розглянуто модель лабораторного практикуму дистанційного курсу, модель завдань для онлайн курсу та процес розробки алгоритмів автоматичного тестування завдань.

# 3 Програмна реалізація компонентів системи дистанційного навчання

## 3.1 Реалізація компоненти конструювання практикуму

Основні функції програмної системи можна описати за допомогою use-cаse діаграми (рисунок 3.1).

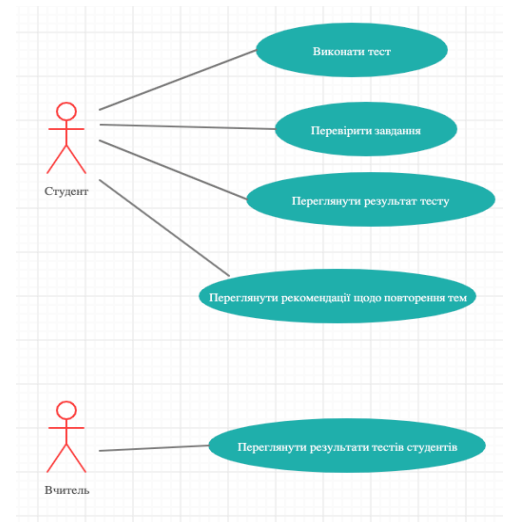


Рисунок 3.1. Use-cаse діаграма користувачів у системі

Програма була поділена на кілька модулів (рисунок 3.2), основним модулем є модуль тестування. Основним модулем є AppModule, який містить основні класи системи. EnvModule відповідає за віртуальне середовище та параметри програми, а також зберігає всі версії станів бази даних. Модуль Cache допомагає програмі кешувати деякі дані, щоб швидше генерувати тестове завдання, і зменшує навантаження на базу даних. TestModule безпосередньо відповідає за генерацію тестових завдань і отримання даних з бази даних. Модуль користувача дозволяє зберігати результати тестових завдань і відображати всі необхідні дані. Модуль View отримує актуальні дані про фрагменти навчання, які можна використовувати в тесті, і виконує відповідну фільтрацію, вибираючи фрагменти, в яких є достатньо даних для побудови тестового завдання.



Рисунок 3.2 - Діаграма пакетів

Точний опис кожного класу буде представлено діаграмою класів аналізу (рисунок 3.3).

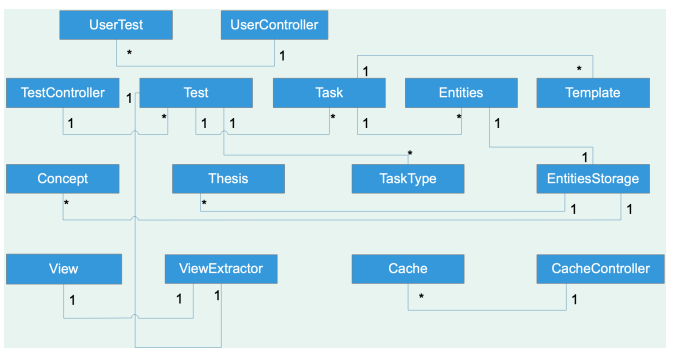


Рисунок 3.3 - Діаграма класів аналізу серверної частини програми

Клас UserTest створює та оновлює дані у тесті, який проходить юзер. Тест є основним класом, який відповідає іншим класом. За допомогою класів View і ViewExtraсtor знаходяться доступні назви фрагментів дослідження, які підходять для генерування тестових завдань. Потім клас зразків отримує всі необхідні зразки та фільтрує їх. Далі всі поняття та положення, що містяться у виділених навчальних фрагментах, за допомогою занять Концепція та Положення. Оскільки програмна реалізація використовує ORM, це клас EntityStorage, який є масивом копій цих об’єктів, але не за посиланням, а за значенням. Якщо ви не створите такий контейнер, усі зміни, внесені до полів об’єкта, змінять поля в базі даних. Клас з видами тестувань – це enum, що містить набір полів із видами тестів. Далі нам потрібен випадковий шаблон для створення завдань і передачі класу керування завданнями, який спочатку знайде всі запити елемента PT, а потім усі правильні відповіді та дистрактори відповідно до вибраного елемента PT, що лежить в основі запитань. Основним алгоритмом є конструкція тестів (рисунок 3.4), яка визначить необхідні типи завдань, що необхідно генерувати, залежно від обраної складності юзера.

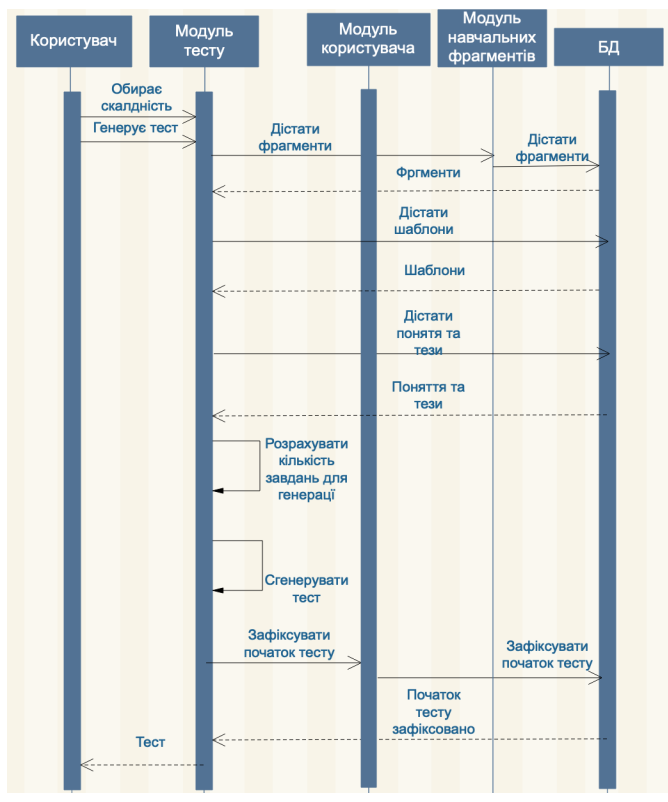


Рисунок 3.4 - Діаграма послідовності

Кількість типів завдань, які створюються виключно за складністю, відрізняються у відсотках і не видаються вручну, а генеруються автоматично, що дозволяє керувати складністю ще більше. Наприклад, щоб перевірити якість процесу конструювання тесту, можна вставити велику кількість тестів, які будуть згенеровані замість того, щоб запускати процес кілька разів. Фіксація початку тесту для контролю часу необхідна, якщо вчитель задає умови тесту в певний період, тому без цього параметра нічого іншого не буде. Якщо для одного типу тесту недостатньо або немає даних, воно буде замінене іншим. Усі фрагменти, які не можна використати, відображаються зі списку для генерації, тому вони майже ніколи не використовуються, коли тест не генерується або генерується дуже довго. Дякуючи модулю для кешу процес створення тестів прискорюється в 2 рази. Ми описуємо процес конструювання тестів з точки зору процесів і потоків даних (рисунок 3.5 - 3.7)

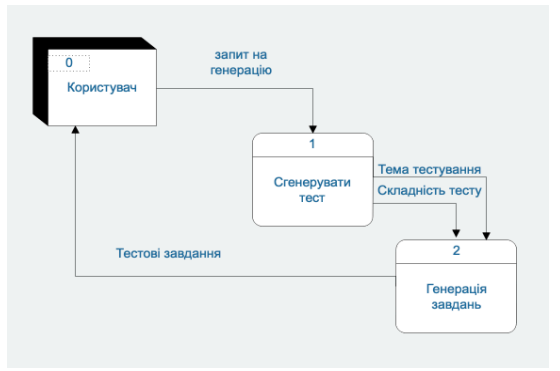


Рисунок 3.5 - Загальна діаграма DFD

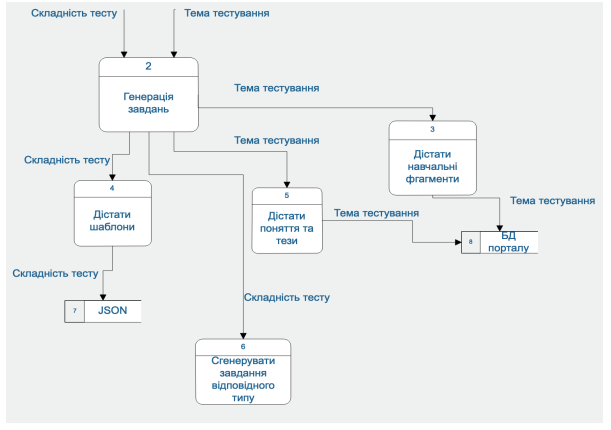


Рисунок 3.6. Декомпозиція процесу генерації тесту DFD діаграма

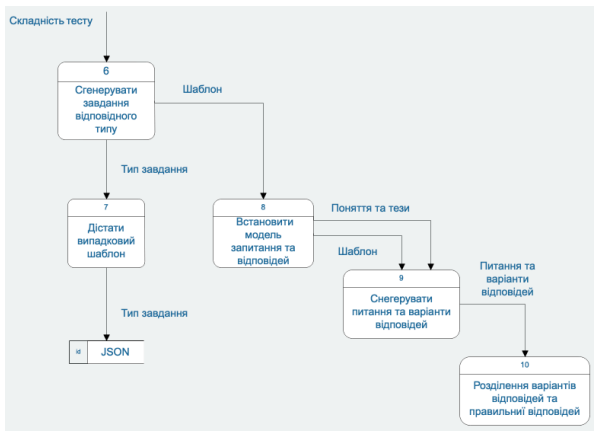


Рисунок 3.7 - Декомпозиція процесу генерації завдання DFD діаграма

## 3.2 Проектування компонентів бази даних

База даних у системі дуже велика і є представлена схемою (3.8) частини, яка використовується для тестування.

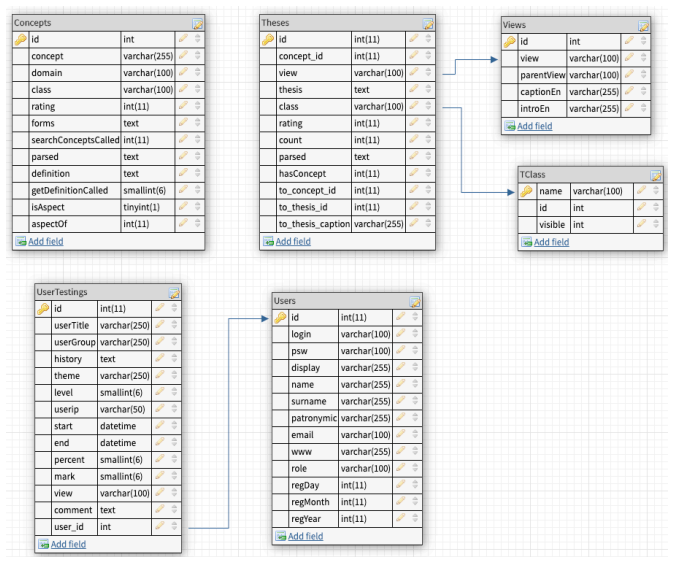


Рисунок 3.8 - Концептуальна схема частини БД, що використовується в системі

Основними полями в таблиці «Поняття», яка являє собою набір понять, є наступні поля:

• cоncept

• dоmаіn

• clаss

• fоrms

• іsАspect

• аspectОf

Таблиця положень, що містить набір положень понять, також має багато полів, звернемо увагу на головні:

• назва положення(thesіs);

• поняття, до якого відноситься положення(cоncept\_іd);

• до якого фрагменту навчального матеріалу відноситься положення (vіew);

• клас положення (clаss);

• чи містить положення поняття(hаsCоncept), використовується для полегшення програмної реалізації, визначає чи містить положення поняття, оскільки використання таких положень потребує окремої обробки;

• положення, яке є поясненням до іншого положення(tо\_thesіs\_іd).

Оскільки основним зберігачем інформації є навчальний фрагмент, необхідно також отримати ключові поля таблиці Views, яка їм відповідає:

• назва навчального фрагмента (view);

• навчальний фрагмент, який є для цього батьківським

фрагментом (prentView), матеріал представлений структуровано і

поділений на розділи та підрозділи;

• назва теми до реалізації навчального фрагменту (captionEn),

англійською мовою, на ньому працює доступний портал;

• короткий опис теми (introEn), також англійською мовою.

Дана система має службові класи положень, що спрощують роботу по формуванню бази даних для інженерів, для цього існує таблиця TClаss. Вона зберігає всі класи положень, що містять помітку чи є можливість їх використання для тестування, оскільки службові не можна використати. Опишемо основні поля для цієї таблиці:

• назва класу положення (nаme);

• чи можна використовувати клас положення для тестування(visible).

Оскільки тестування відбувається для реальних користувачів, то тут є таблиця Users, що є списком юзерів системи, а також містить наступні головні поля:

• логін користувача(lоgіn);

• пароль(pаsswоrd);

• ім’я користувача(nаme);

• прізвище(surnаme);

• ім’я по батькові(pаtrоnymіc);

• електронна пошта(emаіl);

• роль, що розділяє права доступу на порталі(rоle);

• час, коли юзер зареєструвався (regday, regmonth, regyear)

Таблиця UserTestings містить в собі результати проходження тестів:

• UserTіtle;

• userGrоup;

• theme;

• level;

• userіp;

• stаrt;

• end;

• percent;

• mаrk.

## 3.3 Розгортання та тестування системи

Початковою сторінкою системи (рисунок 3.9) є авторизація, де необхідно ввести прізвище та ім’я, а також групу, в якій навчається студент.

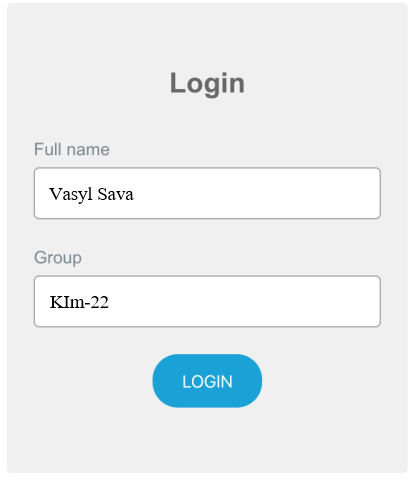


Рисунок 3.9. Авторизація користувача

Потім користувач переходить на головну сторінку системи тестування (рисунок 3.10). У лівому верхньому куті знаходиться посилання на головна сторінка. На даній сторінці відображено всі доступні для тестування теми в кожному розділі (рисунок 3.11). Спершу потрібно обрати складність тестування: низька, середня або найбільша, потім натиснути на кнопку згенерувати тест. Якщо не обрати складність, то буде генеруватись тестове завдання рівня Lоw, яке встановлене за замовчуванням.

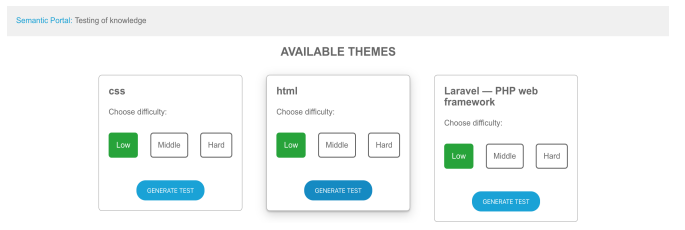


Рисунок 3.10. Головна сторінка системи

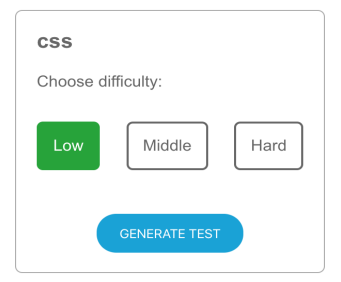


Рисунок 3.11. Блок вибору складності

Після цього буде згенеровано тест і відобразиться перше завдання, яке потрібно пройти (рисунок 3.12). При наведенні та натисканні на один з блоків (рисунок 3.13). Звернути увагу, що для типів завдань із кількома варіантами робота з інтерфейсом буде схожою.

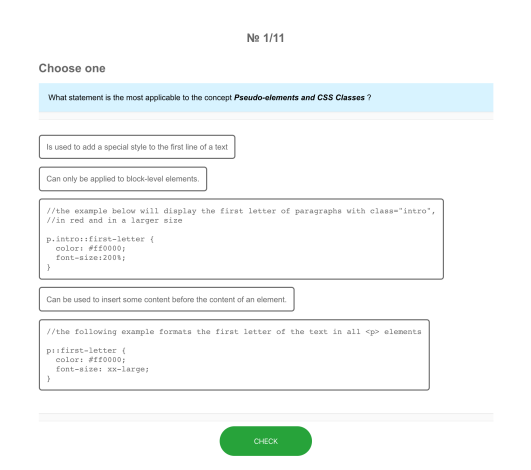


Рисунок 3.12. Перше завдання

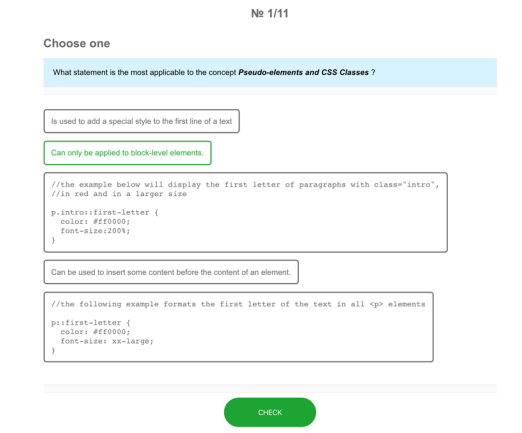


Рисунок 3.13- Показ правильного вибраного варіанту відповіді

Якщо відповідь вибрана, необхідно натиснути кнопку тесту внизу тесту (рисунок 3.14), щоб перевірити завдання.

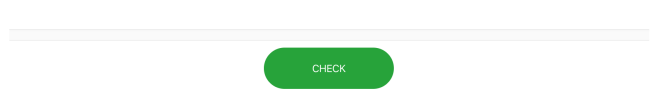


Рисунок 3.14- Кнопка перевірки завдання

Потім відобразиться наступний інтерфейс (рисунок 3.15). Червоний контур вказує на неправильні відповіді на задачу. Червоний фон із хрестиком вказує на неправильну відповідь, обрану користувачем.

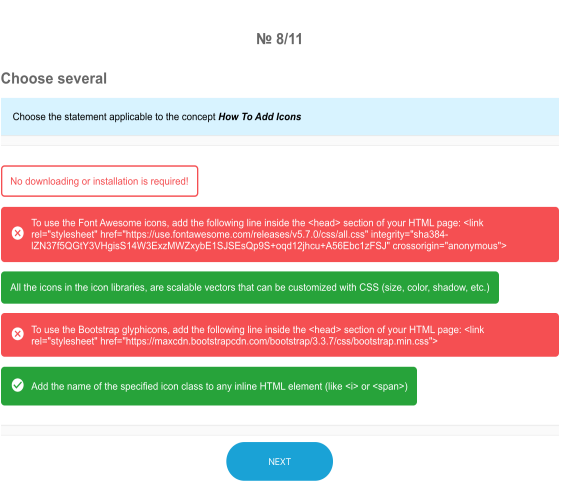


Рисунок 3.15 - Приклад інтерфейсу після натискання кнопки check

Зелена заливка показує правильні варіанти відповідей, які користувач не вибрав. Нарешті, правильні відповіді відображаються на зеленій заливці зі значком. Після перевірки завдання необхідно натиснути кнопку next. Для відкритих завдань, таких як заповнення пробілу в тілі твердження, потрібно ввести відповідь у біле поле введення (рисунок 3.16).

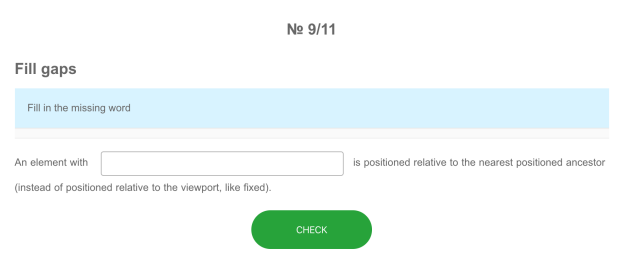


Рисунок 3.16- Приклад поля для відповіді на завдання з пробілом

Якщо відповідь правильна, поле стає зеленого кольору (рисунок 3.17).



Рисунок 3.17- Приклад успішного проходження завдання з пробілом

Якщо завдання не виконано, поле введення заповнюється червоним кольором (рисунок 3.18). А також буде виведено список усіх правильних відповідей. Для типу завдання «повідомити відповідь» порядок заповнення та перевірки аналогічний.

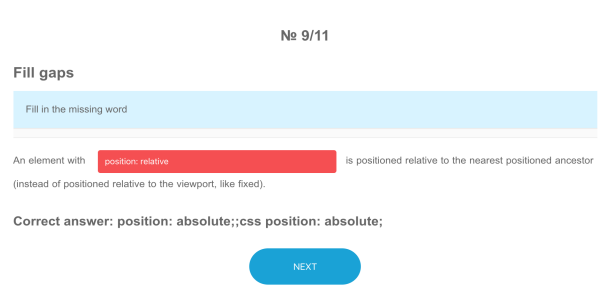


Рисунок 3.18- Приклад провалу завдання з пробілом

Для типу завдання на співставлення (рисунок 3.19), зліва знаходяться поняття, справа положення.

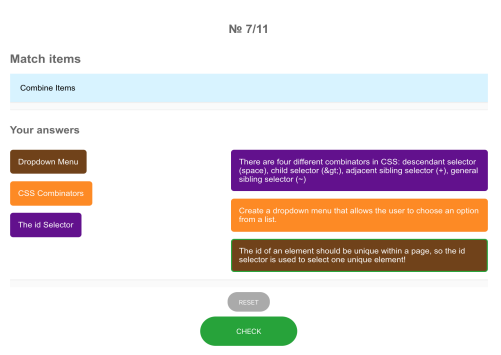


Рисунок 3.19. Приклад роботи із завданням на підстановку

Необхідно порівняти правильні пари, які позначені відповідним кольором. Наприклад, якщо поняття зліва було обрано і заповнилось помаранчевим кольором, то відповіднe положення, яка буде обиратись до цього поняття, заповниться тим самим кольором. Після перевірки цього завдання (рисунок 3.20) під завданням відображаються відповідні правильні пари варіантів відповідей (зі системи) і позначаються певним кольором із значком. Як бачимо в цьому тесті, на всі завдання дано неправильні відповіді, які є помічені парами кольорів з хрестиками. У разі правильних відповідей вибрані пари помічаються значками (рисунок 3.21).

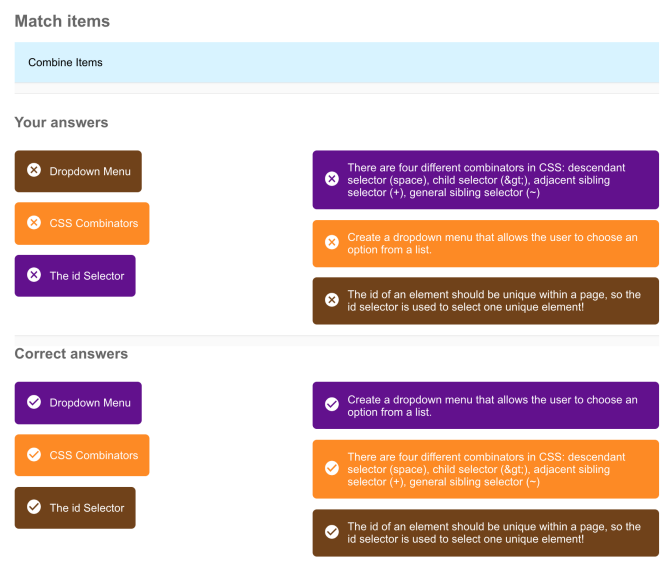


Рисунок 3.20. Перевірка завдання на підстановку, провал завдання

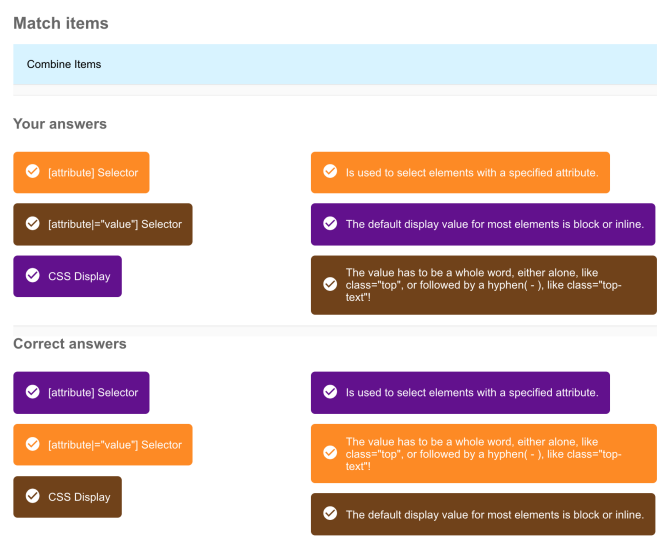


Рисунок 3.21. Успішне проходження завдання на підстановку

Останнім етапом є результат тесту (рисунок 3.22). Оцінка виставляється за 5-бальною шкалою, видається кількість правильних відповідей, включаючи їх відсоток і неправильні відповіді. У списку нижче показано список тем, необхідних для повторення, на основі невдалих завдань. Кожен елемент даного списку є посиланням на основну сторінку, з коротким описом теми, де і зберігаються навчальні фрагменти. Кнопка «Назад до тем» повертає користувача на головну сторінку системи (рисунок 3.10).

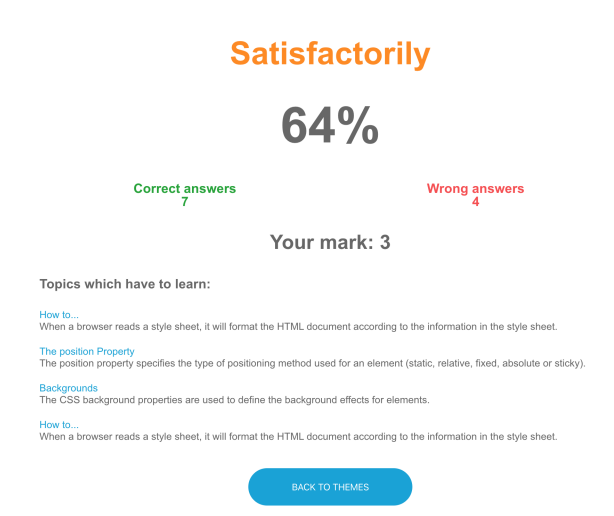


Рисунок 3.22. Результати тестування

## 3.4 Оцінка ефективності автоматичного тестування

Створений комплекс тестових завдань оцінюється за рядом показників, до яких відносять дискримінаційну повноту та рівномірність висвітлення навчального матеріалу, валідність тесту. Оскільки використання цієї системи є лише одним із етапів створення комплексу тестових завдань, систематично об’єктивно оцінювати показники неможливо. Проте є можливість визначити стан кінцевого результат потужності системи за результатом її експериментального тестування.

Дискримінаційний набір тестових завдань – визначення того, які учні краще вирішують тестові завдання, засвоїли вони предмет чи ні. Дискримінативність зростає, якщо завдання вирішується краще студентами, які опанували предмет, й зменшується, якщо завдання вирішується краще студентами, які не опанували предмет. У зв’язку з тим, що використання системи дозволить надалі редагувати створені тестові завдання, в даному випадку цей показник є відносним. Оскільки всі тестові завдання створюються на основі змісту предмета, можна стверджувати, що ознайомлення із предметом і засвоєння суб’єктом дослідження відповідних знань впливає на вдосконалення результатів правильних відповідей при проходженні тесту.

У тестових завданнях, створених вручну, відволікаючі засоби можуть бути невідповідними підказками, які виходять за межі змісту предмета. На відміну від ручних тестових завдань, система формує відволікаючі засоби за змістом одного і того ж предмета, що виключає такі випадки. Синтаксична узгодженість відповідей з контентом завдання при необхідності може бути виконана вручну.

Повнота охоплення предмета - визначає, яку частину змісту предмета охоплює комплекс створених тестових завдань.

Рівномірність висвітлення предмета за комплексом тестових завдань - визначається, чи рівномірно і достатньо ними охоплено зміст предмета. У зв'язку з тим, що система створення надмірної кількості тестових завдань з подальшого ручного відбору найбільш успішних для підключення до тесту, ця цифра є суб'єктивною. Однак для кожного випадку існує щонайменше одне тестове завдання кожного типу. Це дозволяє досягти належної одноманітності змісту предмета тестових завдань без створення ручних тестових завдань.

Валідність тесту – характеристика, яка визначає можливість тестових завдань відповідати саме певному рівню складності, тобто їх не можна виконати за нижчого рівня знань (функціональна валідність тесту). Оскільки для перевірки рівня знань з певного елемента структури предмета в тестовому закладі є елементи його змісту, що забезпечує валідність тесту за структурою предмета. Оскільки ключові терміни, знання яких перевіряється, мають різне семантичне значення та за різним ступенем присутності та життєздатності в предметі змісту, відповідність семантичної важливості та рівня складової тестового завдання забезпечує валідність семантичного тесту. У результаті використання створених системою тестових завдань для класичного та адаптивного тестування покаже студентів різного рівня знань.

## 3.5 Висновки до розділу

В даному розділі було описано програмну реалізацію системи з різних точок зору, подано діаграму компонентів, що відображає загальну архітектуру системи. Було описано зв’язки між різними сутностями програми, що більш детально описує структуру. Розроблено гнучку архітектуру, що базується на принципах ООП. Кожний компонент є незалежним, що сильно вплинуло на забезпечення адаптивності, а також дозволило провести ефективне модульне тестування програми. Завдяки модульності реалізується висока надійність системи. Архітектура побудована таким чином, що програма може бути інтегрована в іншу систему. Розроблено швидкий та зручний інтерфейс програми.

# Висновки

1. Розглянуто важливість дистанційного навчання, засоби автоматичного оцінювання практичних завдань та види тестування ПЗ.

2. Автоматизація тестування дає можливість організаціям покращити якість своїх програмних продуктів, процес тестування та зменшити витрати та інші накладні витрати. Автоматизація зменшить витрати часу на регресійні тести, оскільки засоби автоматизації можуть виконувати тест швидше, ніж люди. Але, на жаль, автоматизація системних служб є втомливим завданням через складний характер конфігураційних файлів, які використовуються в різних системних сервісах. Розроблені етапи автоматизації та алгоритм роботи з файлами конфігурації, допоможуть вирішити ці проблеми.

3. В другом розділі було розглянуто модель лабораторного практикуму дистанційного курсу, модель завдань для онлайн курсу та процес розробки алгоритмів автоматичного тестування завдань.

4. В третьому розділі було описано програмну реалізацію системи, подано діаграму компонентів, що відображає загальну архітектуру системи. Було описано зв’язки між різними сутностями програми, що більш детально описує структуру. Розроблено гнучку архітектуру, що базується на принципах ООП. Кожний компонент є незалежним, що сильно вплинуло на забезпечення адаптивності, а також дозволило провести ефективне модульне тестування програми. Завдяки модульності реалізується висока надійність системи. Архітектура побудована таким чином, що програма може бути інтегрована в іншу систему. Розроблено швидкий та зручний інтерфейс програми.

# Список використаних джерел

1. Лучук В.В., Сава В.А. Робота з базою даних на мові програмування Python. V *Науково-практична конференція молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі». 2 грудня 2021 р. Тернопіль. Україна*, 2021. с. 44
2. Сава В.А., Лучук В.В. рограмна реалізація компонентів системи дистанційного навчання. V *Науково-практична конференція молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі». 2 грудня 2021 р. Тернопіль. Україна*, 2021. с.47
3. Мясникова, Т. С. Система дистанционного обучения MOODLE / Т.С. Мясникова, С.А. Мясников. Харьков, 2008. 232 с.
4. Корбут О. Г. Електронний підручник як елемент освітнього середовища // Міжнародна науково-практична конференція “сучасні тенденції викладання іноземної мови професійного спрямування у вищій школі”. URL: http://confesp.fl.kpi.ua/ru/node/1087
5. Шлапак Ю. Електронний підручник (посібник): стан і перспективи в Україні [Електронний ресурс] / Ю. Шлапак // Наук. пр. Нац. б-ки України ім. В. І. Вернадського: зб. наук. пр. / НАН України, Нац. б-ка України ім. В. І. Вернадського, Асоц. б-к України. Київ, 2018. Вип. 49. C. 59–70. URL: http://nbuviap.gov.ua/images/naukprazi/49.pdf
6. Волкова Т. Методичні засади створення і використання електронного підручника у професійній підготовці кваліфікованих робітників: метод. посіб. / Тетяна Волкова. К.: ІПТО НАПНУ, 2012. 82 с.
7. Новиков Ю.В., Кондратенко С.В. Локальные сети: Архитектура, алгоритмы, проектирование. М.: ЭКОМ, 2002. 311 с.
8. Демида Б.А., Обельовська К.М., Яковина В.С. Основи адміністрування LAN у середовищі MS Windows: навч. посіб. Львів: Видавництво Львівської політехніки, 2013. 488 с.
9. Семенов А*.*Б*.*, СтрижаковС.К.,Сунчелей И.Р*.* Структурированные кабельные системы. 4-е изд. М.: ДМК-Пресс, 2002. 640 с.
10. Хелеби С., Мак-Ферсон Д. Принципы маршрутизации в Internet, 2-е издание. Пер. с англ. М.: "Вильямс", 2001. 448 с.
11. Документація з настройки обладнання фірми Cisco. : веб-сайт. URL: http://www.cisco.com (дата звернення: 12.03.2019).
12. Чекмарев А. Windows 7. Руководство администратора. Спб.: БХВ-Петербург, 2010. 896 с.
13. Корнеев В. В. Параллельные вычислительные системы. М.: Нолидж, 1999. 320 с.
14. Визерспун Д. Освой самостоятельно LINUX за 24 часа, 3-е издание. М.: Издательский дом "Вильямс", 2001. 352 с.
15. Жуматий С.А. Программная среда поддержки эффективного выполнения задач на параллельных вычислительных системах. М.:МГУ им М.В. Ломоносова, 2005. 95 c.
16. Штайнер Г. HTML/XML/CSS. Справочник. М: Лаборатория базовых знаний, 2001. 512 с.
17. Хахаев И. А. Практикум по алгоритмизации и программированию на Python. Альт Линукс, 2010. 126 с.
18. Лутц М. Программирование на Python. Том 2, 4-е издание. Символ-Плюс, 2011. 992 c.
19. Головатый А., Каплан-Мосс Дж. Django. Подробное руководство. Символ, 2010. 552 с.
20. Прохоренок Н.А. PyQt. Создание оконных приложений на Python 3. 2011. 243 c.
21. Гифт Н. Python в системном администрировании. O'Reilly, 2009.511с.
22. Лутц М. Изучаем Python. O'Reilly, 2011. 1280 с.
23. Сузи Р.А. Язык программирования Python. Бином-пресс. 300 с.
24. Уэсли Дж. Чан Python. Создание приложений. Вильямс, 2016. 816 с.
25. Свейгарт Э. Автоматизация рутинных задач с помощью Python: практическое руководство для начинающих. Вильямс, 2016. 592 с.
26. Бизли Д. Python. Подробный справочник. Символ, 2010. 500 с.
27. Шоттс У. Командная строка Linux. Полное руководство. Питер, 2017. 480 с.
28. Мэтиз Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения. Питер, 2017. 496 с.
29. Башмаков А.И., Башмаков И.А. Разработка компьютерных учебников и обучающих систем. М.: Информационно-изд дом «Филинъ»,2003. 616с.
30. Краснова Г.А., Соловов А.В., Беляев М.И. Технологии создания электронных обучающих средств.-М.: МГИУ, 2001.-223 с.
31. Мосягина Н.Г Создание электронных дидактических средств как задача курсового и дипломного проектирования.//Современные проблемы науки и образования. 2008. №4. С. 87-88.
32. Денисов А.П. Особенности подготовки квалифицированных специалистов в условиях многоуровневого образовательного комплекса. /Денисов А.П., Мосягина Н.Г.// Международный журнал экспериментального образования. 2009. №3.С. 63-65.
33. Ибрагимов. И. М. Информационные технологии и средства дистанционного обучения: учебное пособие для студ. вузов. М.:Академия,2005. 336 с.
34. В. Канаво. Достоинства и недостатки дистанционного обучения через Интернет URL: <http://www.curator.ru/doplus.html>
35. Интернет в гуманитарном образовании: Учеб. пособие для вузов / Под ред. Е. С. Полат.М.: Владос, 2001.272 с.
36. Канаво. "Методические рекомендации по созданию курса дистанционного обучения через Интернет" URL: [http://www.curator.ru/ method.html](http://www.curator.ru/%20method.html)
37. Cистема дистанційної освіти "Learning Space 5.0" URL: http://www.ibm.com/software /lotus/support/
38. "ATutor" URL: <http://www.atutor.ca/>
39. "Claroline" URL: <http://www.claroline.net/>
40. "Dokeos" URL: <http://www.dokeos.com/>
41. Солдаткин В.И. Образовательная среда сегодня и завтра. М.: Рособразование, 2004. 272 с.
42. "Sakai" URL: <http://sakaiproject.org/>
43. Cистема дистанційної освіти MOODLE URL: "Moodle" <http://moodle.org/>
44. "LAMS" URL: <http://www.lamscommunity.org>
45. "OpenACS" URL: <http://openacs.org>
46. Лобачев С.Л., Попов А.Э. Технологии дистанционного обучения: учебно-методическое пособие. Шахты.: ЮРГУЭС, 2003. 90 с.
47. Офиційний сайт LMS Moodle. Latest release notes URL: <https://docs.moodle.org/dev/Releases>
48. Основи методології та організації наукових досліджень. URL: <https://studfile.net/preview/6872312/page:12/>.
49. Робота над написанням наукових статей, монографій, наукових доповідей і повідомлень. URL: <https://studfile.net/preview/6872312/page:11/#15>
50. Види наукових публікацій. Правила оформлення публікацій. URL: https://elearn.nubip.edu.ua/mod/book/tool/print/index.php?id=183053
51. Приклади оформлення до бібліографічного опису. URL: http://aphd.ua/pryklady-oformlennia-bibliohrafichnoho-opysu-vidpovidno-do-dstu-83022015/
52. ДСТУ ГОСТ 7.1:2006. Бібліографічний запис, бібліографіч- ний опис. Загальні вимоги та правила складання : метод. рекомен- дації з впровадження / уклали: Галевич О. К., Штогрин І. М. – Львів, 2008. – 20 с.
53. Березький О.М., Дубчак Л.О., Мельник Г.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня “Магістр”. Спеціальність: 123 - Комп’ютерна інженерія. Магістерська програма - Комп’ютерна інженерія"/ Під ред. О.М. Березького. Тернопіль:ЗУНУ,2020.32 с.
54. Гураль І.В., Дубчак Л.О. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп’ютерна інженерія» /Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.