Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи

магістра
(освітній ступінь (освітньо-кваліфікаційний рівень))

на тему: _____

Виконав: студент (ка) VI курсу, групи СПм-62

спеціальності (напряму підготовки) 121

Інженерія програмного забезпечення
(шифр і назва спеціальності (напряму підготовки))

_____ Петрук О.В.
(підпис) (прізвище та ініціали)

Керівник _____ Пастух О.А.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Бойко І.В.
(підпис) (прізвище та ініціали)

Рецензент _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

м. Тернопіль – 2019

АННОТАЦІЯ

Дипломна робота на тему «Розробка програмного забезпечення для автоматизації тестування Інтернет ресурсу на мові програмування C#» Петрука Олександра Володимировича. – Тернопільський національний технічний університет імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, Кафедра програмної інженерії, група СПм–62 // Тернопіль, 2019.

С. – 110, рис. – 15, табл. – 22, слайдів. – 10, додат. – 7, бібліогр. – 46.

Метою даної дипломної роботи є вивчення автоматизованого процесу тестування як явище, аналіз існуючих підходів які знаходяться у вжитку та використовуються фахівцями у галузі автоматизації тестування, вивчення методологій оцінки якості програмного забезпечення, а також розроблення та введення автоматизованого процесу тестування у існуючі процеси розробки програмного забезпечення.

Програмні застосунки та патерни які були використані при виконанні розробки програми: мова програмування C# та її бібліотеки .NET фреймворку, середовище розробки Visual Studio, інструмент Selenium WebDriver, фреймворк NUnit, патерн проектування PageObject, фреймворк Atata, програмне забезпечення для автоматизації розгортання і управління додатками Docker.

Результатом виконаної роботи є програмне забезпечення, яке дозволить з легкістю запускати на виконання тестові сценарії, повертати результати тестових запусків для подальшого їх вивчення та аналізу, а також спроектовані тестові випадки, та реалізовані тестові скрипти.

Ключові слова: РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, C#, SELENIUM WEBDRIVER, NUNIT, PAGE OBJECT, ATATA, DOCKER, ПАТЕРНИ ПРОЕКТУВАННЯ .

ABSTRACT

Graduate work on the topic «Development software for automation testing Internet resource in C# programming language» by student Petruk Oleksandr Volodymyrovich. – Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Software engineering department, group SPm-62 // Ternopil, 2019.

Pages – 110, pictures – 15, tables – 22, slides – 10, adds – 7, bibl. ref. – 46.

The purpose of the work is to study the automated testing process as a phenomenon, to analyze existing approaches that are in use and used by experts in the field of automation of testing, to study the methodologies of software quality assessment, and to develop and introduce an automated testing process in existing software development processes.

Software applications and patterns used in the development of the program: C # programming language, .NET framework, Visual Studio development tool, Selenium WebDriver tool, NUnit framework, PageObject design pattern, Atata framework, Docker automation software and application management.

The result of the work done is software that will allow you to easily run test scenarios, return test run results for further study and analysis, as well as test cases designed, and test scripts implemented.

Keywords: DEVELOPMENT OF SOFTWARE, QUALITY OF SOFTWARE, C #, SELENIUM WEBDRIVER, NUNIT, PAGE OBJECT, ATATA, DOCKER, PATTERN DESIGN.

ЗМІСТ

ВСТУП

Процес розробки програмного забезпечення – сукупність ряду послідовних дій, спрямованих на розробку програмного забезпечення. Існує кілька моделей такого процесу, кожна з яких описує свій підхід, у вигляді завдань і/або діяльності, які мають місце в ході процесу. Вибір тієї або іншої моделі здійснюється відповідно до обраної методології розробки програмного забезпечення.

Якість програмного забезпечення – це будь-який систематичний процес визначення, чи відповідає продукт чи послуга визначеним вимогам.

Якість програмного забезпечення встановлює та підтримує встановлені вимоги щодо розробки або виготовлення надійних виробів. Система забезпечення якості має на меті підвищити довіру клієнтів та надійність компанії, а також покращити робочі процеси та ефективність, а також дає змогу компанії краще конкурувати з іншими. Діяльність тестування, яка тісно вплетена в процес розробки програмного забезпечення, й відбувається на кожному його рівні, сприяє покращенню якості та надійності програмних систем, в результаті чого є підвищення попиту на створене програмне забезпечення на ринку та задоволення потреб замовників та користувачів.

Автоматизоване тестування є необхідною частиною усіх сучасних процесів розробки програмного забезпечення. Аналіз та покращення процесу розробки програмного забезпечення за рахунок впровадження автоматизованого тестування дозволяє проектувати та створювати більш якісні та комплексні програмні системи із меншими фінансовими та затратами людських ресурсів затратами (мова йде про кількість робочих годин).

Метою даної дипломної роботи є вивчення автоматизованого процесу тестування як явище, аналіз існуючих підходів які знаходяться у вжитку та використовуються фахівцями у галузі автоматизації тестування, вивчення методологій оцінки якості програмного забезпечення, а також розроблення та введення автоматизованого процесу тестування у існуючі процеси розробки програмного забезпечення.

Об’єкт дослідження: процес розробки програмних систем. Предмет дослідження: якість сучасного програмного забезпечення та її можливість до покращення виконавши впровадження автоматизованого тестування у процеси розробки програмного забезпечення. У цій дипломній роботі використовуються найсучасніші методи розробки та написання автоматизованого тестування Інтернет ресурсів застосовуючи такі інструменти, як мова програмування C#, фреймворк для створення різноманітних типів тестів NUnit, застосунок для автоматизації дій у браузері Selenium WebDriver, а також тестовий фреймворк Atata.

Написання програмного забезпечення та тестових сценаріїв відбуватиметься на основі системи управління онлайн навчанням з відкритим кодом ATutor та мови програмування C#

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AECT – Association for Educational Communications and Technology (Асоціація освітніх комунікацій та технологій)

LMS – Learning Management Systems (Системи Управління Навчанням)

EMIS – Education Management Information Systems (Інформаційні Системи Управління Освітою)

PDF – Portable Document Format(Портативний формат документів)

VLE – Virtual Learning Environment (Віртуальне навчальне середовище)

W3C – World Wide Web Consortium (Консорціум Всесвітньої павутини)

WCAG – Web Content Accessibility Guidelines (Керівні принципи доступності веб-вмісту)

SDLC – Software Development Life Cycle (Життєвий циклу розробки програмного забезпечення)

BRS – Business Requirement Specification (Специфікація вимог до бізнесу)

SRS – System Requirement Specifications (Специфікація системних вимог)

OS – Operation System(Операційна система)

API – Application Programming Interface (Прикладний програмний інтерфейс)

LXC – Linux Containers (Контейнери операційної системи Linux)

DB – Data Base (База даних)

JSON – JavaScript Object Notation (запис об'єктів JavaScript)

UI – User Interface (Інтерфейс користувача)

1. АНАЛІЗ ПРОГРАМНОЇ СИСТЕМИ ТА МОЖЛИВИХ СПОСОБІВ ТЕСТУВАННЯ

1.1 Аналіз вимог для тестування електронної системи навчання ATutor

Розробка програмного забезпечення - це ітераційний логічний процес, спрямований на створення програмного забезпечення, програмного забезпечення, кодоване або запрограмоване на комп'ютері, для вирішення унікальної ділової чи особистої мети, цілі чи процесу. Розробка програмного забезпечення, як правило, є плановою ініціативою, яка складається з різних етапів або етапів, які призводять до створення операційного програмного забезпечення[1]. Як наслідок розробка програмного забезпечення може включати в себе дослідження, розробку, прототипування, модифікацію, повторне використання, реінжиніринг, технічне обслуговування чи будь-які інші заходи, що результують у програмні продукти [2].

Розробка програмного забезпечення в першу чергу досягається за допомогою комп'ютерного програмування, яке здійснюється програмістом програмного забезпечення та включає такі процеси, як початкові дослідження, проектування потоку даних, проектування технологічного процесу, графіки потоків, технічна документація, тестування програмного забезпечення, налагодження та інші методи архітектури програмного забезпечення.

Необхідність покращення контролювання якості процесу розробки програмного забезпечення вилилась у створення дисципліни програмного забезпечення, головною метою якої є застосування підходу, прикладеного в інженерній парадигмі до процесу розробки програмного забезпечення.

Основними предметними областями які можна виділити у цій магістерській роботі, являються:

- Освіта та сучасні освітні технології – адже, відповідно до теми магістерської роботи та технічного завдання, задачею є дослідити та виконати автоматизоване тестування системи електронного управління навчанням ATutor;

- Тестування програмного забезпечення як необхідна та критична частина життєвого циклу розробки програмного забезпечення.

1.1.1 Аналіз предметної області

1.1.1.1 Роль освіти у сучасному світі

Освіта - найцінніший ключ до успіху. Сила освіти полягає не лише в академічних знаннях, а й у знаходженні свого місця у світі та набутті життєвого досвіду. Він вживає всіх необхідних кроків для полегшення нашого життя та надання нам навичок, які оцінять наші сім'ї та роботодавці. Цінна освіта готує людей до життя та праці та створює міст до забезпеченого майбутнього. Справжня освіта дає навички прийняття рішень та вирішення проблем та чіткий шлях кар’єри. Люди, які мають гідну освіту, часто займають високі позиції у своїй кар’єрі та ведуть задовільне життя. [3].

Освіта може відбуватися у формальній обстановці, а також у неформальній, також будь-який досвід та будь-які знання, які мають формувальний вплив на те, як ми думаємо, аналізуємо, розсуджуємо, відчуваємо, чи діємо, може вважатися освітнім.

Формальна освіта зазвичай відбувається у середовищі яке строго структуризовано. Зазвичай, здобуття повноцінної освіти стається у закладах освітнього типу з групами із визначеної кількості студентів та вчаться у викладача цього предмету, який є сертифікованим спеціалістом у цій області. Основна частина систем університетів мають за ціль певну сукупність цінностей та ідей, які виконують керування більшістю освітніх виборів в системі[4][5].

1.1.1.2 Сучасні освітні технології

Освітні технології - це концепція перетворення традиційного викладання та навчання книг у цифрову форму.

Освітні технології можуть відкрити двері для включення звичайних методів аудиторії з передовими курсами цифрових курсів. Використовуючи цю широку систему, будь-хто може охарактеризувати успішні прийоми та заходи, що підходять до такої своєрідної обстановки, і, за проміжок часу, адекватно дослідити цифрові перетворення в школах[6].

Розрізняють кілька аспектів опису інтелектуального та технічного розвитку освітніх технологій:

- Освітні технології як теорія та практика освітніх підходів до навчання;
- Освітні технології для систем управління навчанням (LMS), такі як інструменти для управління студентами та навчальними процесами, а також інформаційні системи управління освітою (EMIS);
- Освітні технології як технологічні інструменти та засоби для масового розповсюдження інформації, наприклад, онлайн курси, що допомагають в передачі знань ширококій масі людей, а також у розвитку цих знань та обміні;
- Освітні технології як навчальний предмет. Такі курси можна назвати “Комп'ютерні дослідження” або “Інформаційно-комунікаційні технології (ІКТ)”.

Асоціація з освітніх комунікацій та технологій визначила освітні технології як “вивчення та етичну практику полегшення навчання та підвищення продуктивності шляхом створення, використання та управління відповідними технологічними процесами та ресурсами” [7]. Вона позначає навчальну технологію як “теорію та практику проектування, розробки, використання, управління та оцінки процесів та ресурсів для навчання” [8][9].

Таким чином, освітня технологія стосується всіх можливих достовірних та надійних прикладних наук про освіту, а також процесів, процедур та засобів, які випливають з наукових досліджень, і в даному контексті можуть посилатися на теоретичні чи евристичні процеси: тут обов'язково маються на увазі фізичні технології. Освітні технології – це процес інтеграції технології в освіту засобами, які сприяють найбільш різноманітному навчальному середовищу та способом для студентів навчитися використовувати технології.

Спільне навчання – це певна група у якій студенти мають змогу взаємодіяти між собою щоб досягти навчальної мети або завершити навчальне завдання. Завдяки останнім досягненням у технології веб-технологій та смартфонів, потужність комунікації виросла до небувалих величин, а змога використовувати різноманітні додатки додала варіативності у зберіганні знань та її обміную. Багато розробників програм та експерти з питань освіти використовують програми для смартфонів і планшетів як середовище для спільного навчання на постійній основі.

Комп'ютери та планшети дозволяють студентам та викладачам отримувати доступ до Інтернет джерел, а також до програм.

Мобільні пристрої можуть використовуватися для інтерактивного зворотнього зв’язку від аудиторії. Мобільне навчання забезпечує підтримку мобільності, яка проявляється у можливості зажди мати доступ до будьб яких Інтернет джерел та застосунків, завантаженні листів та інструкцій з експлуатації тощо [10][11].

Екранна трансляція - це цифровий відеозапис, який фіксує дії, що відбуваються на екрані комп'ютера. Екрани екрану, які часто містять передачу голосу над голосом, корисні для демонстрації використання конкретних операційних систем, програмних програм або функцій веб-сайту. [12]. Для забезпечення запису екрану потрібне якесь програмне забезпечення для відеозапису та мікрофон. Програмне забезпечення, яке може бути настільним клієнтом або веб-сервісом, фіксує та синхронізує відео та аудіофайли та стискає завершене відео у форматі, яке можна спільно використовувати. Після чого студенти мають можливість скачати та почати використовувати відео.

Віртуальне навчальне середовище (VLE), також відоме як платформа для навчання, імітує віртуальну навчальну аудиторію, одночасно поєднуючи декілька програмних технологій. Програмне забезпечення веб-конференцій дає змогу студентам та інструкторам спілкуватися один з одним за допомогою веб-камери, мікрофона та спілкування в режимі онлайн в групі. Учасники можуть підняти руки, відповідати на опитування чи проходити тести [13].

Віртуальне навчальне середовище надає студентам можливість отримати безпосереднє навчання від кваліфікованого викладача в інтерактивному

середовищі. Студенти мають швидкий доступ до комунікації з викладачем у разі виникнення питань.

1.1.1.3 Переваги та недоліки сучасних освітніх технологій

Правильне та ефективне використання сучасних технологій може цілком замінити викладачів та інструкторів, які здатні до подання адаптивного матеріалу, проведення частих тестувань та опитувань, надання швидких відгуків тощо. Використання комп'ютерів, смартфонів або інших технологій надає можливість для ефективного вивчення теорії та застосування її на практиці, в той час як викладач може працювати з іншими аспектами освіти, проводячи оцінку та надаючи конструктивну критику [14]. Завдяки використанню освітніх технологій, освіта може бути індивідуалізована для кожного студента, що дозволяє краще диференціювати та дозволяє студентам працювати над оволодінням знань власними темпами [15].

Сучасні освітні технології поліпшують доступ до освіти, в тому числі надаючи можливість отримати повну освіту за конкретною спеціальністю. Це дає змогу краще інтегрувати студентів, які не працюють повний робочий день, зокрема в умовах безперервної освіти та покращеної взаємодії між студентами та викладачами. Навчальний матеріал може бути використаний для дистанційного навчання та доступний для широкої аудиторії.

Студенти мають доступ до надзвичайно великої кількості інтернет-ресурсів. Використання онлайнових ресурсів допомагає студентам витрачати більше часу на те, чого вони, за певних умов, не можуть навчитися в навчальних закладах.

Онлайн-навчання сприяє незалежним методам навчання. Інтернет-учні повинні бути спрямовані на досягнення своїх академічних цілей і повинні бути мотивованими. В основному вона покладається на наочний спосіб навчання[16].

Існуючих вікових обмежень немає, тобто студенти можуть йти в своєму власному темпі. Студенти, що редагують свою письмову роботу використовуючи текстові редактори, підвищують якість їх написання. Згідно з дослідженнями, студенти краще критикують та редагують письмові роботи, які обмінюються комп'ютерною мережею зі студентами, яких вони знають [17].

Останнім часом більшість роботодавців оцінюють онлайн освіту на рівні із традиційною. Більш ніж 60% менеджерів з управління персоналом, опитаних під час звіту за серпень 2012 р., розповіли про те, що якщо два кандидати з таким же рівнем досвіду подають заявку на роботу, то на результат співбесіди не впливатиме той факт, чи кандидат отримав свій диплом через Інтернет чи використовуючи традиційні методи навчання [18].

Проте, нові технології часто супроводжуються неправдивою рекламою та обіцянками стосовно їхньої можливості змінити освіту на краще. Прикладами можуть слугувати радіомовлення та телебачення, жоден з яких не втримався в повсякденній практиці основної, формальної освіти майже переставши існувати на сьогодення [19].

1.1.2 Опис системи управління навчанням ATutor

ATutor – це веб-система з відкритим кодом для управління електронним навчанням (LMS), що використовується для розробки та надання Інтернет-курсів. ATutor використовується в різних контекстах, включаючи онлайн-курси, постійне підвищення кваліфікації вчителів, розвиток кар'єри та наукові дослідження. Програмна система вважається унікальною за рахунок її функцій доступності (які корисні для людей із вадами зору та слуху) а також за рахунок придатності для навчального використання відповідно до критеріїв оцінки програмного забезпечення, встановлених Американським товариством з навчання та розвитку (ASTD) [20].

ATutor використовується на міжнародному рівні. Його перекладено більш ніж на п'ятнадцять мов з підтримкою більш ніж сорока додаткових мовних модулів, які активно розробляються.

ATutor – це перша система управління навчанням, яка повністю відповідає специфікаціям доступності W3C WCAG 1.0 на рівні AA+, що дозволяє отримати доступ до всього включеного вмісту системи на всіх рівнях привілеїв користувача, включаючи облікові записи адміністратора [20].

Система ATutor являється єдиною повністю доступною системою управління навчанням на ринку, включаючи платне й закрите програмне забезпечення.

ATutor також згадується в численних технічних оглядах та наукових статтях [21][22]. Також були розроблені різноманітні плагіни та за стосунки для розширення існуючого функціоналу системи.

Серед основних функцій доступності можна виділити альтернативами тексту для всіх візуальних елементів, а також можливість користування клавіатурою для доступу до всіх існуючих елементів системи. За допомогою цих функцій, сліпий користувач може слухати весь інтерфейс системи за допомогою програми зчитування з екрана, і він може отримати доступ до системи, не потребуючи миші. Ці функції також дозволяють ATutor адаптуватися до широкого спектру технологій, включаючи мобільні телефони, асистенти персональних даних (КПК) та текстові веб-переглядачі.

ATutor включає в себе інструмент розробки вмісту, який пропонує розробникам вмісту створювати доступні навчальні матеріали. Підказки заохочують авторів до додавання текстових альтернатив, якщо вони, наприклад, забувають включати їх при додаванні зображень. Інструмент розробки вмісту також включає веб-службу, яка оцінює відповідність контенту до різних міжнародних стандартів. Окрім того, сам інструмент відповідає стандартам доступності, що дозволяє сліпим користувачам самотужки створювати матеріали.

ATutor також призначений для адаптації до будь-якого навчального сценарію. Існує чотири основних напрямки, які відображають цей принцип дизайну: теми, привілеї, інструментальні модулі та групи.

Теми дозволяють адміністраторам легко налаштовувати зовнішній вигляд та компонування системи відповідно до їхніх конкретних потреб. Теми використовуються для того, щоб дати ATutor новий вигляд, або забезпечити декілька версій ATutor в одній системі, з якої користувачі зможуть вибирати ту, яка їм подобається найбільше.

Система привілеїв дозволяє інструкторам призначати привілеї для управління інструментами окремим учасникам курсу. Викладачі можуть додавати

помічників або курсантів, які будуть мати обмежені права до зазначених інструментів розробки чи управління курсом.

У системі ATutor можна виділити 4 основні ролі користувачів:

1. Студенти
2. Інструктори та викладачі
3. Адміністратори
4. Розробники

Список вбудованих можливостей для користувачів системи:

- Доступність – ATutor був розроблений із забезпеченням максимальної доступності як для користувачів так і для розробників. Широкий спектр функцій забезпечує користувачам можливість повноцінно брати участь у навчальній, інструкторській та адміністративній діяльностях.
- Соціальна мережа – Всі користувачі ATutor можуть створювати мережу контактів та об'єднувати їх у групи інтересів, налаштовувати мережевий профіль та ділитися фотографіями через курси або через зону соціальних мереж.
- Актуальний статус – Коли студент або інструктор потрапляють на стартову сторінку, їм надається список всіх поточних даних, що забезпечують швидкий доступ до поточної діяльності на своїх курсах.
- Безпека – Паролі облікових записів шифруються. Забуті паролі анульовуються, й не відсилаються на електрону пошту. Таким чином виключається можливість їх перехоплення.
- Мої курси – Викладачі та студенти можуть керувати курсами, які вони навчають та/або в яких навчаються. Коли студент реєструється на курс, він автоматично вноситься в список активних курсів.
- Обмін повідомленнями – Усі користувачі системи ATutor мають папку "Вхідні", за допомогою якої вони можуть надсилати та отримувати приватні повідомлення від інших користувачів. Відправлені повідомлення зберігаються певний період перед видаленням. Також, повідомлення можна експортувати.

- Профіль студента – Студенти можуть додавати особисті відомості про себе, щоб інші могли їх переглянути, і додати зображення профілю, яке також відображається в повідомленнях форуму. Фотогалерея може бути використана для створення альбому профілю, де можна зберігати колекцію фотографій.
- Адаптивна навігація – Учні можуть переміщатися вмістом ATutor за допомогою глобальних, ієрархічних або послідовних навігаційних інструментів. Навігаційні елементи можна приховати для спрощення середовища.
- Робочі групи – Учні можуть співпрацювати з іншими на курсових проектах, спілкуватися як група через форуми, обмінюватися ресурсами за допомогою утиліти зберігання файлів і спільно працювати над створенням проектних документів. Вправи або завдання можуть бути передані керівнику групи або інструктору курсу.
- Зберігання файлів – Всі користувачі системи ATutor мають власну утиліту для зберігання файлів. Області зберігання файлів також можна розподіляти між групами або цілим курсом. Контроль версії можна активувати для відстеження чернеток або змін до документів.
- Зворотній зв'язок – З виконанням будь-яких дій в межах системи, дається відгук про стан операції. Це може бути повідомлення про успішне виконання, попередження для розгляду або виправлення помилок, тощо.
- Параметри налаштувань – Користувачі можуть контролювати функції та візуальне оформлення ATutor. Студенти можуть контролювати налаштування візуального відображення, настройки адаптації вмісту, елементи керування навігацією та налаштування засобів навчання. Майстер налаштування доступний з будь-якої точки ATutor, щоб швидко змінити налаштування параметрів.
- Відстеження вмісту – Учні можуть відслідковувати сторінки із навчальним матеріалом, які вони відвідували.

- Менеджер тестів – Студенти можуть проходити тести, переглядати результати тестування та стежити за їхніми балами. Гості курсу можуть пройти пробні тести. Студенти можуть продовжити складання тесту, який вони розпочали раніше. Інструктори ж можуть створювати тести з одинарним або множинним вибором, відповіддю типу «Так/Ні», додаванням формул, впорядкуванням відповіді у правильній послідовності, встановленням відповідності між терміном та визначенням. Також є можливість створення відкритих питань.
- Словник термінів – Слова та фрази, додані в словник термінів наставником, можна переглянути безпосередньо у навчальних матеріалах, або ж в повному обсязі в алфавітному порядку за допомогою інструмента «Глосарій».
- Пошук курсів – Пошукова система дозволяє учням шукати по змісту курсу, а також шукати курси по їх назві та інших атрибутах.
- Менеджер резервних копій – Весь вміст і структура курсу можуть бути резервовані та збережені на сервері ATutor або завантажені та збережені на вашому локальному комп'ютері для безпечного зберігання. Створіть копію курсу як шаблон для майбутніх курсів або перемістіть курс до нового місця.
- Новини та оголошення – Інструктори можуть публікувати повідомлення на домашній сторінці курсу, щоб направляти студентів протягом курсу. Новини можуть бути використані для щотижневих сповіщень, оголошувань важливих дат або задля публікації важливої інформації. Сторінка оголошень завжди є першою сторінкою, яку відвідує студент, коли заходить в курс. Також можна скористатись RSS-каналом щоб відображати повідомлення про курси на інших веб-сайтах або через комбінатори новин.
- Опитування – Інструктори можуть створити опитування, щоб швидко почути думки студентів.
- Форуми – Інструктори можуть створювати та керувати багатьма форумами. Повідомлення можуть бути відредаговані, видалені, заблоковані після читання та/або відповіді, а також "закріплені" у

верхній частині списку посилань, якщо повідомлення є важливим. Адміністратори можуть створювати форуми, які є спільними для декількох курсів. Підпишіться на форуми або на конкретну тему, щоб повідомлення про активність автоматично надсилалися на електронну пошту. Інструктори можуть встановити граничний термін для редагування повідомлень форуму, тому повідомлення можуть бути виправлені, якщо в оригінальній публікації виникли помилки. Колишні дискусії на форумі можна заархівувати.

- Менеджер модулів – Адміністратори можуть встановлювати модулі, вмикати та вимикати їх, а також встановлювати стандартний набір модулів та блоків меню для нових курсів. Модулі можна імпортувати безпосередньо із центрального репозиторію, та їх можна автоматично видалити.
- Загальна статистика – Адміністратори можуть переглядати статистику входів у систему.
- Менеджер користувачів – Користувачі в системі можуть бути відсортовані, особисту інформацію можна переглядати, а права доступу можуть бути змінені. Є можливість надіслати повідомлення всім користувачам системи Autor, студентам або інструкторам. Шукайте в базі користувачів, використовуючи різноманітні стратегії пошуку, щоб знайти окремих студентів або групу студентів. Облікові записи користувачів можуть керуватися масово, щоб швидко додавати, змінювати або видаляти облікові записи.
- Менеджер мов – Імпортуйте мовні пакети безпосередньо з atutor.ca або інсталюйте їх в систему з завантаженого мовного пакета. Створіть мовний пакет ATutor, експортуючи мову з системи ATutor. Зробіть мовний пакет доступним для інших користувачів, надіславши його до центрального сховища мов. Всі мови доступні в UTF-8, і курси можуть відображатися декількома мовами одночасно [23].

1.1.3 Постановка задачі

До задач, які необхідно дослідити та вирішити в ході дипломної роботи можна віднести наступні:

1. Аналіз системи управління навчанням ATutor;
2. Виявлення можливих недоліків системи;
3. Аналіз поточного процесу розробки системи;
4. Написання тестових сценаріїв;
5. Інтеграція тестових сценаріїв, в ході якої відбуватиметься покращення процесу розробки за рахунок вдосконалення процесів тестування;
6. Дослідження можливих шляхів автоматизованого тестування тестових сценаріїв;
7. Дослідження необхідного для автоматизації та інтеграції інструментарію;
8. Впровадження автоматизованого тестування в процес розробки.

1.1.4 Опис ключових варіантів використання

Беручи до уваги перелік функцій та можливостей, доступних в системі ATutor, можна виділити наступних головних акторів системи:

1. Студент – проходить навчальний курс;
2. Інструктор – проводить курс та оцінює Студентів;
3. Адміністратор – адмініструє та підтримує систему;
4. Розробник – розробляє та додає модулі в систему.

Проілюструємо ключові варіанти використання для Інструктора на рисунку 1.1, та опишемо їх у таблиці 1.1.

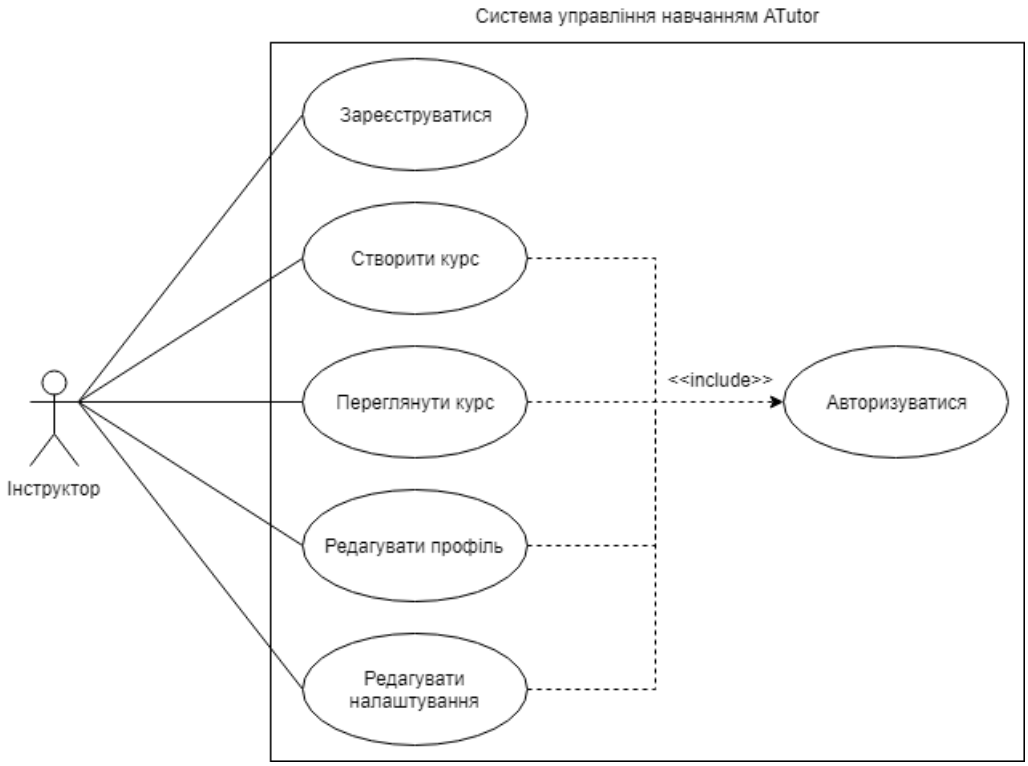

Рисунок 1.1 – Діаграма варіантів використання для Інструктора

Таблиця 1.1 – Опис варіантів використання Інструктора

Варіант використання	Опис
Зареєструватися	Для отримання доступу до системи потрібно щоб користувач мав можливість зареєструватися. Після завершення процесу реєстрації, користувачеві не потрібно підтверджувати електронну пошту, так що користувач може відразу ж увійти до системи

Створити курс	Інструктори мають можливість створити курс. Буває 3 рівні доступу до курсу: 1. Публічний – загальнодоступний для всіх користувачів, які зареєструвалися. Запит на участь в курсі не є обов'язковим; 2. Приватний – відкритий для всіх користувачів, які зареєструвалися. Для курсу потрібно вводити реєстраційну інформацію. Запит на участь в курсі не є обов'язковим; 3. Захищений – відкритий для всіх користувачів, які зареєструвалися. Для курсу потрібно вводити реєстраційну інформацію. Інструктор повинен відтвердити вашу заявку на участь в курсі
Переглянути курс	Інструктор може шукати доступні курси
Редагувати профіль	Інструктор може відредагувати свій профіль
Редагувати налаштування	Інструктор може налаштувати зовнішній вигляд системи, змінюючи теми, шрифт та навігацію

На рисунку 1.2 та в таблиці 1.2 наведено більш детальний опис варіанту використання Створити курс.

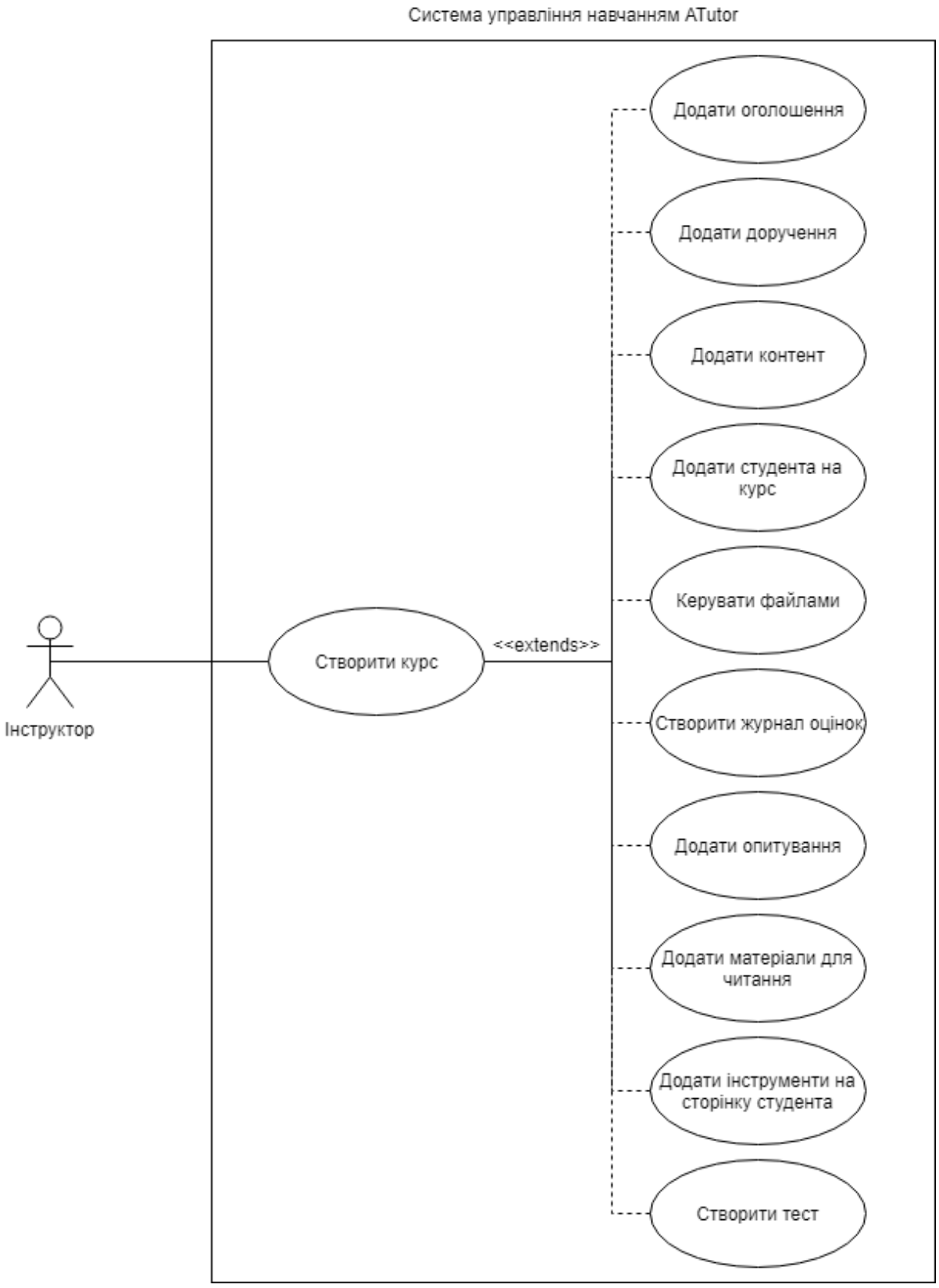

Рисунок 1.2 – Детальний опис варіанту використання Створити курс

Таблиця 1.2 – Детальний опис варіанту використання Створити курс

Варіант використання	Опис
Додати оголошення	Інструктор може додати оголошення
Додати доручення	Інструктор може додати доручення Студентам

Додати контент	Інструктор може створювати та додавати різноманітні навчальні матеріали
Додати студента на курс	Інструктор може затвердити Студентську заявку на участь в курсі
Керувати файлами	Інструктор може керувати матеріалами для навчання
Створити журнал оцінок	Інструктор може створити журнал оцінок для Студентів
Додати опитування	Інструктор може додати опитування
Додати матеріали для читання	Інструктор може додати додаткові матеріали для читання, необхідні для поглиблення в курс
Додати інструменти на сторінку Студента	Інструктори можуть додавати інструменти на сторінку Студента, які сприятимуть для подальшого його стимулювання
Створити тест	Викладачі можуть створити тести
Створити резервну копію	Інструктор може зробити бекап курсу для збереження поточного стану курсу
Надіслати листа	Викладач може надсилати електронні листи Студентам/Інструкторам
Налаштувати меню курсу	Інструктор може додати та налаштувати структуру меню, що розміщується на сторінці курсу
Управління форумом	Інструктор може створювати та адмініструвати форуми

На рисунку 1.3 та в таблиці 1.3 наведено детальний опис варіанту використання Редагувати налаштування.

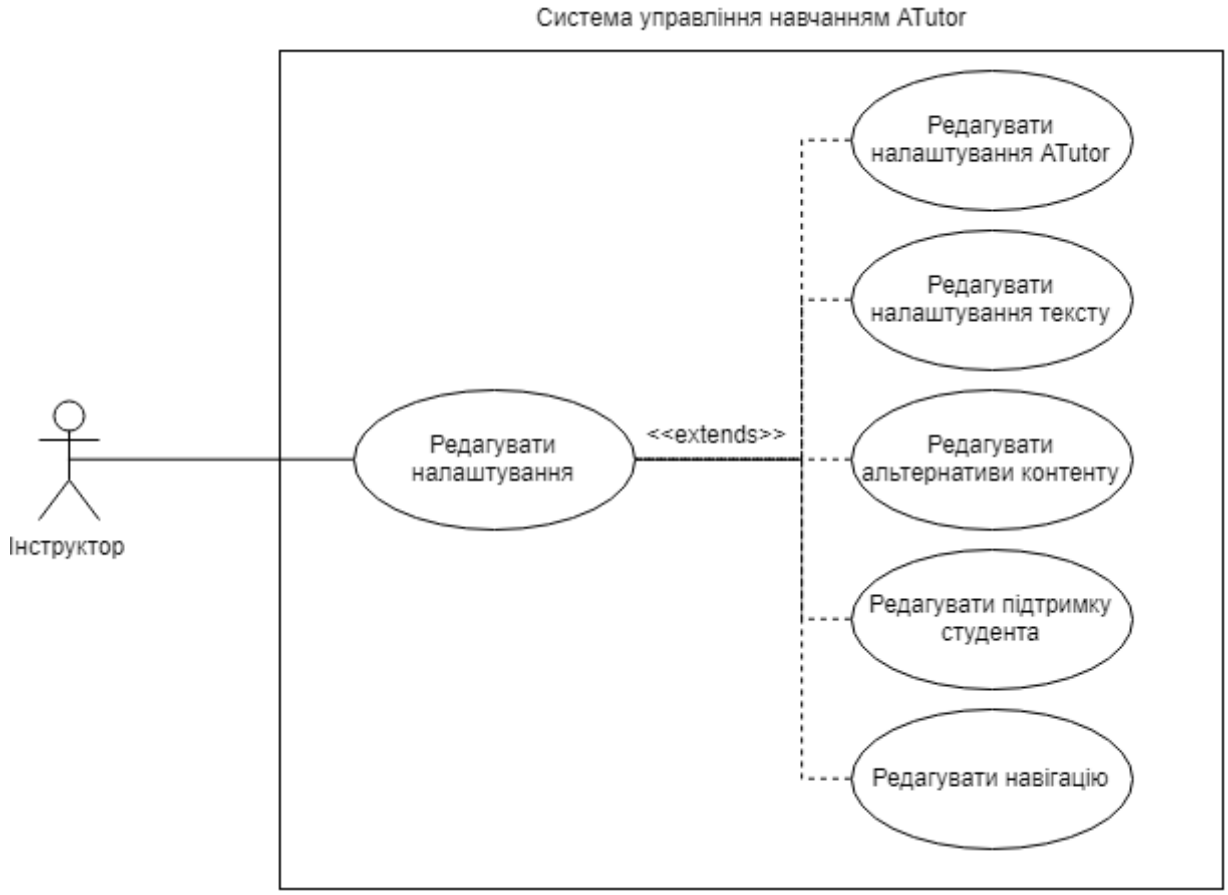

Рисунок 1.3 – Опис варіанту використання Редагувати налаштування

Таблиця 1.3 – Опис варіанту використання Редагувати налаштування

Варіант використання	Опис
1	2
Редагувати налаштування ATutor	Інструктор має змогу змінювати графічне оформлення системи (тему), часовий пояс, вмикати/вимикати сповіщення про нові повідомлення, ввімкнути функцію автологіну, тощо

Продовження таблиці 1.3

1	2
Редагувати налаштування тексту	Інструктор може змінювати шрифт, його розмір, а також колір тексту та його фону

Редагувати альтернативи контенту	Під альтернативою мається на увазі текст, який відображатиметься у випадку, якщо медіа (зображення, текст, звук) не вдалося завантажити. Інструктор може налаштовувати поведінку альтернатив, а також задавати мову, на якій альтернативи будуть відображатися у такому випадку
Редагувати підтримку Студента	Інструктор може надавати додаткові інструменти Студентам. До інструментів можна віднести: словник, енциклопедію, атлас, записник та калькулятор
Редагувати навігацію	Інструктор має можливість змінювати навігацію по курсу. Навігаційні елементи включають в себе сторінку Зміст для курсу, навігаційні кнопки типу Вперед/Назад, навігаційний ланцюг (breadcrumb)

Студент має майже аналогічні варіанти використання Інструктора, за винятком неможливості використання Створити курс, адже Студентам не надається відповідних прав. Вони лише можуть записуватися на курс та навчатися на ньому.

На рисунку 1.4 та таблиці 1.4 зображено та детально описано варіант використання Переглянути курс для Студента.

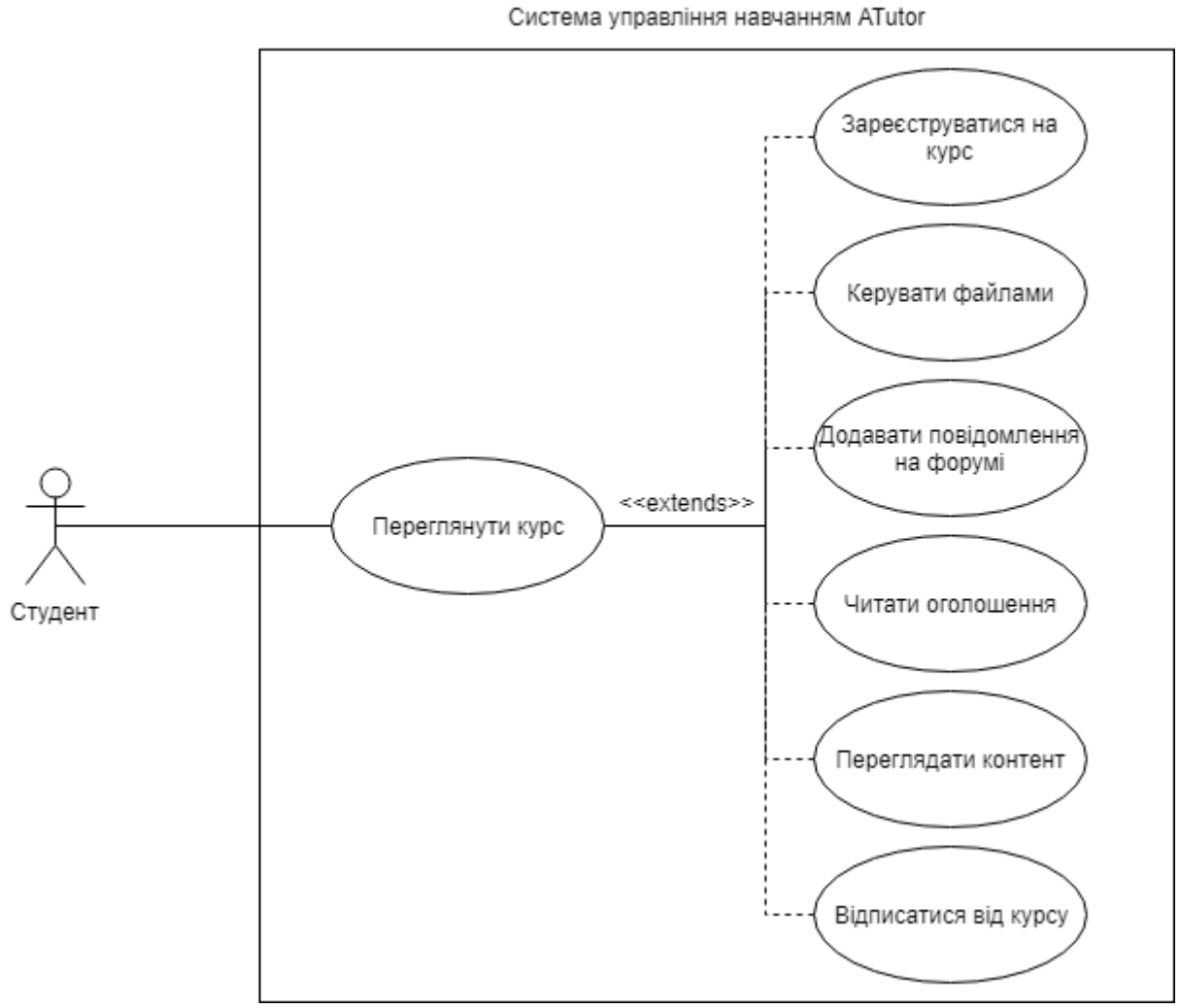

Рисунок 1.4 – Варіант використання Переглянути курс для Студента

Таблиця 1.4 – Деталізований варіант використання Переглянути курс

Варіант використання	Опис
1	2
Зареєструватися на курс	Студенти можуть підписатися на потрібний курс, пдля того щоб Інструктор отримав заявку на участь в курсі, та схвалив або відхилив її

Продовження таблиці 1.4

1	2
Керувати файлами	Студент може керувати збереженням файлів. Вони можуть завантажувати файли на сервер, а також скачувати їх собі на комп’ютер

Додати повідомлення на форумі	Студент може розміщувати повідомлення на форумі
Читати оголошення	Студент може прочитати оголошення, яке залишив Інструктор
Переглядати контент	Студент може переглядати навчальні матеріали, які розмістили в межах курсу.
Відписатися від курсу	Студент може за бажання відписатися від курсу

1.2 Роль тестування у життєвому циклі програмного забезпечення

Тестування програмного забезпечення – процес аналізу програмного продукту для виявлення відмінностей між існуючими та необхідними умовами (тобто дефектів) та оцінки особливостей програмного продукту[24]. Тестування програмного забезпечення дає змогу забезпечити об'єктивний, незалежний погляд на тестоване програмне забезпечення, що у свою чергу дозволяє компанії оцінити та зрозуміти ризики впровадження такого програмного забезпечення. Методи тестування включають в себе процес тісної роботи з програмою або додатком з метою виявлення програмних помилок та перевірки того, що програмний продукт підходить для використання.

Тестування передбачає виконання програмного компонента або компонента системи для оцінки однієї або декількох властивостей. Загалом, ці властивості вказують на те, наскільки випробуваний компонент чи система:

- Виконує вимоги, якими керуються при його проектуванні та розробці;
- Правильно реагує на передбачені типи вхідної інформації;
- Може бути встановлена і запускатися в різних середовищах;
- Досягає результату, обговореного зацікавленими сторонами (замовниками та користувачами).

Так як кількість потенційних тестів для навіть найпростіших програмних компонентів майже нескінченна, у всіх автоматизованих тестах використовується

певна лише певний набір тестів, які є найбільш оптимальнми для наявного часу та ресурсів. Як результат, тестування програмного забезпечення, як правило, (але не завжди) намагається протестувати програму або додаток з наміром знайти помилки програмного забезпечення (неточності або інші дефекти). Тестування є ітераційним процесом, оскільки коли виправлено один дефект, він може висвітлити нові помилки.

Розділімо основне визначення тестування програмного забезпечення на наступні частини:

1. Процес: тестування – це процес, а не окрема активність;
2. Усі функції життєвого циклу: Тестування – це процес, який відбувається протягом всього життєвого циклу розробки програмного забезпечення (SDLC);
3. Статичне тестування: воно може протестувати та знаходити дефекти без виконання коду. Статичне тестування виконується під час процесу верифікації. Це тестування включає в себе перевірку документів (включаючи вихідний код) та статичний аналіз.
4. Динамічне тестування: у динамічному тестуванні програмний код виконується, щоб продемонструвати результат виконання тестів. Воно виконується під час процесу валідації. Наприклад: тестування модулів, інтеграційне тестування, тестування системи тощо;
5. Планування: необхідно планувати ті частини роботи, які планується зробити.
6. Підготовка: потрібно вибрати тип тестування, яким планується займатися, шляхом вибору умов тестування та розробки тестових випадків;
7. Оцінка: під час проведення оцінки необхідно перевірити результати та оцінити тестування програмного забезпечення у контексті критерії завершення, що допоможе вирішити, чи програмний продукт успішно пройшов тестування.

1.2.1 Мета та основні цілі тестування програмного забезпечення

Тестування програмного забезпечення є необхідним, тому що ми всі помиляємося. Деякі з цих помилок мінорні, але деякі з них можуть нести загрозу усьому розроблюваному програмному забезпеченню. Тому для уникнення вищеописаних ситуацій варто перевіряти те, що ми розробляємо.

Тестування програмного забезпечення є важливою частиною розробки програмного забезпечення тому що:

1. Тестування необхідне, адже воно вказує на дефекти та помилки, які були зроблені під час фази розробки;
2. Воно важливе, адже гарантує створення довіри між замовником та компанією. Надійне та правильно працююче програмне забезпечення сприяє підвищенню рівня довіри до IT-компанії;
3. Тестування необхідне для надання замовникам послуг, наприклад, постачання високоякісного продукту або програмного забезпечення, що потребує менших витрат на технічне обслуговування;
4. Тестування необхідне для ефективного виконання програмного забезпечення або продукту [25].

Основними цілями програмного тестування є:

1. Пошук дефектів, які програміст може створити під час розробки програмного забезпечення;
2. Отримання впевненості та надання інформації про рівень якості;
3. Запобігання дефектам створеним під час розробки;
4. Переконатися, що кінцевий результат відповідає вимогам бізнесу та користувачам;
5. Переконатися, що програмне забезпечення задовольняє специфікацію вимог до бізнесу (BRS) та специфікацію системних вимог (SRS) [26];

Тестування допомагає завершити розробку програмного забезпечення або продукту відповідно до вимог бізнесу та користувачів. Дуже важливо мати хороше покриття програмного забезпечення різними тестами, щоб повністю

протестувати програмне забезпечення та переконатися, що воно правильно працює й відповідає специфікаціям.

1.2.2 Види тестової документації

Процес тестування програмного забезпечення включає в себе кілька основних артефактів:

1. План випробувань (тест план);
2. Матриця відстеження (трасування);
3. Варіант тестування (тест кейс);
4. Тестовий скрипт;
5. Тестовий набір;
6. Тестові дані.

План випробувань – документ, що описує обсяг, підхід, ресурси та графік передбачуваних тестових заходів. Він визначає серед інших тестових елементів, функції, що підлягають тестуванню, завдання тестування, хто буде виконувати кожне завдання, ступінь незалежності тестера, середовище тестування, методи проектування тесту та критерії входу та виходу, що використовуються, та обґрунтування їх вибір та будь-які ризики, що потребують планування дій у надзвичайних ситуаціях[27]. План тестування може бути частиною широкої “тестової стратегії”, яка документує загальні підходи до тестування.

Матриця відстеження – це таблиця, яка співвідносить вимоги або проектні документи для тестові випадки. Вона використовується для зміни тестів, коли змінюються відповідні вихідні документи, а також для вибору тестів для виконання при плануванні регресійного тестування шляхом розгляду вимог покриття.

Тестовий випадок – це документ, який складається з певного набору умов або певних дій, які виконуються над системою для перевірки системи на очікувані результати. Тестовий випадок, як правило, складається з унікального ідентифікатора, посилань на вимоги із специфікації проекту, передумов, подій, серії кроків (також відомих як дії), які слід виконувати, вхідної та вихідної

інформації та очікуваного результату[28]. Він навіть ніколи може виглядати як "для умови Z ваш кінцевий результат являється F". Правда зазвичай тестові випадки більш докладно описують сценарії виконуваних дій та які результати потрібно очікувати. Іноді це може бути серія кроків (але часто кроки містяться в окремій процедурі тестування, яка може бути застосована для декількох випадків тестування, що забезпечує кращу підтримуваність тестової документації), але з одним очікуваним результатом. Більші тестові випадки також можуть містити інструкції, які повинні бути виконані до початку виконання тестового випадку (так звані передумови). Тест кейс можна зберігати у документах, таблицях, базах даних, репозиторіях або інших загальних сховищах. У системі баз даних ви також можете бачити минулі результати тестування, і яка конфігурація системи використовувалася для створення цих результатів. Ці попередні результати зазвичай зберігаються в окремій таблиці.

Тестовий скрипт – це процедура або програмний код, який повторює дії користувача. Це коротка програма, написана на мові програмування, використовується для перевірки частини функціональності програмної системи. Тестові скрипти, можуть бути записані за допомогою спеціального автоматизованого функціонального тестового інструменту (наприклад, HP QuickTest Professional, Borland SilkTest) або добре відомими мовами програмування (наприклад, C ++, C # Tcl, Expect, Java, PHP, Go, Perl, Powershell, Bash, Python або Ruby).

Тестовий набір являється набіром тестових випадків, які потрібні для тестування програмної системи, щоб визначити чи працюють певні компоненти системи. Набір тестів зазвичай містить докладні кроки та інструкції для кожної збірки тестових випадків та інформацію про конфігурацію системи, яка буде використовуватися під час тестування. Група тестових випадків також може містити спільні передумови чи кроки та опис наступних тестів.

Тестові дані. У більшості випадків декілька наборів значень або файлів з данними які використовуються для тестування однакових функцій. Всі тестові значення та змінні компоненти навколишнього середовища збираються в окремі файли та зберігаються як дані тесту. Також можна показати ці дані клієнту та надати їх разом із продуктом. Створення тестових даних – це ще одна важлива

частина тестування програмного забезпечення. Цей процес являється створенням набору даних для перевірки адекватності нових або існуючих програмних додатків. Це можуть бути як фактичні дані, взяті з попередніх операцій, або у розробників так і штучні дані, спеціально створені для цієї мети.

1.3 Аналіз можливих шляхів тестування програмної системи

Програмне забезпечення тестується протягом усього часу розробки і підтримки програмного забезпечення. Тому існує поділ на ступені тестування. Ступінь тестування - те, над чим проектуються та виконуються тестові сценарії: над певним модулем, спільнотою модулів або цілою системою. Тестування всіх ступенів системи – це гарантія вдалої імплементації і завершення проекту, та створенню якісного ПЗ.

Ступені тестування поділяються на:

- Тестування на вимоги;
- Тестування модулів;
- Тестування системи;
- Тестування інтеграції.

Тестування модулів виконує перевірки функціональності модуля та вишукає проблеми в частинках програми, які являються загальнодоступними і тестуються окремо (функції додатку, класи, інтефейси і т.д.).

У більшості випадків тестування модулів виконується викликаючи функції, які мають бути перевірені використовуючи середовище розробки, за допомогою інструментів таких як фреймворки для тестування модулів або інструменти для відлагодження. Дефекти які були знайдені, завзичай виправляються в коді уникаючи їх опису в системах багів.

Одним з найефективніших підходів до тестування модулів системи вважається – створення автоматизованих скриптів перед розроблюванням кінцевого продукту. Цей підхід називається “тестування спочатку”. Цей підхід передбачає створення й інтегрування невеличких шматочки коду, які і будуть цілями тестів, написаних до початку розробки. Процес розробки та виправлень буде вестися доти, доки всі тести не будуть успішно пройдені [29].

Тестування інтеграції передбачає перевірку зв’язків між різними системними компонентами і взаємодії з рештою частин системи (включаючи операційну систему, додаткове обладнання яке забезпечує зв’язок між різними компонентами системами тощо).

Ступені тестування інтеграції:

- Інтеграційний рівень компонентів, виконує перевірку взаємодії між різноманітними програмними системами після закінчення проведення компонентного тестування;
- Інтеграційний рівень системи, виконує перевірку взаємодії між різноманітними системами після закінчення проведення системного тестування [30].

Тестування системи – тестування проводиться в умовах специфікації системних вимог (SRS) та/або специфікацій функціональних вимог (FRS). Це остаточне випробування, щоб перевірити, чи відповідає товар, що поставляється, вимогам, зазначеним у документі вимоги. Він повинен досліджувати як функціональні, так і нефункціональні вимоги. Натепер розрізняють два основних підходи до тестування системи:

- По базі вимог – Кожна вимога описується тестовими випадками, які виконують перевірку належного виконання даної вимоги;
- За базою можливих випадків використання – Зважаючи на можливі варіанти використання програмної системи створюється список найбільш загальних способів користування системою. Маючи певний випадок користування системою, є можливість спроектувати декілька тестових сценаріїв для реалізації. Щоб виконати перевірку якогось з описаних сценаріїв використання пишуться тестові сценарії системи, які мусять бути ретельно протестовані [31].

Тестування на вимоги проводиться для того щоб зрозуміти, чи може програмна система виконати поставлені до неї вимоги. Фінальне рішення зазвичай приймається клієнтом який замовив програмне забезпечення або іншою уповноваженою особою.

Рішення про необхідність проведення тестування на вимоги обговорюється та узгоджується, в той момент коли продукт має необхідний рівень якості та замовник був ознайомлений з орієнтовним планом приймальних робіт (Product Acceptance Plan) або іншим документом, в якому наявний детально описаний набір дій, пов'язаних з проведенням приймального виду тестування, дати проведення, відповідальних і т.д [32].

1.4 Автоматизація тестів

Автоматизація тестів – це набір процесів, який перевіряє, чи програмне забезпечення функціонує належним чином та відповідає вимогам до його випуску у виробництво. Цей метод тестування програмного забезпечення використовує сценарії сценаріїв, які виконуються інструментами тестування. Автоматизовані інструменти тестування виконують перевірку програмного забезпечення, повідомляють результати та порівнюють результати з попередніми тестовими пробігами.

Коли тестер програмного забезпечення вручну перевіряє систему, він може помилитися, особливо коли програма містить сотні-тисячі рядків коду. Автоматизація допомагає команді мануальних тестувальників уникнути цих помилок людини при тестуванні додатків та виконує перевірки в більш швидкі часові рамки, ніж якщо це було зроблено особисто.

Деякі засоби автоматизації тестування мають можливості звітування, які реєструють кожен тестовий скрипт, щоб показати користувачам статус кожного тесту. Потім тестер може порівняти результати з іншими звітами, щоб оцінити, як працює програмне забезпечення порівняно з очікуваннями та вимогами.

Загалом, автоматизоване тестування дозволяє персоналу уникати ручних тестів та зосереджуватися на інших пріоритетах проекту. Команда з контролю якості може повторно використовувати автоматизовані тестові сценарії, щоб забезпечити виконання кожної перевірки кожен раз однаково. Крім того, автоматичне тестування допомагає команді швидко знаходити помилки на ранніх стадіях розвитку, що може скоротити загальний робочий час та витрати на проект

Для тестування програмного забезпечення існує безліч методів, деякі з них описані нижче:

- User Interface Testing (Тестування графічного інтерфесу користувача). Головним чином полягає у забезпеченні правильного функціонування інтерфейсу користувача, що додаток дотримується його письмових специфікацій та виявлення дефектів. Крім цього, ми перевіряємо, чи хороші елементи дизайну. Це включає перевірку екранів за допомогою елементів керування, таких як панелі меню, панелі інструментів, кольори, шрифти, розміри, піктограми, вміст, кнопки тощо;
- API Testing (Тестування АПІ). Зазвичай призначене для тестування серверної частини додатку та полягає у надсиланні запитів до серверу. Часто виконується такими інструментами як Postman або RestSharp.

Часто інструменти для автоматизації тестів не є безкоштовними тому усі тестові сценарії немає змоги. У таких випадках автоматизоване тестуванн поєднується з мануальним тестуванням. Автоматизивуються зазвичай такі тестові випадки які призначені для загального програмного використання, оскільки вони мають бути виконані кожного разу, коли в додатку відбулась зміна. Таким чином досягається економія ресурсів компанії, так як потреба у великій кількості ручних тестувальників відпадає.

Для правильної та швидкої роботи розробних автоматизованих тестів мусить мати хороший рівень володіння мовою або мовами програмування для написання автоматизованих тестів.

1.4.1 Необхідність автоматизації

Автоматизація звичайно зменшує час для тестування, але існують сценарії які неможливо автоматизувати або потребуть значних зусиль для написання автоматизованих сценаріїв виконання. Тому розуміти необхідність у автоматизації є дуже критичним завданням для досвідченого тестувальника.

Нижче наведені найбільш часті випадки, коли автоматизоване тестування є необхідним:

- Димне тестування (Smoke Testing) – Для того щоб дізнатись про загальний поточний стан стистеми та можливі помилки;
- Регресійне тестування (Regression Testing) – Для того щоб перевірити наявний функціонал на сумісніть з новою версією програмного продукту;
- Тестування статичних компонентів (Static Testing) – Для того щоб виконати перевірку компонентів які практично завжди незмінні;
- Тестування на основі даних (Data-Driven Testing) – Для тестування компонентів системи яка приймає різні типи вхідних даних
- Тестування навантаження(Load Testing) – Для перевірки максимально допустимої загрузки на сервер без виклику критичних помилок та є важко виконуваним видом тестування для ручного виконання;
- Тестування продуктивності(Perfomance Testing) – Для перевірки швидкості відклику та продуктивності системи за оптимальних умов.

Для полегшення проведення автоматизації існуючих ручних сценаріїв дуже корисними є різноманітні програми та застосунки призначені для спрощення автоматизації. До таких можна віднести такі застосунки як Ranorex(дозволяє виконувати тестування усього, навіть операційної системи), Visual Studio(слугує для написання коду в цілому), Selenium(дозволяє записувати дії користувача у веб браузері), Postman(має можливість надсилати веб запити) тощо.

1.4.2 Переваги автоматизації

До основних переваг автоматизації можна віднести:

- Статичність – Написані тести завжди будуть виконувати одну й ту саму послідовність дій незалежну від людського фактору що у свою чергу виключить помилки пов’язані з людським фактором.
- Швидке виконання – Автоматизованому тесту не потрібно звірятись з інструкціями чи документацією так як він має визначений набір кроків;

- Автоматична система репортування – При потребі є можливість імплементувати автоматизовану систему розсилання звітів яка буде виконуватись після кожного виконання тестів;
- Цілодобова підтримка системи – При наявному наборі авто тестів тестування та підтримка системи може проводитись у будь який час.

1.4.3 Недоліки автоматизації

До основних мінусів автоматизації можна віднести:

- Статичність – На жаль також є недоліком, так як виконуючи один і то самий набір інструкцій тест не буде звертати увагу на не передбачені помилки;
- Висока початкова вартість – Для отримання автоматизованого набору тестів потрібна велика кількісь як людських так і матеріальних ресурсів.
- Необхідність у інструментах для автоматизації – Для написання якісних автоматизованих тестових сценаріїв зазвичай потребуються якісні ліцензійні інструменти. Альтернативою виступають безкоштовні інструменти для розробки, але вони зазвичай містять велику кількість помилок та мізерну підтримку;

2. СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Налаштування локальної версії ATutor

Так як система управління електронним навчанням ATutor є програмою з відкритим кодом та є безкоштовною ми маємо можливість налаштувати її на локальному комп’ютері, для проведення подальшого тестування. Для досягнення цієї цілі нам потрібен інструмент для створення віртуальних контейнерів. Одним з таких інструментом являється Docker.

2.1.1 Інструмент для створення віртуальних контейнерів Docker

Docker – програмне забезпечення для автоматизації розгортання і управління додатками в середовищах з підтримкою контейнеризації. Дозволяє «упакувати» додаток з усім його оточенням і залежностями в контейнер, який може бути перенесений на будь-яку Linux-систему з підтримкою cgroups в ядрі, а також надає середовище з управління контейнерами. [33].

Контейнери дозволяють розробнику пакувати додаток з усіма необхідними йому частинами, такими як бібліотеки та інші залежності, і доставляти все це як один пакет. Тим самим, завдяки контейнеру, розробник може бути впевнений, що програма буде працювати на будь-якій іншій машині Linux незалежно від будь-яких налаштованих параметрів, які можуть мати машини, які можуть відрізнятися від машини, що використовується для написання та тестування коду.

Певним чином Докер трохи схожий на віртуальну машину. Але на відміну від віртуальної машини, а не для створення цілої віртуальної операційної системи, Docker дозволяє програмам використовувати те саме ядро Linux, що і система, в якій вони запущені, і вимагає лише доставки програм із речами, які вже не працюють на хост-комп'ютері. Це дає значне підвищення продуктивності та зменшує розмір програми.

Основні переваги Docker:

- Стандартизація та продуктивність;
- Сумісність та ремонтопридатність;

- Простота та швидша конфігурація;
- Швидке розгортання;
- Безперервне розгортання та тестування:
- Ізоляція [34].

Стандартизація та продуктивність. Docker - контейнери забезпечують узгодженість у декількох циклах розробки та випуску, стандартизуючи ваше середовище. Однією з найбільших переваг архітектури на основі Докера є насправді стандартизація.

Docker забезпечує повторювані середовища для розробки, побудови, тестування та виробництва. Стандартизація сервісної інфраструктури по всьому трубопроводу дозволяє кожному члену команди працювати в середовищі паритету виробництва. Роблячи це, інженери більш оснащені для ефективного аналізу та виправлення помилок у програмі. Це зменшує витрачену кількість часу на дефекти та збільшує кількість часу, доступного для розробки функції.

Сумісність та ремонтопридатність. Усуньте проблему "це працює на моїй машині" раз і назавжди. Однією з важливих переваг, є паритет. Паритет, з точки зору Docker, означає, що ваші зображення системи працюватимуть однаково, незалежно від того, на якому сервері чи ноутбуці вони працюють. Для розробників це означає, що менше часу витрачається на налаштування середовищ, налагодження проблем, пов’язаних із оточенням, та більш портативну та просту в налаштуванні базу даних коду. Паритет також означає, щоваша виробнича інфраструктура буде надійнішою та легшою в обслуговуванні.

Простота та швидша конфігурація. Однією з ключових переваг Докера є спосіб його спрощення. Користувачі можуть приймати власну конфігурацію, вводити її в код і без проблем розгортати її. Оскільки Docker можна використовувати в найрізноманітніших середовищах, вимоги інфраструктури більше не повязані із середовищем програми.

Швидке розгортання. Docker дозволяє скоротити розгортання до секунд. Це пов’язано з тим, що він створює контейнер для кожного процесу і не завантажує ОС. Дані можна створювати та знищувати без побоювання, що вартість їх відновлення буде вище, ніж доступна.

Безперервне розгортання та тестування. Docker забезпечує послідовне середовище від розробки до виробництва. Докер-контейнери налаштовані на підтримку всіх конфігурацій та залежностей внутрішньо; ви можете використовувати один і той же контейнер від розробки до виробництва, переконуючись у відсутності розбіжностей або ручного втручання.

Ізоляція. Docker забезпечує ваші програми та ресурси ізольованими та відокремленими. Docker переконує, що кожен контейнер має свої власні ресурси, які ізольовані від інших контейнерів. Ви можете мати різні контейнери для окремих додатків, які працюють із абсолютно різними стеками. Docker допомагає вам забезпечити чисте видалення програми, оскільки кожна програма працює у власному контейнері. Якщо вам більше не потрібна програма, ви можете просто видалити її контейнер. Він не залишить жодних тимчасових або конфігураційних файлів у вашій хост-операційній системі.

Docker складається з наступних компонентів:

- Docker daemon(Docker демон), який запускається на машині яка його виконує (якщо це Лінукс), або всередині Hyper-V (якщо це Windows);
- Docker image (Docker образ) – містить обмежену версію операційної системи, програму і всі її залежності. Образи в Docker є багатошаровими. Це дозволяє наприклад скористатись шаром з операціною системою та шаром з веб сервером одночасно уникнувши використання лишнього. Образами можна обмінюватись через DockerHub;
- Клієнта, через який виконуюється основна взаємодія з демоном;
- Контейнер Docker – це запущений образ. Контейнери Docker можна запускати, спиняти, переміщувати, клонувати і видаляти. Також можна зняти образ, або зробити знімок з поточного стану контейнера [35].

2.1.2 Розгортання системи електронного навчання ATutor

Для розгортання системи електронного навчання ATutor потрібно декілька програмних компонентів, тому для їх встановлення та налаштування нам потрібно використати складний Docker образ написаний у окремому Dockerfile.

Образ створюється за допомогою консольної команди docker build. В якості аргументу потрібно вказати шлях до Dockerfile.

Dockerfile скрипт з переліком усіх команд та необхідний для створення образу ATutor, продемонстрований у Додатку В.

У цьому скрипті виконуються наступні команди:

1. Розгортається контейнер із середовищем для мови PHP;
2. Відбувається налаштування на PHP-сервері необхідних компонентів;
3. Відбувається встановлення порту для контейнеру під номером 80. Це потрібно для з'єднання контейнеру з системою ATutor із базою даних;
4. Виконується завантаження коду програмної системи ATutor, та копіюється у вказану директорію в контейнері;
5. Створюється папка, у якій ATutor у подальшому буде зберігати медіа-файли, завантажені користувачами системи.

Першим кроком потрібно створити Dockerfile який відповідає по змісту Додатку В, відкрити командну стрічку у директорії, де було створено Dockefile, виконати команду наведену нижче та яка створить образ під назвою " atutor_container":

```
docker image build -t atutor_container .
```

Наступним кроком потрібно створити мережу в межах Docker середовища, у якій будуть існувати контейнери з ATutor та базою даних. Таким чином, контейнери матимуть змогу взаємодіяти один з одним.

Для створення мережі із назвою "docker_network" потрібно викнати наступну команду:

```
docker network create --driver bridge docker_network
```

Після налаштування мережі та створення образу з ATutor, потрібно запустити створений образ, а також образ з базою даних MySql, з'єднавши контейнери спільною мережею, забезпечивши комунікацію між ними.

Запуск образу бази даних виконується наступною команди:

```
docker run --network=docker_network -e MYSQL_ROOT_PASSWORD=qwerty
--name mysql -d mysql:5.6;
```

У цій команді вказується значення для змінної середовища MYSQL_ROOT_PASSWORD, яка виконує роль паролю для адміністратора бази даних.

Останньою командою у командному рядку, яка служить запуском налаштованого образу є:

```
docker run --network=docker_network -p 80:80 --name atutor_container -d
atutor_container
```

Ця команда також виконує співставлення віртуального порту контейнеру під номером 80, з портом 80 локального комп'ютера, для забезпечення комунікації. Тому тепер якщо перейти по адресі localhost:80 нас переадресує на середовище ATutor.

2.1.3 Виконання первинних налаштувань системи ATutor

Після виконання усіх вищеописаних кроків ми можемо розпочати налаштування системи ATutor. Наступними кроками стануть:

- Налаштування з'єднання із MySql базою даних;
- Створення первинних користувачів (системний адміністратор та інструктор);
- Додаткові налаштування системи.

Для початку роботи давайте перейдемо на ATutor. Для цього наберемо у браузері “localhost”. На рисунку 2.1 зображена сторінка, яка повинна появитись після переходу по вказаній адресі, якщо система ще не була налаштована.

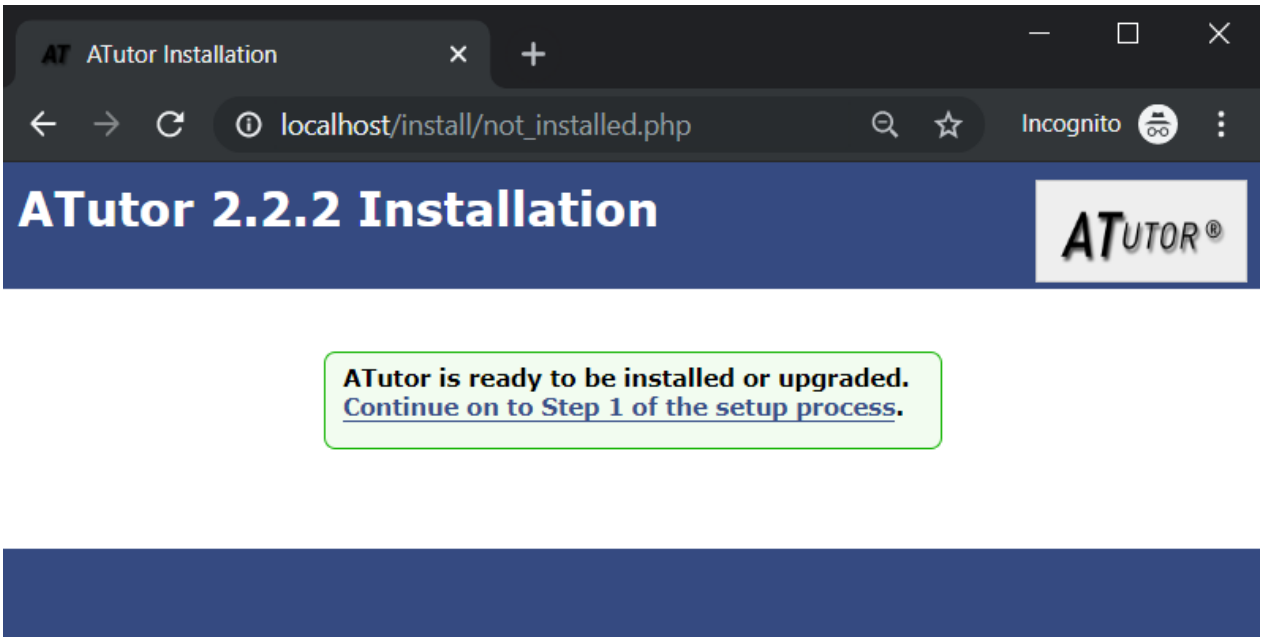

Рисунок 2.1 – Базова сторінка для встановлення ATutor

Натиснемо на “Continue on to Step 1 of the setup process”, для початку процесу встановлення. Після натискання з’явиться сторінка з привітанням, та декількома опціями встановлення ATutor. У разі уже встановленої системи ATutor є можливість модифікувати налаштування, для цього унизу є окрема кнопка. Наразі нас цікавить первинне налаштування ATutor.

Також, ця сторінка містить таблицю з списком компонентів, які необхідні для встановлення та успішного функціонування системи.

До основних пунктів вимог можна віднести наступні:

- PHP 5.0.2+;
- Mbstring;
- Mysql;
- safe_mode = Off;
- file_uploads = On.

Повний їх перелік вказаний на рисунку 2.2.

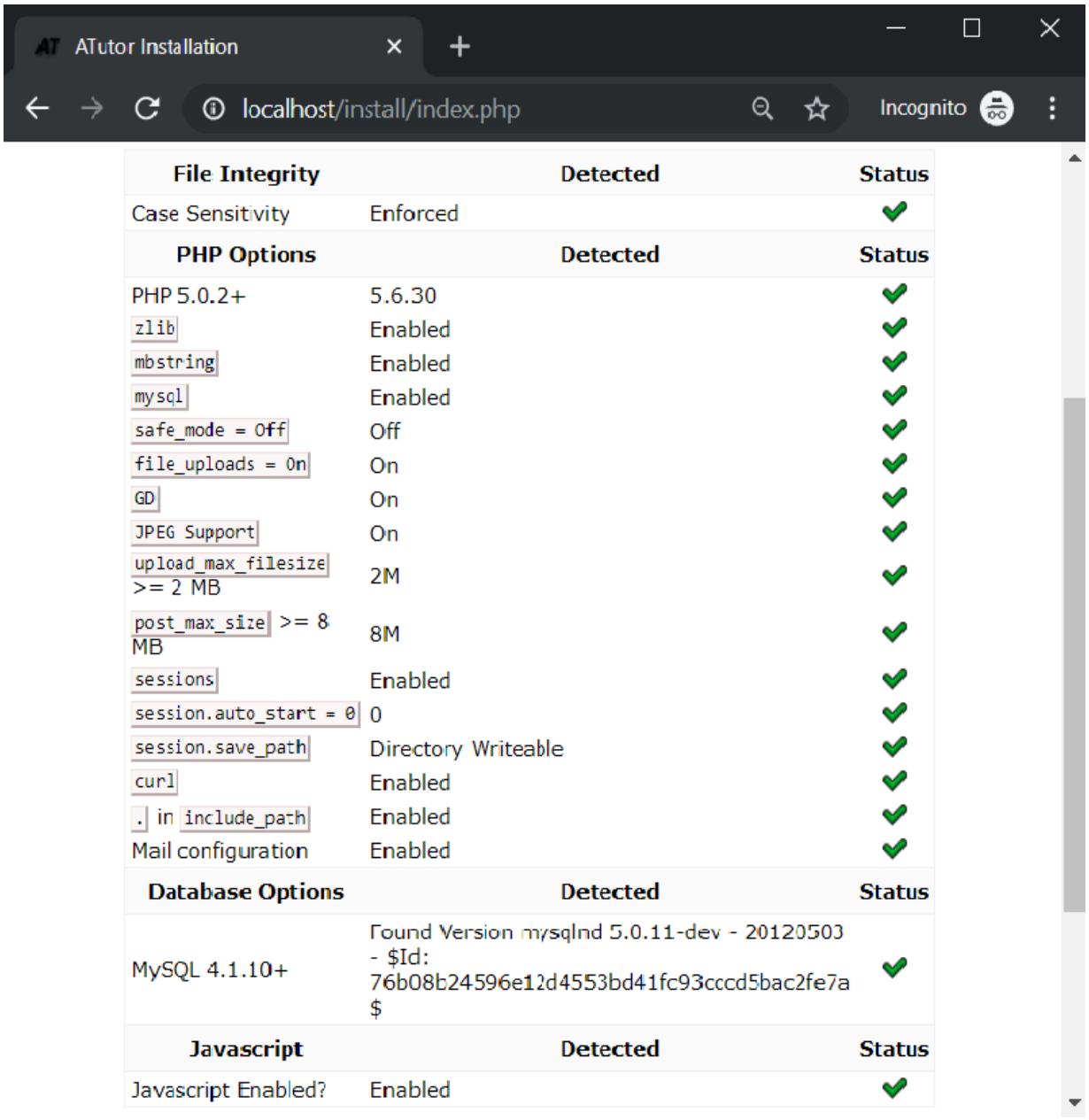

Рисунок 2.2 – Необхідні компоненти для встановлення ATutor

Так як це первинне встановлення системи ATutor, натиснемо кнопку “Install”. Далі нас переадресує на наступний крок, а саме прийняття угоди про використання системи ATutor (рисунок 2.3).

Рисунок 2.3 – Умови використання ATutor

Після детального ознайомлення із умовами використання, потрібно натиснути “I Agree” для подальшого процесу встановлення.

Наступним кроком стане налаштування доступу до бази данних. Так як база даних розгорнута на окремому Docker контейнері, та знаходиться в одній мережі із ATutor контейнером, у нас є можливість отримати доступ до контейнеру за адресою mysql.docker_network. Звернення має відбувається до стандартного порту MySql, а саме 3306 порту, адже це порт, який відкривається по замовчуванню MySql контейнером. Також, MySql контейнер надає автоматично створеного користувача з адміністраторськими правами, який має змогу створити необхідні бази даних. Стандартний логін ім’я адміністратора – root. Пароль було вказано під час запуску MySql контейнеру використовуючи параметр -e. Отже, пароль – qwerty. Рисунок 2.4 показує форму з остаточними налаштування бази даних.

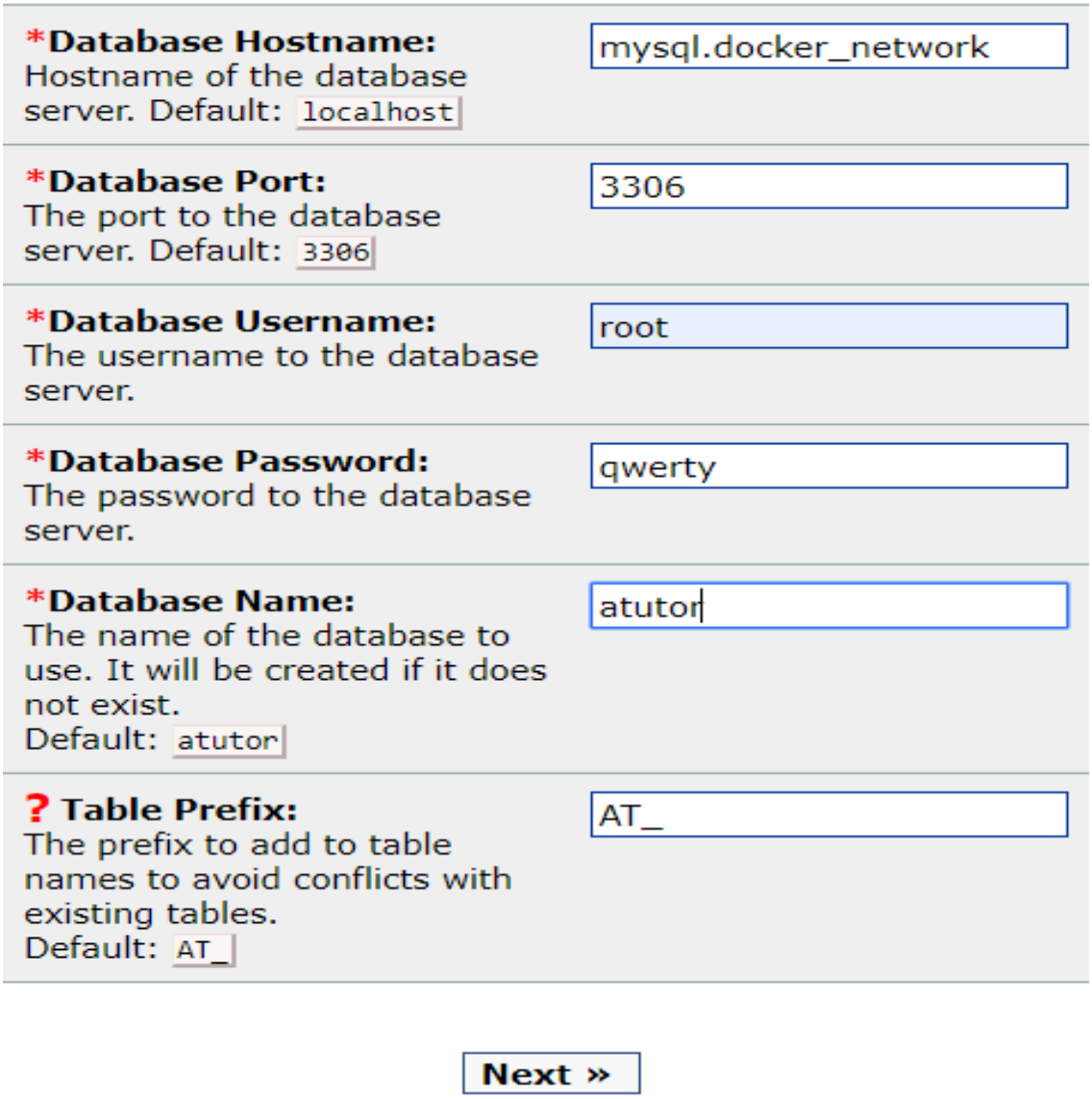

Рисунок 2.4 Налаштування бази даних

Після натискання кнопки “Next”, ми потрапимо побачимо сторінку з повідомленням про успішне або неуспішне з’єднання із базою даних (рисунок 2.5). Також, буде виведена детальна інформація про створені таблиці у базі даних.

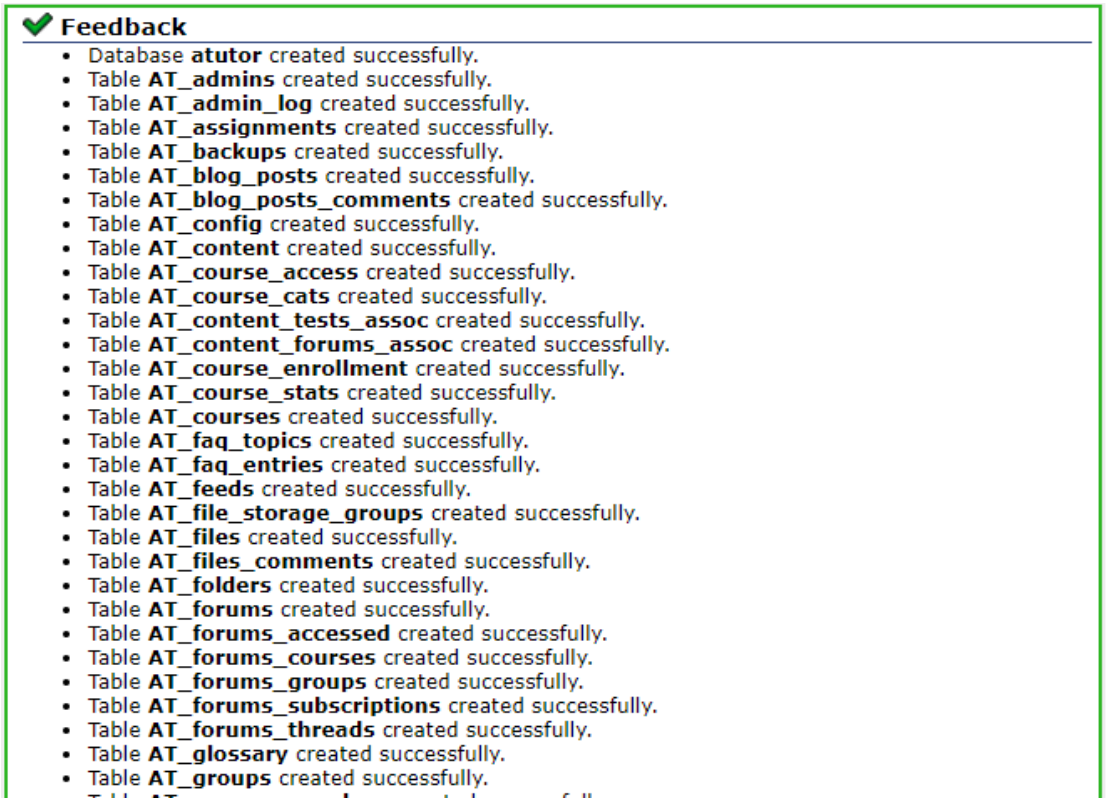

Рисунок 2.5 – Список створених таблиць у базі даних

Натиснувши кнопку “Next” ми попадемо на сторінку створення користувача-адміністратора, інструктора та деякі додаткові системні налаштування в системі ATutor. Форма створення адміністратора зображена на рисунку 2.6, і є стандартною, адже містить лише три необхідних поля – ім’я користувача, пароль, а також електронний адрес користувача.

Форма реєстрації інструктора містить теж саме, з двома додатковими полями – ім’я користувача та його прізвище.

Додаткові налаштування системи містять поля для назви сайту системи, контактну електронну адресу(для зв’язку у випадку різних типів проблем), необов’язкове поле для встановлення адреси базової сторінки та опція для відображення сайту лише в якості соціальної платформи.

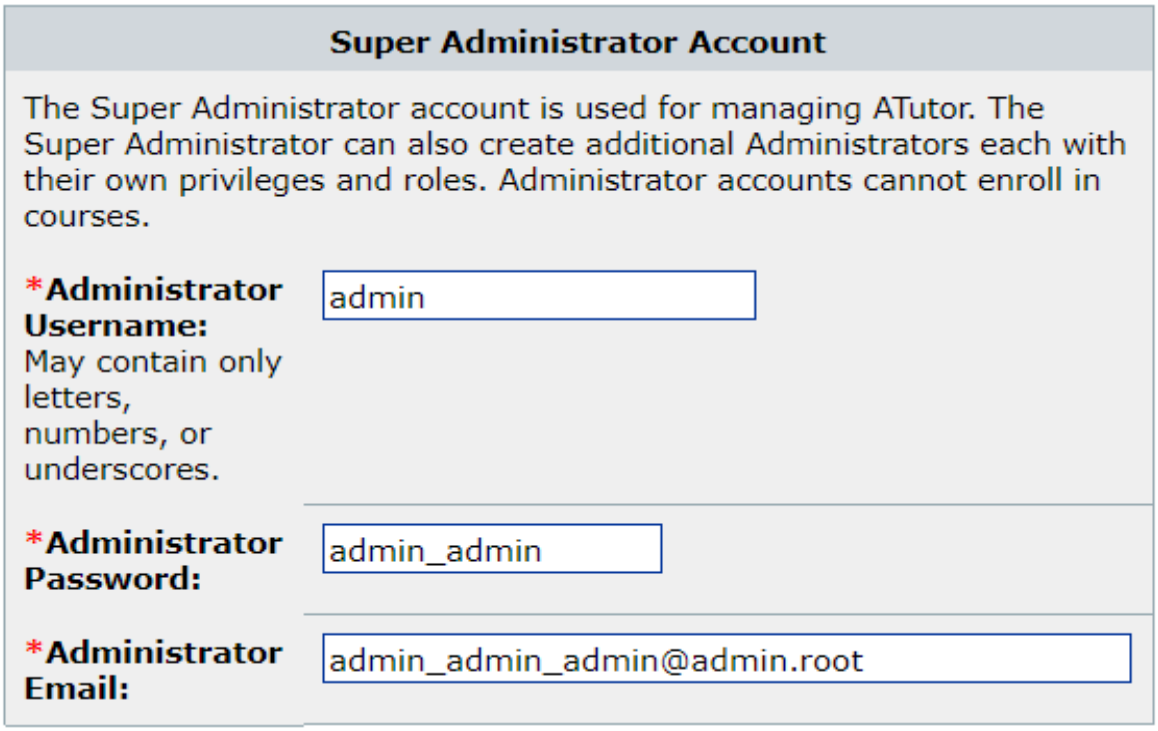

Рисунок 2.6 – Форма реєстрації адміністратора системи

Наступною дуже важливою сторінкою для конфігурації ATutor є сторінка, на якій вказується директорія, в якій розміщуватиметься контент – медіа файли, завантажені інструкторами та студентами у систему. Сторінка й правильне значення шляху продемонстровані на рисунку 2.7.

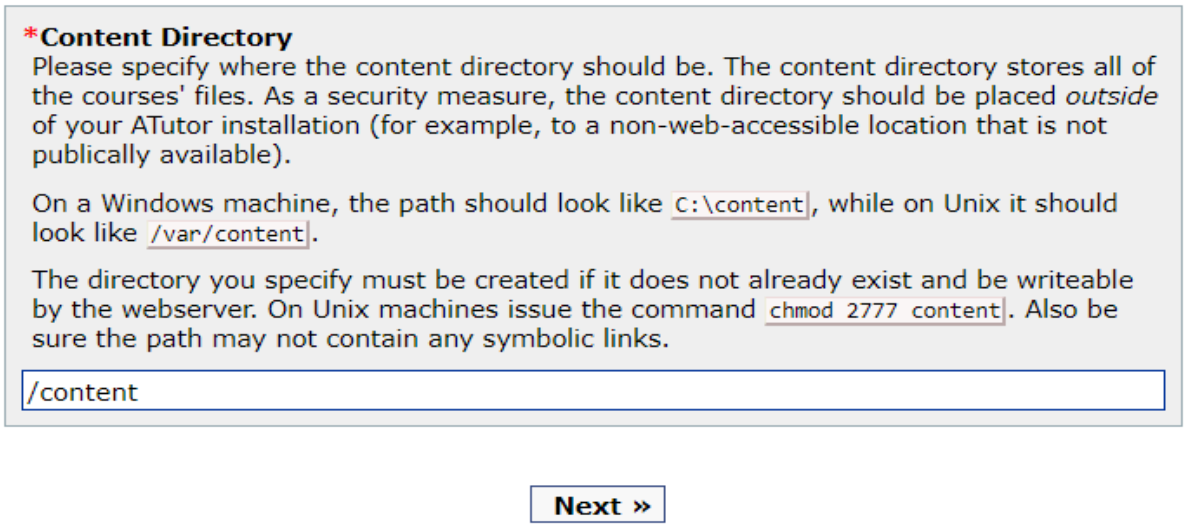

Рисунок 2.7 – Форма вказання директорії для контенту системи

Наступні кроки носять інформаційний характер та потребують лише натискання кнопки “Next”. Останній крок відобразить вікно з повідомленням про успішне налаштування системи та запропонує залогінуватися у неї. Це вікно зображене на рисунку 2.8.

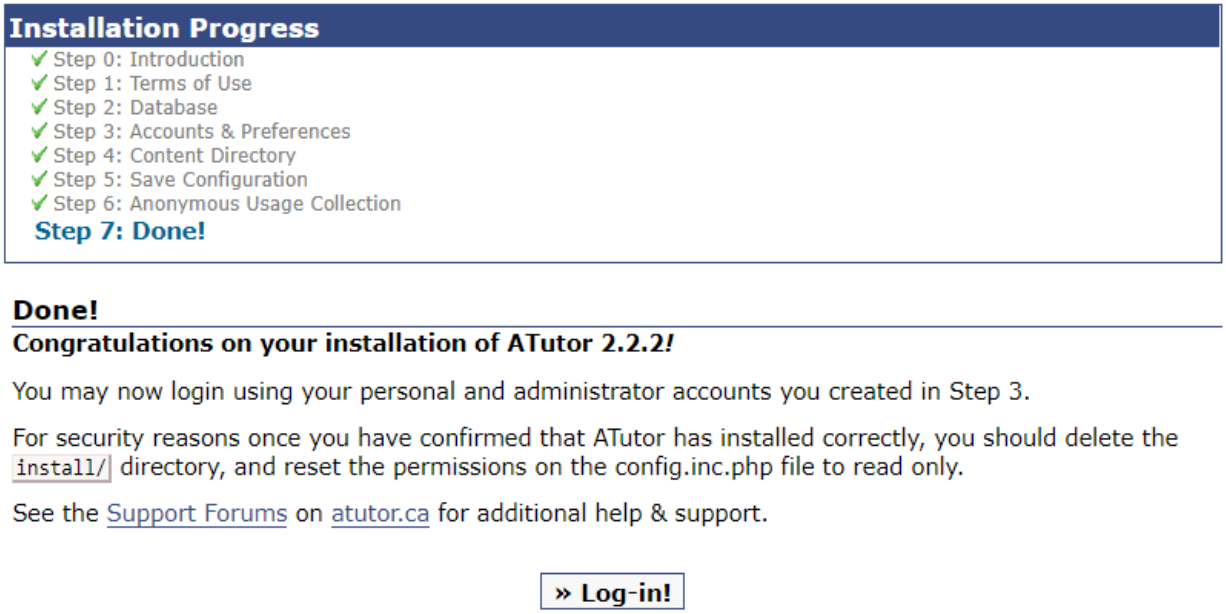

Рисунок 2.8 – Екран успішного завершення інсталяції системи ATutor

Натискання на “Log-in” направить користувача на сторінку авторизації у систему ATutor, щоб увійти в систему, можна скористатися раніше створеними користувачами.

2.2 Сценарії тестування системи

Після встановлення ATutor, важливо перевірити систему на предмет функціонування базового функціоналу. Для цього варто виконати димне тестування, яе буде включати наступні сценарії:

- Авторизування з валідними та невалідними даними;
- Реєстрація з валідними та невалідними даними;
- Перевірка працездатності сторінок Home, Photo Gallery, Users, Courses, Modules, System Preferences для адміністратора;
- Перевірка сторінок My Courses, Browse Courses, Profile, Preferences, Networking, Calendar для інструктора й студента;
- Створення нового курсу та додавання студентів до новоствореного курсу, а також перевірка працездатності запису на курс студентом.

У таблиці 2.1 описано тестовий випадок логінування адміністратора.

Таблиця 2.1 –Логінування адміністратора

Назва тестового скріпта: Логінування адміністратора	
Опис тестового скріпта: Переконатись, що при введені правильних реквізитів адміністратор буде авторизований у систему	
Дії	Очікуваний результат
1. Вводимо логін адміністратора	–
2. Вводимо пароль адміністратора	–
3. Натискаємо кнопку Login	Користувач ввійшов в систему в якості адміністратора

У таблиці 2.2 описано тестовий випадок логінування інструктора.

Таблиця 2.2 – Логінування інструктора

Назва тестового скріпта: Логінування інструктора	
Опис тестового скріпта: Переконатись, що при введені правильних реквізитів інструктор буде авторизований у систему	
Дії	Очікуваний результат
1. Вводимо логін інструктора	–

Продовження таблиці 2.2

1	2
2. Вводимо пароль інструктора	–
3. Натискаємо на кнопку Login	Користувач ввійшов в систему в якості інструктора

У таблиці 2.3 описано тестовий випадок реєстрування нового інструктора.

Таблиця 2.3 –Тестовий випадок реєстрації нового інструктора

Назва тестового скріпта: Реєстрація нового інструктора	
Опис тестового скріпта: Переконатись, що система дає змогу створити нового інструктора з коректними даними	
Дії	Очікуваний результат
1. Натискаємо на кнопку Register	Відобразиться форму реєстрації нового користувача

2. Вводимо коректний логін для нового користувача (user1)	–
3. Вводимо коректний пароль для нового користувача (^&*tyuGhj)	–
4. Ввводимо ще раз коректний пароль для нового користувача (^&*tyuGhj)	–
5. Вводимо коректну електронну адресу нового користувача (qwerty@qwerty.ua)	–
6. Вводимо ще раз коректну електронну адресу нового користувача (qwerty@qwerty.ua)	–
7. Вводимо коректне значення імені нового користувача (Sasha)	–

Продовження таблиці 2.3

1	2
8. Вводимо коректне значення прізвища нового користувача (Petruk)	–
9. Натискаємо на кнопку Save	Відбудеться реєстрація нового інструктора. Його автоматично залогінить в систему

У таблиці 2.4 описано тестовий випадок створення для курсу нової категорії.

Таблиця 2.4 –Тестовий випадок створення нової категорії для курсу

Назва тестового скріпта: Створення нової категорії для курсу	
Опис тестового скріпта: Переконатись, що система дає змогу створити для курсу, нову категорію із коректними даними	
Передумови: Користувач залогінувався у систему у ролі адміністратора	
Дії	Очікуваний результат
1. Натискаємо на опцію меню Courses	Відобразиться форма з списком наявних в системі курсів
2. Натискаємо на опцію Categories	Відобразиться форма наявних у системі категорій

3. Натискаємо на опцію Create Category	Відобразиться форма для створення нової категорії
4. Вводимо коректну назву для нової категорії (Category1)	–
5. Натискаємо на кнопку Save	Буде створено нову категорію й відображено форму усіх категорій. Новостворена категорія мусить знаходитись у цій формі

У таблиці 2.5 описано тестовий випадок створення нового курсу у системі

.

Таблиця 2.5 – Тестовий випадок створення нового курсу у системі

Назва тестового скріпта: Створення нового курсу у системі	
Опис тестового скріпта: Переконатись, що система дає змогу створити новий курс з коректними даними	
Передумови: 1. Користувач залогінувався у систему у ролі інструктора 2. У системі наявна хоча б одна категорія	
Дії	Очікуваний результат
1. Натискаємо на кнопку My Courses	Відобразиться форма курсів для авторизованого інструктора
2. Натискаємо на кнопку підменю Create Course	Відобразиться форма для створення нового курсу
3. Вводимо коректну назву для нового курсу (Course1)	–
4. Обираємо категорію для створюваного курсу (Category1)	–
5. Натискаємо на кнопку Save	Новий курс буде створено, та буде відображено початкову сторінку новоствореного курсу

У таблиці 2.6 описано тестовий випадок реєстрування нового студента у системі.

Таблиця 2.6 – Тестовий випадок реєстрації нового студента у системі

Назва тестового скріпта: Реєстрація нового студента у системі

Опис тестового скріпта: Переконатись, що система дає змогу створити нового студента з коректними даними
Передумови: Користувач залогінувався у систему у ролі адміністратора

Продовження таблиці 2.6

Дії	Очікуваний результат
1. Натискаємо на Users	Система відобразить форму наявних в системі користувачів
2. Натискаємо на Create User Account	Система відобразить форму реєстрації нового користувача
3. Вводимо логін для створюваного користувача (studentOne)	–
4. Вводимо пароль створюваного користувача (^&*tyuGhj)	–
5. Повторно вводимо пароль створюваного користувача (^&*tyuGhj)	–
6. Вводимо електронний адрес для створюваного користувача (studentOne@studentOne.ua)	–
7. Повторно вводимо електронний адрес створюваного користувача (studentOne@studentOne.ua)	–
8. Вводимо значення імені створюваного користувача (Dante)	–
9. Вводимо значення прізвища створюваного користувача (Sparda)	–
10. Встановоюємо статус для облікового запису “Студент”	–
11. Натискаємо на кнопку Save	Буде збережено нового студента у системі й відображено форму із усіма наявними користувачами, і у якій буде знаходитись тільки що створений Студент

У таблиці 2.7 описано тестовий випадок запису студентом на існуючий навчальний курс.

Таблиця 2.7 – Тестовий випадок запису на навчальний курс студентом

Назва тестового скріпта: Запис на навчальний курс студентом	
Опис тестового скріпта: Переконатись, що система дає змогу студенту записатись на навчальний курс	
Передумови: 1. Користувач залогінувався у системі у ролі Студент 2. У системі існує курс Course1	
Дії	Очікуваний результат
1. Натискаємо на Browse Courses	Відобразиться форма існуючих курсів у системі
2. Натискаємо на курс з захищеним доступом (Course1)	Відобразиться сторінка з цим курсом
3. Поруч з назвою курсу, натискаємо на посилання Enroll Me	Відобразиться форму для підтвердження запису на курс Course1
4. Натискаємо на кнопку Enroll Me	Студента буде зареєстровано на курс та відправить на сторінку його курсів, на якій має бути відображено курс, на який студент зареєструвався

У таблиці 2.8 описано тестовий випадок відписки від курсу студентом.

Таблиця 2.8 – Тестовий випадок відписки від курсу студентом

Назва тестового скріпта: Відписка від курсу студентом	
Опис тестового скріпта: Переконатись, що система дає змогу студенту виконати відписку від курсу на який він записався	

Продовження таблиці 2.8

Передумови: 1. Користувач залогінувався у систему у ролі Студент 2. Студент підписаний на якийсь з існуючих курсів	
Дії	Очікуваний результат
1. Натискаємо на My Courses	Відобразиться форма з курсами на які підписаний студент
2. Натискаємо на кнопку відписки (Unenroll) навпроти курсу (Course1)	Відобразиться сторінка з підтвердженням наміру користувача

3. Підтверджуємо дію, натиснувши на кнопку Yes	Студента буде відписаний від курсу й буде переадресований на сторінку його курсів (My Courses)

У таблиці 2.9 описано тестовий випадок створення нової категорії для тестових питань у курсі.

Таблиця 2.9 – Тестовий випадок створення категорії для питань тесту

Назва тестового скріпта: Створення нової категорії для питань тесту	
Опис тестового скріпта: Переконатись, що система дає змогу інструктору створити категорію для питань тесту	
Передумови: 1. Користувач залогінувався у системі у ролі Інструктор 2. Авторизований інструктор має створений начвальний курс 3. Авторизований інструктор перейшов на головну сторінку його курсу	
Дії	Очікуваний результат
1. Натискаємо на табу Manage	Відобразиться форма налаштувань для поточного курсу

Продовження таблиці 2.9

1	2
2. Натискаємо на пункт Tests and Surveys	Відобразиться форма з існуючими у курсі тестами
3. Натискаємо на підпункт Question Categories	Відобразиться список з усіма наявними категоріями тестових питань в обраному курсі
4. Натискаємо на ще один підпункт Create Category	Відобразиться форма для створення категорій питань
5. Введимо назву для нової категорії питань (CategoryForQuestions)	–
6. Натискаємо на кнопку Save	Нова категорія для тестових питань буде збережено та відобразиться повідомлення про збереження, після чого користувача переадресує на форму із доступними категоріями питань

У таблиці 2.10 описано тестовий випадок створювання нового тестового питання для курсу.

Таблиця 2.10 –Тестовий випадок створення нового тестового питання

Назва тестового скріпта: Створення нового тестового питання
Опис тестового скріпта: Переконатись, що система дає змогу інструктору створити нове тестове питання для курсу
Передумови: 1. Користувач залогінувався у систему в ролі Інструктор 2. У системі наявний курс, інструктором якого виступає залогінований користувач 3. У системі існує створена категорія для питань тесту 4. Авторизований інструктор перейшов на основну сторінку його курсу

Продовження таблиці 2.10

Дії	Очікуваний результат
1. Натискаємо опцію меню Manage	Відобразиться форма налаштувань для поточного курсу
2. Натискаємо на опцію налаштувань Tests and Surveys	Відобразиться форма з існуючими у курсі тестами
3. Натискаємо на підопцію меню Question Bank	Відобразиться список доступних тестових запитань для поточного курсу
4. У списку Create New Question обираємо з доступних опцій тип питання True or False й натискаємо на кнопку Create	Відобразиться форма створення нового запитання з типом Так/Ні
5. Обираємо раніше створену категорію тестового питання (CategoryForQuestions)	–
6. Вводимо тестове запитання (“Mario – крута гра?”)	–

7. Обираємо правильну відповідь (Yes)	–
8. Натискаємо на кнопку Save	Буде додано нове тестове запитання, й відображено відповідне повідомлення. Після чого користувач буде переадресований на сторінку із списком існуючих тестових запитань, в якому буде наявне щойно створене запитання

У таблиці 2.11 описано тестовий випадок створення анкети тестування.

Таблиця 2.11 – Тетовий випадок створення анкети тестування

Назва тестового скріпта: Створення анкети тестування	
Опис тестового скріпта: Переконатися, що система дає змогу інструктору створити нову анкету тестування	
Передумови: 1. Користувач залогінувався у системі у ролі Інструктор 2. У системі наявний курс, інструктором якого є залогінований користувач 3. Авторизований інструктор перейшов на основну сторінку його курсу	
Дії	Очікуваний результат
1. Натискаємо опцію меню Manage	Відобразиться форма налаштувань курсу
2. Натискаємо на опцію налаштувань Tests and Surveys	Відобразиться форма із існуючими у курсі тестами
3. Натискаємо підопцію меню Create Test/Survey	Відобразиться форма для створення нової анкети тестування
4. Вводимо валідну назву анкети (Test1). Решту полів не чіпаємо	–

5. Натискаємо на кнопку Save	Буде успішно створена нова тестова анкета, також показано відповідне повідомлення. Потім інструктора переадресує на сторінку з існуючими у курсі анкетами тестування

У таблиці 2.12 описано тестовий випадок додавання запитання до існуючої тестової анкети

Таблиця 2.12 – Тестовий випадок додавання питання до анкети тестування

Назва тестового скріпта: Додавання питання до анкети тестування	
Опис тестового скріпта: Переконатись, що система дає змогу інструктору додати наявне тестове запитання у інаявну анкету тестування	
Передумови: 1. Користувач залогінувався у системі у ролі Інструктор 2. У системі наявний курс, інструктором якого є залогінований користувач 3. На існуючому курсі є принаймні одне тестове запитання 4. На існуючому курсі є принаймні одна анкета для тестування 5. Авторизований інструктор перейшов на основну сторінку його курсу	
Дії	Очікуваний результат
1. Натискаємо опцію меню Manage	Відобразиться форма налаштувань для курсу
2. Натискаємо на опцію налаштувань Tests and Surveys	Відобразиться форма з існуючими у курсі анкетами тестування
3. У списку наявних анкет знаходимо потрібну (Test1) та вибираємо її, натиснувши на радіо кнопку з лівої сторони від назви анкети тестування	Тестова анкета відобразиться як обрана анкета
4. Натискаємо на кнопку Questions, яка знаходиться під списком наявних анкет тестування	Відобразиться список усіх тестових запитань в даній анкеті тестування
5. Натискаємо підопцію меню Add Questions	Відобразиться список усіх тестових запитань в поточному курсі

Продовження таблиці 2.12

1	2
6. Знаходимо потрібне запитання(“Mario – крута гра?”) у списку доступних запитань та обираємо його, натиснувши на прапорець зліва від назви запитання	Тестове запитання стане обраним
7. Натискаємо на кнопку Add to Test/Survey, вона знаходиться внизу, під списком наявних тестових запитань	Відобразиться форма, на якій нас запитає про підтвердження виконання попередньої дії
8. Натискаємо на кнопку Yes	Тестове запитання буде успішно додане до вказаної анкета, та видасть відповідне повідомлення. Після чого переадресує інструктора на сторінку з усіма запитаннями до поточної анкети тестування, серед яких має бути додане нами запитання
9. Вводимо кількість балів, яку студент получить за правильну відповідь на це запитання	–
10. Натискаємо на кнопку Save	Вартість питання буде змінено, про що нас сповістить відповідне повідомлення

У таблиці 2.13 описано тестовий випадок проходження анкети тестування студентом

Таблиця 2.13 – Тестовий випадок проходження анкети тестування студентом

Назва тестового скріпта: Проходження анкети тестування студентом
Опис тестового скріпта: Переконатися, що система дає змогу студенту пройти наявну анкету тестування

Продовження таблиці 2.13

Передумови: 1. Користувач залогінувався у систему у ролі Студент 2. Користувач є записаним на курс(Course1) 3. На курсі існує принаймні одна анкета тестування 4. В анкеті тестування існіє принаймні одне запитання 5. Студент перейшов на головну сторінку курсу	
Дії	Очікуваний результат
1. Натискаємо на опцію меню My Tests and Surveys, що розміщена на головній сторінці	Відобразиться список усіх доступних тестових анкет на курсі
2. Натискаємо на активну анкету тестування	Відобразиться інформаційна форма анкету
3. Натискаємо на кнопку Begin	Відобразиться форма з тестовими запитаннями, які є у анкеті
4. Обираємо будь-які варіанти відповіді	–
5. Натискаємо на кнопку Submit	Відобразиться результат проходження тесту, з інформацією про список питань, нотатки про усі дані неправильні та правильні відповіді. Також відобразиться результат проходження тесту з повідомленням, про успішність складення тесту.

2.3 Необхідні засоби для автоматизації тестів

2.3.1 Локатори веб-елементів

XPath (Language Path Language) – це мова, яка описує спосіб пошуку та обробки елементів у документах мови розширеної розмітки (XML) з

використанням синтаксису адресації на основі шляху через логічну структуру або ієрархію документа.

При цьому використовується абстракція інформації, визначена у наборі інформації XML (Infoset). Оскільки XPath не використовує синтаксис XML, він може використовуватися в інших контекстах, а не тільки у XML.

XPath використовує синтаксис, подібний до неофіційного набору вказівок для пошуку певного географічного положення[36].

Приклади XPath виразів описані у таблиці 2.14.

Таблиця 2.14 – Приклади використання XPath виразів

Вираз	Опис
shop	Виділяє всі наявні вузли з назвою “shop”
/shop	Виділяє корінний елемент shop
shop/products	Виділяє всі вузли products, які є дітьми вузла shop
//shop	Виділяє всі вузли shop незалежно від того, де вони знаходяться в документі
shop//products	Виділяє всі вузли products, які є предками вузла shop незалежно від того, де вони знаходяться під shop вузлом
//@fruit	Виділяє всі атрибути, які називаються fruit

2.3.2 Selenium WebDriver

Selenium – це набір API з відкритим кодом, який використовується для автоматизації тестування веб-програми.

Інструмент Selenium WebDriver використовується для автоматизації тестування веб-додатків, щоб перевірити, чи працює він, як очікувалося. Він підтримує безліч браузерів, таких як Firefox, Chrome, IE та Safari. Однак, використовуючи Selenium WebDriver, ми можемо автоматизувати тестування лише для веб-додатків. Він не підходить для віконних додатків. Він також підтримує різні мови програмування, такі як C#, Java, Perl, PHP та Ruby для написання тестових сценаріїв. Selenium Webdriver не залежить від платформи, оскільки один і той же код може використовуватися в різних операційних

системах, таких як Microsoft Windows, Apple OS та Linux. Це один із компонентів сімейства селену, який також включає ID ID Selenium, API клієнта Selenium, дистанційне керування Selenium та селенієву сітку[37].

Нижче наведено фрагмент коду, який дає змогу створити екземпляр Selenium веб-драйвер браузера Google Chrome:

```
var chromeDriver = new ChromeDriver("E:\\ chrome");
```

Для навігації у браузері використовуючи вебдрайвер, використовується наступна команда:

```
driver.Navigate().GoToUrl(“whatis.techtarget.com”);
```

Для знаходження потрібних елементів на веб-сторінці можна скористатись методом FindElement (або FindElements, якщо потрібно знайти декілька елементів), а також By, в якому ми задаємо спосіб пошуку елементу: через ідентифікатор елемента, по його імені, через XPath та Css локатори.

2.3.3 Фреймворк NUnit

NUnit – це платформа з відкритим кодом, розроблена для написання та запуску тестів на мовах програмування Microsoft .NET.

NUnit має схожий графічний інтерфейс користувача (GUI) з інтерфейсом в JUnit. Тести можна проводити постійно. Результати надаються негайно. Кілька тестів можна проводити одночасно. Ніяких суб'єктивних суджень людини або тлумачень результатів тесту не потрібно. Простота фреймворку дозволяє легко виправити помилки в міру їх виявлення[38].

NUnit використовує різноманітні атрибути задля впровадження своєї функціональності в тестові класи й методи. До основних атрибутів можна віднести наступні:

- TestFixture – Позначає клас, який містить тестові методи;

- Test – Позначає метод у тестовому класі. Цей метод перетворюється у тест і починає відображатись у тест провіднику;
- SetUp – Зазначає що метод помічений атрибутом виконається безпосередньо перед кожним тестом;
- TearDown – Зазначає що метод помічений цим атрибут буде виконуватись після кожного тестового методу;
- Repeat – Зазначає що метод має бути виконаний декілька разів
- Order – Зазначає що метод має бути виконаний у відповідному порядку

2.4 Створення програмного забезпечення для автоматизованого тестування

Створення програмного забезпечення для автоматизованого тестування відбувалося із використанням патерну PageObject, а також використовуючи інструмент з відкритим кодом Atata.

PageObject (з англійської «об'єкт сторінки») — патерн проектування, є найбільш широко використовуваною схемою дизайну спільнотою Selenium, на якій кожна веб-сторінка розглядається як інший клас. Для кожного класу цієї сторінки (тобто об’єктів сторінки) визначаються елементи цієї сторінки та конкретні методи для цієї сторінки. Кожен об’єкт сторінки представляє сторінку веб-сторінки чи програми. Це шар між тестовими сценаріями та інтерфейсом користувача який також інкапсулює функції сторінки[39].

Переваги PageObject підходу:

- Легкий в обслуговуванні. Після того, як ви створили дизайн, навіть будь-який локатор або функціональні зміни на веб-сторінці ви точно знаєте, де змінити свій код;
- Чистий і зрозумілий код. Замість того, щоб використовувати один і той же код у всіх тестових рамках, багато функцій визначаються в межах об’єктів сторінки. Отже, рядки написаних кодів скорочуються, і ваш код стає легше зрозуміти;

- Допомагає вказати інтерфейс користувача. Оскільки ви створюєте класи сторінок майже для кожної сторінки на своїй веб-сторінці, легко зрозуміти функціональність та модель вашої веб-сторінки навіть при погляді на тестові рамки.

Atata – це тестовий фреймворк написаний на мові програмування C # для автоматизації Web тестування на основі Selenium WebDriver. Atata фреймворк складається із:

- Компонентів;
- Атрибутів для пошуку веб-компонентів;
- Атрибутів налаштувань;
- Стратегій;
- Тригерів;
- Методів та атрибутів для верифікації.

Було спроектовано ось таку структуру для програми:

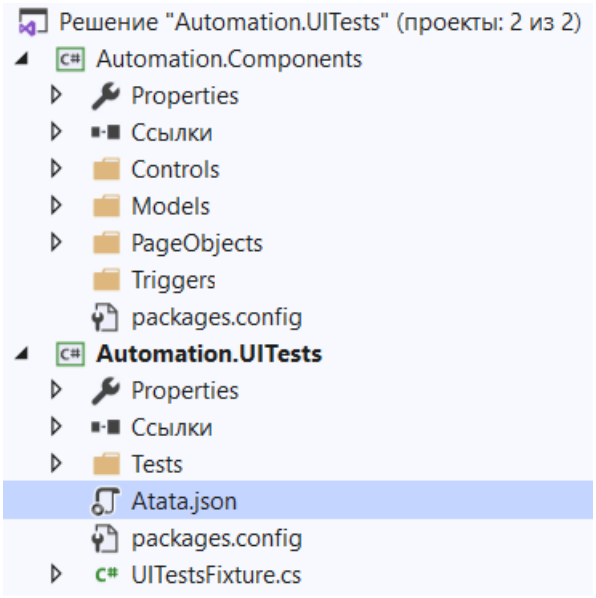

Рисунок 2.10 – Структура програмного рішення

Папка PageObjects потрібна для зберігання описаних сторінок веб-сайту. Її структура зображена на рисунку 2.11.

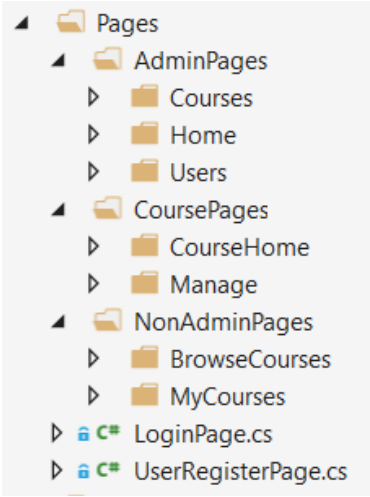

Рисунок 2.11 – Структура об’єктів сторінок веб-сайту

Зважаючи на те як побудований ATutor, моделі сторінок можна розділити на три основні категорії:

- Моделі сторінок адміністратора;
- Моделі сторінок інструктора та студента;
- Моделі сторінок курсу.

Спільними для всіх користувачів являються сторінки реєстрації та логінування тому їх також можна виділити у окремі файли.

Модель сторінки реєстрації побудована наступним чином:

```
public class UserRegisterPage : Page<_>
{
    public TextInput<_> LoginName { get; private set; }
    public PasswordInput<_> Password { get; private set; }
    public PasswordInput<_> PasswordAgain { get; private set; }
    [RandomizeStringSettings("{0}@mail.com")]
    public TextInput<_> EmailAddress { get; private set; }
    public TextInput<_> EmailAddressAgain { get; private set; }
    public TextInput<_> FirstName { get; private set; }
    public TextInput<_> LastName { get; private set; }
    [FindByXPath("div[text() = 'Account Status']")]
    public RadioButtonList<AccountStatus, _> AccountStatus { get; private set;
}

    [ScrollDown]
```

```
        public Button<MyCoursesPage, _> Save { get; private set; }
    }
```

Так як у Atata фреймворку усі об’єкти сторінок унаслідуються від класу Page/PageObject це також дозволяє використати Fluent підхід. Це підхід який представляє собою ланцюжок викликів, до самого закінчення виконання програми. Об’єктом поля введення слугує клас TextInput. PasswordInput є підвидом об’єкту TextInput, та використовується здебільшого для полів введення пароля. Клас Button потрібно користатись у тому випадку коли є потреба представити кнопку. За замовчуванням Atata виконує пошук елементів веб сторінка за назвою змінної та контенту веб елемента. Завдяки цьому, Button<UserRegisterPage, _> Register знайде кнопку з відповідно текстом Register.

Хорошою демонстрацією можливостей Atata є нижче описаний код, який являється сторінкою курсів студента та інструктора:

```
    public class MyCoursesPage : Page<_>
    {
        [FindByXPath("//div[@id='feedback']//ul")]
        public UnorderedList<ListItem<_>, _> Feedback { get; private set; }
        public NonAdminMainPageMenu<_> MainMenu { get; private set; }
        public NonAdminMyCoursesSubMenu<_> SubMenu { get; private set; }
        public CustomTable<CourseRow, _> CoursesTable { get; private set; }
        public class CourseRow : CourseTableRow<_>
        {
            [FindByColumnHeader]
            public Link<CourseHomePage, _> Course { get; private set; }
            public Text<_> Instructor { get; private set; }
            public Text<_> Status { get; private set; }
            [FindByContent]
            public LinkDelegate<CourseUnenrollConfirmationPage, _> Unenroll { get; private set; }
        }
    }
```

Можна виділити FindByContent атрибут, який за пошук елемента веб сторінки по контенту підставляючи ім’я змінної. Також ще одним зручним інструментом для зіставлення значень таблиці та її заголовків являється FindByColumnHeader. Завдяки цьому атрибути береться імя змінної та вираховується її позиція у шапці таблиці, після чого цей індекс підставляється в потрібний рядок. Більше коду програмного забезпечення можна знайти у Додатку Г.

Також в ході виконання завдання, було спроектувано та створено автоматизовані тестові сценарії Smoke тестування, які є детально описані у розділі 2.2.

Створення тестових скриптів відбувалося за допомоги нижчевказаних інструментів:

- Описані тестові сценарії;
- Фреймворк Atata;
- Інструмент для автоматизації дій браузера Selenium WebDriver;
- Фреймворк для роботи з юніт тестами NUnit;
- Моделі сторінок, які використовуються у скриптах та виконують основні функції взаємодії із системою ATutor.

Для зменшення об’єму написаного однотипного коду було вирішено оптимізувати процес логінізації різними користувачами, тому доцільним рішенням стало використання атрибуту SetUp який виконується перед кожним запуском тесту. Завдяки цьому в метод помічений атрибутом SetUp можна винести логіку однакову для кожного тестового скріпту. У вказаному нище коді описано процес конфігурування AtataContext:

```
[SetUp]
public void SetUp()
{
   AtataContext.Configure().
      ApplyJsonConfig<AppConfig>().
      Build();
}
```

Нижче описано код декілької методів які є у вжитку тестовими скріптами та виконують логінування різних типів користувачів використовуючи змодельовані об’єкти сторінок:

```
public AdminMainPage LoginAsAdmin()
    {
       return Go.To<LoginPage>().
          LoginNameOrEmail.Set(AppConfig.Current.AdminAccount.Login).
          Password.Set(AppConfig.Current.AdminAccount.Password).
          Login.ClickAndGo<AdminMainPage>();
    }

    public MyCoursesPage LoginAsDefaultInstructor()
    {
       return LoginAsSimpleUser(
                AppConfig.Current.InstructorAccount.Login,
                AppConfig.Current.InstructorAccount.Password
       );
    }

    public MyCoursesPage LoginAsSimpleUser(string login, string password)
    {
       return Go.To<LoginPage>().
          LoginNameOrEmail.Set(login).
          Password.Set(password).
          Login.ClickAndGo<MyCoursesPage>();
    }
```

Завдяки цим методам логінізації, тепер декілька типів користувачів мають змогу авторизуватись у системі не сильно збільшуючи обсяг самого автоматизованого скріпта.

Для спрощення роботи з різними сталими інформаційними значеннями такими як логін, пароль було прийняте рішення винести це у окремий файл конфігурації, а саме JSON файл. Уривок з цього файлу наведено нижче:

```
{
 "adminAccount": {
  "login": "admin",
  "password": "admin_admin"
 },
 "instructorAccount": {
  "login": "instructor",
  "password": "instructor"
 },
 "defaultPassword": " ^&*tyuGhj"
```

Частина автоматизованих тестових випадків та їх код, які були написані у ході виконання роботи наведені нижче, а решта описана у Додатку Д.

Автоматизований тестовий скрипт який відповідає тестовому сценарію, описаному у таблиці 2.1:

```
[Test, Order(1)]
public void AdminLogIn()
{
   LoginAsAdmin().
      Feedback.Items.
      Contents.Should.Contain(SuccessfulLogInMessage);
}
```

Автоматизований тестовий скрипт який відповідає тестовому сценарію, описаному у таблиці 2.2:

```
[Test, Order(2)]
public void InstructorLogIn ()
{
   LoginAsDefaultInstructor().
```

```
        Feedback.
        Items.Contents.Should.Contain(SuccessfulLogInMessage);
    }
```

Автоматизований тестовий скрипт який відповідає тестовому сценарію, описаному у таблиці 2.3:

```
    [Test, Order(3)]
    public void RegisterNewUser()
    {
       Go.To<LoginPage>().
          Register.ClickAndGo().
             LoginName.SetRandom().
             Password.Set(AppConfig.Current.DefaultPassword).
             PasswordAgain.Set(AppConfig.Current.DefaultPassword).
             EmailAddress.SetRandom(out string email).
             EmailAddressAgain.Set(email).
             FirstName.SetRandom().
             LastName.SetRandom().
          Save.ClickAndGo().Feedback.Items.Contents.Should.
          Contain(SuccessfulRegistrationMessage);
    }
```

Метод Go.To використовується для умовної навігації між моделями сторінок, описаних кодом, що дозволяє закінчивши роботу з компонентами на одній сторінці перейти на іншу(наприклад після натискання на кнопку, виконати перехід на іншу сторінку). Також для безпосереднього переходу на іншу сторінку зазвичай використовується метод ClickAndGo() який виконує клік по контролі на веб сторінці.

Для встановлення значень здебільшого використовується Set, який також є віртуальним, тому дозволяє імплементувати свою логіку встановлення значення тому чи іншому елементу веб-сторінки.

Для перевірки поведінки веб сторінки доступний метод Should, який дає змогу виконати ряд верифікацій. Наприклад підметод Contain() перевіряє чи текстове значення компоненту містить в собі вказану стрічку.

У разі падіння тестового скріпта Atata дозволяє додати скріншот екрану на момент падіння тесту та зберегти його у окремій папці. Також Atata виконує логування кожної дії виконаної на веб сторінці що дозволяє детально аналізувати результат перебігу тесту.

3. ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНА ЧАСТИНА

Вартість програмного забезпечення включає в себе витрати на:

- основу та додаткову заробітні плати;
- оплату електроенергії;
- амортизаційні витрати;
- накладні витрати;
- матеріальні витрати.

Проведемо їх розрахунок, розпочавши із заробітної плати. Оподаткування заробітної плати програмістів підприємства-розробника, яким встановлено певний посадовий оклад, відбувається аналогічно оподаткуванню зарплати працівників інших галузей господарства. Тобто витрати на оплату праці в податковому обліку включаються до валових витрат підприємства [40], а в бухгалтерському обліку – до виробничої собівартості продукції (п. 11 П(С)БО 16), оскільки можуть бути віднесені до конкретного об'єкта витрат та включаються до прямих витрат на оплату праці [41].

У проекті будуть задіяні наступні людські ресурси:

- Керівник проекту;
- Розробник тестового фреймворку;
- Розробник API тестів;
- Розробник тестів графічного інтерфейсу.

Вважатимемо, що вся команда розробників працює у штаті підприємства-розробника і їм встановлено певний посадовий оклад. Місячний оклад, денна заробітна плата, трудомісткість і основна заробітна плата кожного учасника техпроцесу представлено у таблиці 3.1.

Таблиця 3.1 – Економічні показники учасників техпроцесу

	Місячна з.п., грн.	Денна з. п., грн	Трудомісткість, людино-дні		Основна заробітна плата, грн.	
			Процедурний підхід	Об’єктно-орієнтований підхід	Процедурний підхід	Об’єктно-орієнтований підхід

Керівник проекту	16275	775	4	4	3100	3100
Розробник тестового фреймворку	13650	650	10	13	6500	8450
Розробник API тестів	11550	550	7	7	3850	3850
Розробник тестів графічного інтерфейсу	11550	550	7	7	3850	3850
Всього			28 днів	31 день	17300 грн.	19250 грн.

Визначимо витрати на основну заробітну плату. Вони складають суму заробітних плат за кожну із робіт. Витрати на основну зарплату для процедурного та об’єктно-орієнтованого підходів:

$ЗП_{осн(п)} = 3100 + 6500 + 3850 + 3850 = 17300$ (грн.);

$ЗП_{осн(о)} = 3100 + 8450 + 3850 + 3850 = 19250$ (грн.);

Додаткова заробітна плата обчислюється за формулою:

$$ЗП_{дод} = ЗП_{осн} \cdot 0.2 \quad (3.1)$$

Обчислимо витрати на додаткову зарплату для двох підходів:

$ЗП_{дод(п)} = 17300 \cdot 0.2 = 3460$ (грн.);

$ЗП_{дод(о)} = 19250 \cdot 0.2 = 3850$ (грн.);

Визначимо загальний фонд заробітної плати, який рівний сумі затрат на основну та додаткову зарплати:

$ФЗП_{п} = 17300 + 3460 = 20760$ (грн.);

$ФЗП_{о} = 19250 + 3850 = 23100$ (грн.);

Крім того, слід визначити відрахування на соціальні заходи:

- єдиний соціальний внесок – 3,6 %;
- військовий збір – 1,5 %;
- ПДФО (прибутковий податок) – 15 %.

Отже, сума відрахувань на соціальні заходи буде становити:

$Відр_{ЄСВп} = 0.036 \cdot ФЗП_{п} = 0.036 \cdot 20760 = 747.36$ грн.;

$Відр_{ЄСВо} = 0.036 \cdot ФЗП_{о} = 0.036 \cdot 23100 = 831.6$ грн.;

$Відр_{ВЗп} = 0.015 \cdot ФЗП_п = 0.015 \cdot 20760 = 311.4$ грн.;

$Відр_{ВЗо} = 0.015 \cdot ФЗП_о = 0.015 \cdot 23100 = 346.5$ грн.

$Відр_{ПДФОп} = 0.15 \cdot ФЗП_п = 0.15 \cdot 20760 = 3114$ грн.;

$Відр_{ПДФОп} = 0.15 \cdot ФЗП_о = 0.15 \cdot 23100 = 3465$ грн..

Щоб визначити повні витрати на заробітну плату також до загального фонду необхідно додати суму відрахувань до фонду оплати праці. Для підприємств, які займаються виданням програмного забезпечення ці відрахування встановлені у розмірі 36,77% від фонду заробітної плати.

Отже загальні витрати на заробітну плату становитимуть:

$В_{зп(п)} = ФЗП_п + ФЗП_п \cdot 0.3677 = 20760 + 20760 \cdot 0.3677 = 28393.45$ (грн.)

$В_{зп(о)} = ФЗП_о + ФЗП_о \cdot 0.3677 = 23100 + 23100 \cdot 0.3677 = 31593.87$ (грн.)

Ще однією складовою витрат на розробку програмного забезпечення є матеріальні витрати. Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни: $M_{Bi} = q_i \cdot p_i$, де q_i – кількість витраченого матеріалу і-го виду; p_i– ціна матеріалу і-го виду.

Звідси, загальні матеріальні витрати можна визначити за формулою:

$$З_{м.в.} = \Sigma M_{B\,i} \quad (3.2)$$

Проведені розрахунки подано в таблиці 3.2.

Таблиця 3.2 – Розрахунок матеріальних витрат

№ п/п	Найменування матеріальних ресурсів	Од. виміру	Фактично витрачено матеріалів	Ціна 1-ці, грн.	Загальна сума витрат, грн.
1	SCRUM-дошка	шт	1	449,00	449,00
2	Маркери	шт	4	15,00	60,00
3	Губка для дошки	шт	1	40,00	40,00
4	Скрам картки	шт	10	10,00	100,00
Всього					649,00

Отже, загальна сума матеріальних витрат становить 649 гривень.

Розрахуємо витрати на оплату електроенергії. Витрати на електроенергію одиниці обладнання визначаються за формулою:

$$З_с = W \cdot T \cdot S \quad (3.3)$$

де W – необхідна потужність, кВт; Т – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії. Вартість одиниці електроенергії визначаємо для юридичної особи, яка є непромисловим підприємством. Згідно постанов Національної комісії, що здійснює державне регулювання у сфері енергетики та комунальних послуг (НКРЕКП), на з 1 грудня 2019 року ця сума становить 2.5 грн (ВАТ «Тернопільобленерго»).

$З_{е(п)} = 0.5 \cdot 28 \cdot 8 \cdot 2.5 = 280$ (грн.)

$З_{е(о)} = 0.5 \cdot 31 \cdot 8 \cdot 2.5 = 330$ (грн.)

Ще однією частиною вартості ПЗ є амортизаційні відрахування.

Комп’ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 % (квартальна – 15 %).

Комп'ютерні програми як інструментальні засоби створення ПЗ – компілятори, інтегровані середовища розробки, системи перевірки програмного коду тощо і вартість яких перевищує 1000 грн – включаються до 4 групи основних фондів (пп. 8.2.2 Закону про прибуток) і амортизуються за річною нормою 60% (або іншою, що не перевищує цієї норми).

Амортизація за час (період) використання обладнання (комп’ютера) складає долю затрат, які припадають на дослідницьку роботу (написання програми), і визначається:

$$А = \frac{С_Б \quad N_А \quad Т_{ФАК}}{Т_{ГОД}}, \quad (3.4)$$

де $С_Б$– балансова вартість обладнання, грн; $N_А$– норма амортизаційних відрахувань в рік, %; $Т_{год}$ – річний робочий фонд часу, год; $Т_{ФАК}$ – фактичний час роботи обладнання по написанню програми, год.

Таблиця 3.3 – Перелік обладнання, необхідного для розробки

Найменування	Кількість, шт.	Ціна за одиницю продукту, грн.	Сума, грн.

Ноутбук HP 250 G6	3	11999	35997
Операційна система Windows 10	1	2066	2066
Всього (вартістю більше 1000 грн.)			38063
Маніпулятор Real-El RM-213 USB	3	75	225
Всього (вартістю менше 1000 грн.)			225
Всього			38288

$$A_{П} = \frac{38063 \quad 0{,}6 \quad 28 \quad 8}{2016} = 2537{,}53 \text{ грн}; \; A_{о} = \frac{38288 \quad 0{,}6 \quad 31 \quad 8}{2016} = 2826{,}01 \text{ грн}$$

Накладні витрати – це витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці.

Залежно від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$НВ = 0{,}4 \cdot ЗП_{осн}$.

$НВ_{п} = 0{,}4 \cdot 17300 = 6920$ грн.;

$НВ_{о} = 0{,}4 \cdot 19250 = 7700$ грн.

Проведемо розрахунок собівартості програмного продукту.

Собівартість продукції – це сума грошових витрат підприємства (фірми) на виробництво і збут одиниці продукції, виконання робіт та надання послуг.

Повна собівартість програмного продукту дорівнює сумі усіх витрат на його виробництво:

$Св = В_{зп} + З_{мв} + З_{е} + А + НВ$;

$Св_{п} = 28393.45 + 649 + 280 + 2537.53 + 6920 = 38824.78$ (грн.)

$Св_{о} = 31593.87 + 649 + 330 + 2826.01 + 7700 = 43128.48$ (грн.)

Приймемо прибуток на рівні 30%. Для нових інноваційних продуктів, що користуються високим попитом на ринку, ринкову вартість B_p можна встановити вищу.

Отже, вартість розробленого програмного забезпечення:

$В_п = Св_п + 0.3 \cdot Св_п = 38824.78 + 0.3 \cdot 38824.78 = 50472.28$ (грн.)

$В_о = Св_о + 0.3 \cdot Св_о = 43128.48 + 0.3 \cdot 43128.48 = 56067.02$ (грн.)

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність (E) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E = \frac{П}{Св} \quad (3.5)$$

де П – прибуток, П= В - Св; Св – собівартість.

Якщо ринкова вартість програмного продукту рівна прийнятій, то економічна ефективність визначається встановленим рівнем прибутку. Отже,

$Е_п = (50438.567 – 38824.782) / 38824.782 = 0,3;$

$Е_о = (56029.824 – 43128.48) / 43128.48 = 0,3.$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($T_{ок}$):

$$T_{ок} = \frac{1}{E}$$

(3.6)

У нашому випадку $Т_{ок(п)} = Т_{ок(о)} = 1/0,3 = 3,33$ роки, що є нормальним, оскільки допустимим вважається термін окупності до 5 років.

Даний розрахунок виконаний у розрахунку на 1 екземпляр програмного продукту без врахування його тиражування.

Крім розробки не менш важливим етапом життєвого циклу є супровід і модернізація програмного забезпечення. За різними оцінками фахівців витрати на підтримку і модернізацію програмного забезпечення, написаного процедурним методом, становлять більше 50% витрат на його створення, а інколи і перевищують доходи від його реалізації. Об’єктно-орієнтоване представлення програми дозволяє навіть середньому програмісту швидко і ефективно

супроводжувати і модернізувати програми довжиною до декількох десятків тисяч рядків, що значно скорочує подальші витрати на супровід і модернізацію.

Виходячи із експертних оцінок і складності програми, приймемо величину витрат на супровід і модернізацію програмного забезпечення, створеного за процедурним методом 60% від початкових витрат, а за об'єктно-орієнтованим – 20%.

Собівартість модернізації:

$СвМ_п = 0.6 \cdot Св_п = 0.6 \cdot 38824.78 = 23294.87$ грн.,

$СвМ_о = 0.2 \cdot Св_о = 0.2 \cdot 43128.48 = 8625.70$ грн.

Для споживача вартість модернізації:

$М_п = 0.6 \cdot В_п = 0.6 \cdot 50472.22 = 30283.33$ грн.,

$М_о = 0.2 \cdot В_о = 0.2 \cdot 56067.02 = 11213.40$ грн.

Таким чином, уже після першої модернізації, загальні витрати на створення і супровід ПЗ для виробника за об'єктно-орієнтованим методом менші, ніж за процедурним, навіть якщо його собівартість є дещо дорожчою.

$ЗВ_{п\ (вир)} = 38824.78 + 23294.87 = 62119.65$ грн.;

$ЗВ_{о\ (вир)} = 43128.48 + 8625.70 = 51754.18$ грн.

Як і для споживача:

$ЗВ_п = 50472.22 + 30283.33 = 80755.55$ грн.;

$ЗВ_о = 56067.02 + 11213.40 = 67280.42$ грн.

Чистий приведений дохід (ЧПД) визначається як різниця між сукупними доходами (сукупний грошовий потік) і сукупними витратами (сукупними інвестиціями) взятими за весь період життя інвестицій і дисконтова ними в кожному році на фактор часу. Дисконтування являє собою визначення вартості майбутніх грошових потоків у теперішній момент часу. Ефективним вважається той проект, який забезпечує максимум ЧПД, оскільки при цьому досягається найвища дохідність власників інвестицій.

Визначення ЧПД відбувається за формулою:

$$ЧПД = \sum_{i=1}^{t} ГП_i \alpha_{TBi} \quad \sum_{i=1}^{t} ІК_i \alpha_{TBi}; \qquad (3.7)$$

де $ГП_i$ – грошовий потік і-го розрахункового року;

$ІК_i$ – сума інвестицій і-го розрахункового року;

α_{TBi} – коефіцієнт дисконтування (коефіцієнт приведення інвестицій і грошового потоку до теперішньої вартості).

Грошовий потік – це фінансовий показник, що характеризує ефект інвестицій у вигляді грошових коштів, які повертаються інвестору.

Коефіцієнт дисконтування показує, яку величину грошових коштів ми отримаємо з урахуванням фактору часу та ризиків.

Для розрахунку коефіцієнта дисконтування (коефіцієнта приведення) грошових потоків за роками періоду економічного життя інвестицій використовується формула:

$$\alpha = \frac{1}{(1+i)^n}; \qquad (3.8)$$

де і – ставка дисконтування або норма дисконту, і = 0,2;

n – час або кількість періодів (років), протягом якого планується отримання доходу.

$$\alpha_0 = 1, \ \alpha_1 = \frac{1}{(1+0{,}2)} = 0{,}83$$

Вважатимемо, що обидва програмних продукти однаково забезпечують потреби і вимоги споживача, і тому придбання першої чи другої програми однаково вплинуть на розмір його додаткових доходів на вкладений капітал. Тому приймемо цю величину за постійну, а порівняння дохідності двох проектів проведемо тільки за витратами.

$ЧПД_1$ = ГП + 0.83 · ГП - 50472.22 - 0.83 · 30283.33 = 1.83 ГП - 75607.38 грн

$ЧПД_2$ = ГП + 0.83 · ГП - 56067.02 - 0.83 · 11213.40 = 1.83 ГП - 65374.14 грн

Чим менші витрати, тим більша дохідність проекту.

Усі проведені розрахунки подано в таблиці 3.4

Таблиця 3.4 – Розрахунок вартості ПЗ за процедурним і об'єктним підходами

№ п.п.	Назва	Процедурний підхід	Об'єктно-орієнтований підхід
1	2	3	4
1	Зарплата основна, грн	17300	19250
2	Зарплата додаткова, грн	3460	3850
3	Фонд заробітної плати, грн	20760	23100
4	Відрахування на ФОП, грн	7633.45	8493.87
5	Разом на оплату праці, грн	28393.45	31593.87
6	Матеріальні витрати, грн	649	649
7	Електроенергія, грн	280	330
8	Амортизація, грн	2537.53	2826.01
9	Накладні витрати, грн	6920	7700
10	Разом на ін. витрати (6+7+8+9)	10431.33	11534.61
11	Собівартість, грн	38824.78	43128.48
12	Прибуток, грн.	11613.79	12901.34
13	Вартість розробленого ПЗ, грн	50472.22	56067.02
14	Економічна ефективність	0,3	0,3

Продовження таблиці 3.4

1	2	3	4
15	Термін окупності, років	3,33	3,33
16	Собівартість модернізації, грн	23294.87	8625.70
17	Супровід і модернізація, грн	30283.33	11213.40
18	Загальні витрати на розробку, грн.	62119.65	51754.18
19	Порівняльна економія витрат (для виробника), грн	-	10365.47
20	Загальні витрати (для споживача, на придбання програмного прод.) , грн	80755.55	67280.42
21	Порівняльна економія витрат (для споживача) , грн	-	13475.13

22	Дохідність проекту для споживача, грн.	-75607.38	-65374.14
23	Економія, грн.	-	10233.24

Економія витрат у випадку придбання, супроводу і одноразової модернізації програмного продукту, створеного за об’єктно-орієнтованим підходом, становить 10233.24 грн. Отже, сучасну комп’ютеру програму доцільно проектувати та розробляти по об’єктно орієнтованій парадигмі.

Загальна вартість пропонованих робіт по розробці програмного продукту становить 38824.78 грн. для першого варіанту та 43128.48 грн. для другого. Оскільки ефективність для обидвох проектів відповідно до встановленого рівня прибутку становить 0,3, що є високим показником, то проводити дані роботи варто і вкладені кошти окупляться за 3,33 року.

Отже, програмний продукт може бути впроваджений та мати подальший розвиток, оскільки він є економічно вигідною за всіма основними техніко-економічними показниками.

4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Оскільки розробка програмного забезпечення для тестування Інтернет ресурсу відбувалась на комп'ютері, потрібно розглянути основні нормативно-правові документи та відповідні особливості охорони праці.

Перелік нормативно-правових актів, що так чи інакше регулюють дане питання, є досить широким. Обов'язки роботодавця щодо забезпечення працівникам комфортних та безпечних умов для здійснення роботи, а також права працівників на такі умови передбачено частиною 2 ст. 2 та ч. 1 ст. 21 КЗпП, а також ст. 13 Закону України «Про охорону праці». Даний закон визначає основні положення щодо реалізації конституційного права працівників на охорону їх життя і здоров'я у процесі трудової діяльності, на належні, безпечні і здорові умови праці, регулює за участю відповідних органів державної влади відносини між роботодавцем і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок організації охорони праці в Україні. Більшість актів у даній сфері становлять акти підзаконного рівня, а саме, численні правила, інструкції, державні санітарні правила і норми (ДСанПІН) тощо, якими врегульовуються окремі моменти щодо власне конструкції електронно-обчислювальної техніки, особливостей облаштування приміщень для роботи з нею та низки інших подібних вимог[43].

На сьогодні до основних підзаконних актів у даній сфері можна віднести:

- Наказ Міністерства соціальної політики України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» від 14.02.2018 № 207;
- Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98, затверджені постановою Головного державного санітарного лікаря України від 10.12.1998 № 7;

- Примірну інструкцію з охорони праці під час експлуатації електронно-обчислювальних машин, затверджену наказом Міністерства доходів і зборів України від 05.09.2013 № 443.

Вимоги щодо розміщення і планування приміщень для роботи з комп'ютером: відповідні робочі місця заборонено облаштовувати у підвальних або цокольних приміщеннях будинків. В обладнанні приміщень забороняється використання полімерних матеріалів, що виділяють шкідливі хімічні речовини. Також слід приділити увагу забезпеченню достатнім для здійснення роботи рівнем освітлення (природного та штучного – у темну пору доби) та звукоізоляції. Для регуляції рівня освітлення природним світлом бажано застосовувати жалюзі. Окрім того, у приміщеннях, де здійснюється робота з комп'ютерами, щодня має здійснюватися вологе прибирання з метою недопущення запиленості підлоги та меблів.

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця операторів (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном), мають бути надійно захищені діелектричними щитками або сітками з метою недопущення потрапляння людини під напругу.

Особливої уваги заслуговують заходи дотримання протипожежної безпеки. Так, у всьому офісі лінії електромережі мають бути забезпечені від виникнення короткого замикання, а також від перепадів мережевої напруги, що може спричинити збої в роботі електронно-обчислювальної техніки. Приміщення (окрім тих, де розташовуються сервери) мають бути оснащені системою автоматичної пожежної сигналізації та вогнегасниками. Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, застосовувати негорючу ізоляцію. У приміщенні, де одночасно експлуатуються понад п'ять комп'ютерів, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

Вимоги щодо організації та обладнання робочих місць: площа, відведена на одне робоче місце має становити не менше 6 кв. м., а об'єм – не менше 20 куб. м.

Конструкція робочого місця повинна забезпечувати підтримання оптимальної робочої пози (тобто такої, яка дозволяє працівникові виконувати роботу з мінімальним напруженням тіла, і яка дозволяє уникнути перевтоми в ході і після закінчення робочого процесу)[43].

За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою 1,5 - 2 м.

Перед початком роботи нового працівника роботодавець згідно зі ст. 29 КЗпП зобов'язаний проінформувати його під розписку про умови праці, наявні на його робочому місці. У тому числі, про всі небезпечні чи шкідливі виробничі фактори, які ще не усунуто, та про можливі наслідки їх впливу на здоров'я працівника, а також про можливі пільги та компенсації за роботу в таких умовах.

Крім того, при прийнятті на роботу всі працівники повинні за рахунок роботодавця пройти вступний інструктаж, навчання, перевірку знань, первинний інструктаж на робочому місці, стажування і набуття навичок безпечних методів праці. Тільки після цього працівники допускаються до самостійної роботи. Вступний інструктаж проводить спеціаліст з охорони праці, а первинний – безпосередній керівник працівника. Надалі з працівниками повинні проводитися повторні інструктажі (раз на квартал при виконанні робіт підвищеної небезпеки або раз на півріччя), решту позапланові (при зміні правил охорони праці, зміни в обладнанні або при порушенні працівником правил охорони праці) та цільові інструктажі (зокрема, при разових роботах, не пов'язаних зі спеціальністю). Інформація про проведення інструктажів має вноситися до відповідного журналу, завірені підписом як того, кого інструктували, так і того, хто інструктував.

В даному підрозділі було розглянуто основні нормативно-правові документи, коротко оглянуто вимоги до приміщень, та організації обладнання робочих місць.

4.2 Безпека в надзвичайних ситуаціях

Приміщення із робочими місцями користувачів комп'ютерів для забезпечення електробезпеки обладнання, а також для захисту від ураження

електричним струмом самих користувачів ПК повинні мати достатні технічні засоби захисту відповідно до ГОСТ 12.1.009-76, НПАОП 40.1-1.07-01 “Правила експлуатації електрозахисних засобів”, НПАОП 40.1-1.21-98 “Правила безпечної експлуатації електроустановок споживачів”, НПАОП 40.1-1.32-01 “Правила будови електроустановок. Електрообладнання спеціальних установок”

З метою запобігання ушкодженням, що можуть статися через ураження електричним струмом, загоряння, коротке замикання тощо, розроблено загальний стандарт безпеки IEC 950. Загальним стандартом електробезпечності для країн Європейської співдружності є Cemark.

Під час проектування систем електропостачання, монтажу силового електрообладнання та електричного освітлення будівель та приміщень для ПЕОМ необхідно дотримуватись вимог вищеназваних нормативно-правових актів, а також СН 357-77 "Инструкция по проектированию силового осветительного оборудования промышленных предприятий", затверджених Держбудом СРСР, ГОСТу 12.1.006, ГОСТу 12.1.030 "ССБТ. Электробезопасность. Защитное заземление, зануление", ГОСТу 12.1.019 "ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты", ГОСТу 12.1.045, ВСН 59-88 Держкомархітектури СРСР "Электрооборудование жилых и общественных зданий. Нормы проектирования", Правил пожежної безпеки в Україні, ДСанПіН 3.3.2.007-98, розділів СНиП, що стосуються штучного освітлення і електротехнічних пристроїв, та вимог нормативно-технічної і експлуатаційної документації заводу-виробника ПЕОМ.

ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники тощо), електропроводи та кабелі за виконанням та ступенем захисту мають відповідати класу зони за ПУЕ, мати апаратуру захисту від струму короткого замикання та інших аварійних режимів.

Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, перейти на негорючу ізоляцію.

Лінія електромережі для живлення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ виконується як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів.

Використання нульового робочого провідника як нульового захисного провідника забороняється. Нульовий захисний провід прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток живлення. Не допускається підключення на щиті до одного контактного затискача нульового робочого та нульового захисного провідників. Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі повинна бути не менше площі перерізу фазового провідника.

Усі провідники повинні відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту, вимогам ПУЕ.

У приміщенні, де одночасно експлуатується або обслуговується більше п'яти персональних ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

ПЕОМ, периферійні пристрої ПЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ повинні підключатися до електромережі тільки з допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників повинні мати спеціальні контакти для підключення нульового захисного провідника. Конструкція їх має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним. Необхідно унеможливити з'єднання контактів фазових провідників з контактами нульового захисного провідника.

Неприпустимим є підключення ПЕОМ та периферійних пристроїв ПЕОМ до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв.

Електромережі штепсельних з'єднань та електророзеток для живлення ПЕОМ, периферійних пристроїв слід виконувати за магістральною схемою, по 3…6 з'єднань або електророзеток в одному колі. Штепсельні з'єднання та електророзетки для напруги 12 В та 36 В за своєю конструкцією повинні відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В і мають бути пофарбовані в колір, який візуально значно відрізняється від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

Індивідуальні та групові штепсельні з'єднання та електророзетки необхідно монтувати на негорючих або важкогорючих пластинах з урахуванням вимог ПУЕ та Правил пожежної безпеки в Україні.

Електромережу штепсельних розеток для живлення ПЕОМ, периферійних пристроїв ПЕОМ при розташуванні їх уздовж стін приміщення прокладають по підлозі поряд зі стінами приміщення, як правило, в металевих трубах і гнучких металевих рукавах з відводами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання.

При розташуванні в приміщенні за його периметром до 5 ПЕОМ, використанні трипровідникового захищеного проводу або кабелю в оболонці з негорючого або важкогорючого матеріалу дозволяється прокладання їх без металевих труб та гнучких металевих рукавів.

Електромережу штепсельних розеток для живлення ПЕОМ при розташуванні їх у центрі приміщення, прокладають у каналах або під знімною підлогою в металевих трубах або гнучких металевих рукавах. При цьому не дозволяється застосовувати провід і кабель в ізоляції з вулканізованої гуми та інші матеріали, що містять сірку. Відкрита прокладка кабелів під підлогою забороняється. Металеві труби та гнучкі металеві рукави повинні бути заземлені. Заземлення повинно відповідати вимогам НПАОП 40.1-1.21-98.

Для підключення переносної електроапаратури застосовують гнучкі проводи в надійній ізоляції.

Тимчасова електропроводка від переносних приладів до джерел живлення виконується найкоротшим шляхом без заплутування проводів у конструкціях машин, приладів та меблях. Доточувати проводи можна тільки шляхом паяння з наступним старанним ізолюванням місць з'єднання.

Є неприпустимими:

- експлуатація кабелів та проводів з пошкодженою або такою, що втратила захисні властивості за час експлуатації, ізоляцією; залишення під напругою кабелів та проводів з неізольованими провідниками;
- застосування саморобних подовжувачів, які не відповідають вимогам ПВЕ до переносних електропроводок;
- застосування для опалення приміщення нестандартного (саморобного) електронагрівального обладнання або ламп розжарювання;
- користування пошкодженими розетками, розгалужувальними та з'єднувальними коробками, вимикачами та іншими електровиробами, а також лампами, скло яких має сліди затемнення або випинання;
- підвішування світильників безпосередньо на струмопровідних проводах, обгортання електроламп і світильників папером, тканиною та іншими горючими матеріалами, експлуатація їх зі знятими ковпаками (розсіювачами);
- використання електроапаратури та приладів в умовах, що не відповідають вказівкам (рекомендаціям) підприємств-виготовлювачів[44].

ВИСНОВКИ

В результаті виконання дипломної роботи було детально освітлено процес тестування сучасних програмних продуктів, наведена доцільність та доречність використання автоматизованого тестування при оцінці якості програмного забезпечення. Були використані кращі практики та підходи, які використовуються фахівцями у сфері тестування.

В якості системи автоматизації процесу тестування програмного забезпечення було спроектовано та реалізоване програмне забезпечення, яке дозволяє з легкістю запускати тестові скрипти, надавати результати тестових прогонів задля подальшого їх аналізу. Програма була спроектована таким чином, щоб слідувати принципам повторного використання коду, масштабованості та підтримуваності. Таким чином, будь які зміни у програмному забезпеченні можна з легкістю відтворити у структурі та коді програми.

Програмне забезпечення описує об’єкти сторінок, які дозволяють легко та ефективно взаємодіяти із системою ATutor, слідуючи принципам PageObject підходу, який далеко не новий, проте зарекомендував себе з позитивної сторони, й широко використовується фахівцями й сьогодні.

Окрім цього, були спроектовані та документовані тестові випадки, а також на їх основі реалізовані тестові скрипти. Тестові випадки є різноплановими, що дозволяє використовувати як для димного, так і для регресійного тестування програмного забезпечення.

Програмне забпечення дозволяє з легкістю створювати набори нових тестів, та, в разі зміни дизайну системи ATutor, дозволяє проводити зміни лиш в об’єктах сторінок, а не у самих тестах.

Даний дослідницький проект та його реалізація виконувався з метою аналізу та впровадження процесів оцінки якості програмного забезпечення у життєвий цикл розробки. Програмне рішення може бути використане для написання інших тестових сценаріїв задля покривання тестуванням інших частин інформаційної системи управління навчанням ATutor, а вже спроектовані та реалізовані сценарії можна використати для покращення аналізу якості системи ATutor вже сьогодні.

Підсумовуючи проведену роботу, для реалізації програмного рішення та автоматизованих тестових сценаріїв, було використано наступні інструменти, підходи та технології:

- Мову програмування C# та .NET фреймворк;
- Інтегроване середовище розробки Visual Studio;
- Фреймворк для юніт-тестування NUnit;
- Інструмент автоматизації браузерів Selenium;
- XPath та CSS локатори;
- Інструменти розробника браузера Google Chrome;
- Патерн проектування PageObject;
- Фреймворк для автоматизації Atata;
- Інструмент віртуалізації Docker.

Предметна область є цікавою та надзвичайно корисною для сучасних процесів розробки програмного забезпечення. Програмне рішення повинно значно скоротити час та витрати на виконання перевірки якості та тестування програмних продуктів, а також допомогти менеджерам та клієнтам отримати загальну картину про стан тестуємого програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is Software Development? - Definition from Techopedia [Electronic resource] – Mode of access: WWW.URL: https://www.techopedia.com/definition/16431/software-development – Last access: 2019. – Title from the screen.
2. New Product Development Glossary [Electronic resource] – Mode of access: WWW.URL: http://www.npd-solutions.com/glossary.html – Last access: 2019. – Title from the screen.
3. What is education? - Quora [Electronic resource] – Mode of access: WWW.URL: https://www.quora.com/What-is-education-2 – Last access: 2019. – Title from the screen.
4. What is the difference between "informal" and "non formal" learning? [Electronic resource] – Mode of access: WWW.URL : https://web.archive.org/web/20141015103040/http:/www.competencecentre.eu/index.php/home/74-what-is-the-difference-between-qinformalq-and-qnon-formalq-learning. – Last access: 2019. – Title from the screen.
5. Robinson, Rhonda; Molenda, Michael; Rezabek, Landra : Facilitating Learning [Electronic resource] – Mode of access: WWW.URL : https://www.aect.org/publications/EducationalTechnology/ER5861X_C002.pdf – Last access: 2018. – Title from the screen.
6. What is education? - Quora [Electronic resource] – Mode of access: WWW.URL: https://www.quora.com/What-is-educational-technology – Last access: 2019. – Title from the screen.
7. D. Randy Garrison; Terry Anderson. E-Learning in the 21st Century: A Framework for Research and Practice [Text] – 2003 – 184 p.
8. Al Januszewski A.; Molenda Michael. Educational Technology: A Definition with Commentary [Text] – 2007 – 384 p.
9. Lowenthal, P. R.; Wilson, B. G. Labels do matter! A critique of AECT's redefinition of the field [Electronic resource] – Mode of access: WWW.URL : http://patricklowenthal.com/publications/LabelsDoMatter--CritiqueAECTsRedefinitionOfTheField.pdf – Last access: 2019. – Title from the screen.

10. Terras, Melody; Ramsay. The five central psychological challenges facing effective mobile learning [Electronic resource] – Mode of access: WWW.URL : https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8535.2012.01362.x – Last access: 2019. – Title from the screen.

11. Top 5 Benefits of Mobile Learning [Electronic resource] – Mode of access: WWW.URL : https://www.eidesign.net/top-5-benefits-mobile-learning/ – Last access: 2019. – Title from the screen.

12. What is screencasting? [Electronic resource] – Mode of access: WWW.URL : https://whatis.techtarget.com/definition/screencast – Last access: 2019. – Title from the screen.

13. Virtual Learning Environment (VLE) [Electronic resource] – Mode of access: WWW.URL: https://fireflylearning.com/what-is-a-virtual-learning-environment-vle – Last access: 2019. – Title from the screen.

14. *Ross, S., Morrison, G., Lowther, D. Educational technology research past and present: balancing rigor and relevance to impact learning* [Electronic resource] – Mode of access: WWW.URL : http://www.cedtech.net/articles/11/112.pdf – Last access: 2019. – Title from the screen.

15. Getting At-Risk Teens to Graduation [Electronic resource] – Mode of access: WWW.URL : https://www.educationnext.org/getting-at-risk-teens-to-graduation – Last access: 2019. – Title from the screen.

16. Testimonial, Modern education systems [Electronic resource] – Mode of access: WWW.URL: https://xpertcube.com/modern-education-system/ – Last access: 2019. – Title from the screen.

17. Why PT3? An Analysis of the Impact of Educational Technology [Electronic resource] – Mode of access: WWW.URL : https://www.citejournal.org/volume-4/issue-3-04/general/why-pt3-an-analysis-of-the-impact-of-educational-technology/ – Last access: 2019. – Title from the screen.

18. Hiring Practices and Attitudes: Traditional vs. Online Degree Credentials SHRM Poll [Electronic resource] – Mode of access: WWW.URL : https://www.shrm.org/hr-today/trends-and-forecasting/research-and-surveys/pages/hiringpracticesandattitudes.aspx – Last access: 2019. – Title from the screen.

19. Culp, K.M.; Honey, M.; Mandinach, E. Retrospective on twenty years of education technology policy [Electronic resource] – Mode of access: WWW.URL :

http://ocw.metu.edu.tr/file.php/118/Week12/Culp_JECR.pdf – Last access: 2018. – Title from the screen.

20. Technical Evaluation Report 37. Assistive Software for Disabled Learners [Electronic resource] – Mode of access: http://www.irrodl.org/index.php/irrodl/article/viewArticle/198/280 – Last access: 2018. – Title from the screen.

21. Web Content Accessibility Guidelines 1.0 [Electronic resource] – Mode of access: https://www.w3.org/TR/WAI-WEBCONTENT/ – Last access: 2018. – Title from the screen.

22. St.Amant, Kirk. Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives [Text] – 2007 – 767 p.

23. ATutor Features [Electronic resource] – Mode of access: https://atutor.github.io/atutor/features.html – Last access: 2019. – Title from the screen.

24. What is software testing - Definition [Electronic resource] – Mode of access: https://www.softwaretestingmaterial.com/software-testing/ – Last access: 2019. – Title from the screen.

25. Why is software testing necessary? [Electronic resource] – Mode of access: http://tryqa.com/why-is-testing-necessary/ – Last access: 2019. – Title from the screen.

26. What are software testing objectives and purpose? [Electronic resource] – Mode of access: http://tryqa.com/what-is-the-software-testing-objectives-and-purpose/ – Last access: 2019. – Title from the screen.

27. Software Testing Fundamentals – Test Plan [Electronic resource] – Mode of access: http://softwaretestingfundamentals.com/test-plan/ – Last access: 2019. – Title from the screen.

28. IEEE standard for test documentation [Electronic resource] – Mode of access: https://web.cs.dal.ca/~arc/teaching/CS3130/Templates/TestingTemplates/Test%20Plan%20Templates/IEEEStandardTestPlans.doc – Last access: 2019. – Title from the screen.

29. Bourque, Pierre; Fairley, Richard E. Guide to the Software Engineering Body of Knowledge [Electronic resource] – Mode of access: http://www4.ncsu.edu/~tjmenzie/cs510/pdf/SWEBOKv3.pdf – Last access: 2018. – Title from the screen.

30. Dooley, J. Software Development and Professional Practice [Text] – 2011 – 193 p.

31. Definition of System Testing [Electronic resource] – Mode of access: https://economictimes.indiatimes.com/definition/system-testing – Last access: 2019. – Title from the screen.

32. Clapp, J.A.; Stanten, S.F.; Peng, W.W. Software Quality Control, Error Analysis, and Testing [Text] – 1995 – 192 p.

33. Theory: Complexity and testing reductions [Electronic resource] – Mode of access: https://pdfs.semanticscholar.org/f908/824c11bf9a698d9f74de69e18358f7f48f7d.pdf – Last access: 2019. – Title from the screen.

34. Docker – Wikipedia [Electronic resource] – Mode of access: https://ru.wikipedia.org/wiki/Docker – Last access: 2019. – Title from the screen.

35. Vivek Ratan. Docker: A Favourite in the DevOps World [Electronic resource] – Mode of access: https://opensourceforu.com/2017/02/docker-favourite-devops-world/ – Last access: 2019. – Title from the screen.

36. When and Why Use Docker [Electronic resource] – Mode of access: https://www.linode.com/docs/applications/containers/when-and-why-to-use-docker/ – Last access: 2019. – Title from the screen.

37. XPath [Electronic resource] – Mode of access: https://whatis.techtarget.com/definition/XPath – Last access: 2019. – Title from the screen.

38. Selenium WebDriver [Електронний ресурс] – Режим доступу: https://economictimes.indiatimes.com/definition/selenium-web-driver – Title from the screen.

39. Definition NUnit [Електронний ресурс] – Режим доступу: https://searchsoftwarequality.techtarget.com/definition/NUnit – Title from the screen.

40. PageObject pattern [Electronic resource] – Mode of access: https://www.swtestacademy.com/page-object-model-c/ – Last access: 2019. – Title from the screen.

41. Податковий кодекс України [Електронний ресурс] – Режим доступу: http://zakon5.rada.gov.ua/laws/show/2755-17. – Title from the screen.

42. Положення (стандарт) бухгалтерського обліку [Електронний ресурс] – Режим доступу: http://zakon0.rada.gov.ua/laws/show/z0027-00 - Назва з екрану.

43. В.І. Голінько, М.Ю. Іконніков, Я.Я. Лебедєв Охорона праці в галузі інформаційних технологій [Текст] – Дніпропетровськ : НГУ – 2015. – 247 с.

44. Вимоги до електробезпеки у офісних приміщеннях [Електронний ресурс] – Режим доступу: https://cpo.stu.cn.ua/Oksana/posibnik/1140.html – Title from the screen.

45. М.Р. Петрик, Д.М. Михалик, Я.І. Кінах, С.В. Гладьо, Г.Б. Цуприк. Методичні вказівки до виконання магістерської роботи рівня “магістр” студентами усіх форм навчання для напряму підготовки 121 – “Інженерія програмного забезпечення” [Текст] – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя – 2016 – 27 с.

46. Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. Методичні рекомендації по виконанню розділу техніко-економічного обґрунтування дипломних робіт студентами технічних спеціальностей напряму підготовки 8.05010302 “Інженерія програмного забезпечення” освітньо-кваліфікаційного рівня “Магістр” [Текст] – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя – 2016 – 28 с.

ДОДАТКИ

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра програмної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедрою

програмної інженерії

"___ "_____ 2019 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської дипломної роботи

на тему: «Розробка програмного забезпечення для автоматизованого тестування Інтернет ресурсу на мові програмування C#»

Петрук Олександру Володимировичу

Керівник роботи:

д.т.н., професор Пастух І.В.

"___" _____ 2019 р.

_____

Виконавець:

студент групи СПм-62

Петрук Олександр Володимирович

"___" _____ 2019р.

_____

м. Тернопіль – 2019

ЗМІСТ

1 ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану на 2019 рік, та згідно наказу на виконання дипломної роботи студента-магістра.

Тема проекту: «Розробка програмного забезпечення для автоматизованого тестування Інтернет ресурсу на мові програмування C#».

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Якість програмного забезпечення – надзвичайно важливий та незамінний критерій будь-якої сучасної програмної системи. Оскільки роль та призначення сучасних комп’ютерних систем варіюється як за важливістю, так і за ризиками та впливом на людське життя, неякісне програмне забезпечення ставить під загрозу багато речей, починаючи з рейтингу та авторитетності компаній-замовників та їхньої конкурентоспроможності, й закінчуючи фінансовим становищем користувачів цих систем, а також станом їхнього здоров’я (у випадку, якщо мова йде про медичне програмне забезпечення).

Автоматизоване тестування, що тісно вплетене в процес розробки програмного забезпечення, сприяє загальному покращенню якості сучасних програмних систем та підвищенню їхнього попиту на ринку.

Об’єктом дослідження є процес розробки сучасних програмних систем. Предмет дослідження: якість програмного забезпечення та можливість її покращення впроваджуючи автоматизоване тестування у процес розробки програмного забезпечення. У даній дослідницькій роботі застосовуються сучасні методи розробки та написання автоматизованого тестування веб- додатків за допомогою таких інструментів, як мова програмування C#, а також NUnit та Selenium. Засобом для цього є існуюча програмна система управління навчанням ATutor.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Функціональні характеристики

Програмне забезпечення має виконувати наступні дії:

- надавати доступ до виконання тестових скриптів за допомогою розробленого програмного рішення;
- автоматизовано тестувати комплексні веб-сайти за допомогою інструментів Selenium WebDriver та NUnit;
- допомагати у створенні нових скриптів автоматизації тестування програмного забезпечення;

3.2 Склад та параметри технічних засобів

1) ПК із 4096 Мб оперативної пам‘яті, встановленою операційною системою Windows 7, 8, 8.1, 10. Не менше 200 Мб вільного місця на жорсткому диску. Двох ядерний процесор з тактовою частотою від 1.3 GHz і більше.

2) Наявність встановленого .NET фреймворку версії 4.5 й більше.

3) Наявність встановленого NUnit Console Runner для запуску тестів із dll-файлів.

3.3 Інформаційна та програмна сполучність

Програмний продукт повинен коректно функціонувати в операційних системах Windows 7, 8, 8.1, 10, на яких доступний для встановлення .NET фреймворк версії 4.5 й більше. Розроблювана система повинна бути пристосована для автоматизованого тестування інформаційних систем. Розробку виконувати у середовищі розробки Visual Studio з використанням інструментів Selenium 2.45 та NUnit 3

4. СТАДІЇ РОЗРОБКИ

В ходів реалізації роботи проект повинен пройти крізь наступні стадії розробки:

- аналіз предметної області;

- аналіз важливості тестування у процесах розробки програмного забезпечення;

- аналіз автоматизованого тестування програмних систем;

- розгортання системи ATutor для демонстрації тестового процесу;

- написання тестових випадків;

- проектування та розробка програмного рішення для автоматизованого тестування;

- проектування та розробка скриптів для автоматизованого тестування;

- оформлення супровідної документації;

- здача роботи.

5. ПРОГРАМНА ДОКУМЕНТАЦІЯ

Для програмного продукту повинні бути розроблені наступні документи:

- Пояснювальна записка;

- Технічне завдання;

- Презентаційний матеріал;

- Додатки.

6. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Розроблений програмний продукт має виконувати всі вимоги, що складаються з перерахованих у п. 3.1 характеристик.

Приймання проводиться спеціально створеною екзаменаційною комісією в термін до:

“__” грудня 2019р.

УДК 004.422.81
О.В. Петрук – магістрант
Тернопільський національний технічний університет імені Івана Пулюя, Україна
О. А. Пастух – кандидат технічних наук, доцент
Тернопільський національний технічний університет імені Івана Пулюя, Україна

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ ІНТЕРНЕТ РЕСУРСУ

O.V. Petruk – magistrate, O. A. Pastuh – Ph.D, Assoc. Prof.

DEVELOPMENT SOFTWARE FOR AUTOMATION TESTING INTERNET RESOURCE

Уся сучасна інформація в тому чи іншому вигляді – є віртуальною, де користувач має змогу знайти потрібне йому Інтернет джерело та виконати необхідні йому дії над потрібною інформацією. Тому є необхідність в певному рівні якості веб сайту, який надає ту чи іншу інформацію. З розвитком інформаційних технологій з’явилось безліч програмних систем для допоги забезпечення необхідного рівня якості, в тому числі і вільного програмного забезпечення. Для прикладу розглянемо найпопулярніші з них. Програмна система типу Selenium[1] - це інструмент для автоматизації дій веб-браузера. У більшості випадків використовується для тестування Web-додатків, але цим не обмежується. Здатна записувати дії користувача у веб-браузері та відтворювати їх у подальшому. Сумісна з операційними системами Linux/Windows/ MAC OS.

Програмна система Apache Jmeter[2] - інструмент для проведення навантажувального тестування, що розробляється Apache Software Foundation. Хоча спочатку JMeter розроблявся як засіб тестування web-додатків, в даний час він здатний проводити навантажувальні тести для JDBC-з'єднань, FTP, LDAP, SOAP, JMS, POP3, IMAP, HTTP і TCP.

Цікава можливість створення великої кількості запитів за допомогою декількох комп'ютерів при управлінні цим процесом з одного з них. Архітектура, що підтримує плагіни сторонніх розробників, дозволяє доповнювати інструмент новими функціями.

У програмі реалізовані механізми авторизації віртуальних користувачів, підтримуються призначені для користувача сеанси.

Підсумовуючи весь матеріал можна дійти висновку, що розробка програмного забезпечення для автоматизованого тестування Інтернет ресурсів є необхідною для забезпечення відповідного рівня якості Інтернет ресурсу. Особливу увагу потрібно приділити вибору інструментів для розробки програмного забезпечення та сценаріям тестування. Проте перед використанням автоматизованих сценаріїв тестування їх необхідно розробити та перевірити, щоб вони відповідали всім вимогам, були зрозумілими у використанні не тільки професіоналами, але і звичайнми користувачами.

Література

1. Вікіпедія / Selenium [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Selenium
2. Вікіпедія / JMeter [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/JMeter

```
FROM php:5.6.30-apache
LABEL "maintainer"="Daniele Demichelis <demichelis@danidemi.com>" \
      "version.php"="5.6.30" \
      "version.Atutor"="2.2.2"
EXPOSE 80

RUN apt-get update; \
    apt-get install -y wget unzip; \
    docker-php-ext-install mysql; \
    apt-get install -y libfreetype6-dev libjpeg62-turbo-dev wget unzip; \
    docker-php-ext-configure gd --with-freetype-dir=/usr/include/ --with-jpeg-
dir=/usr/lib; \
    docker-php-ext-install gd

RUN touch /var/log/php-errors.log; \
    echo "date.timezone=Europe/Rome" >> /usr/local/etc/php/conf.d/docker-php.ini; \
    echo "display_errors = Off" >> /usr/local/etc/php/conf.d/docker-php.ini; \
    echo "log_errors = On" >> /usr/local/etc/php/conf.d/docker-php.ini; \
    echo "error_log = /dev/stdout" >> /usr/local/etc/php/conf.d/docker-php.ini;

RUN wget -O /tmp/atutor.zip --quiet
https://github.com/atutor/ATutor/archive/atutor_2_2_2.zip; \
    unzip /tmp/atutor.zip -d /tmp; \
    rm -rf /var/www/html; \
    mv /tmp/ATutor-atutor_2_2_2 /var/www/html/; \
    touch /var/www/html/include/config.inc.php; \
    chmod a+rw -R /var/www/html/; \
    echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php; \
    rm /tmp/atutor.zip
```

```
RUN mkdir /content; \
    chmod a+rw -R /content
```

ДОДАТОК Е – ДИСК

Код класу-тригера ScrollIntoElementCenterAttribute:

```
using Atata;
namespace ATutorAutomation.Framework.Triggers
{
   public class ScrollIntoElementCenterAttribute : TriggerAttribute
   {
      public ScrollIntoElementCenterAttribute(TriggerEvents @on =
TriggerEvents.BeforeAccess, TriggerPriority priority = TriggerPriority.Highest) :
base(@on, priority)
      {
      }

      protected override void Execute<TOwner>(TriggerContext<TOwner> context)
      {
         context.Driver.ExecuteScript("arguments[0].scrollIntoView({block: 'center'});",
context.Component.Scope);
      }
   }
}
```

Код класу AdminMainPageMenu:

```
using Atata;
using ATutorAutomation.Framework.Pages;
using ATutorAutomation.Framework.Pages.AdminPages.Home;
using ATutorAutomation.Framework.Pages.AdminPages.Courses;
using ATutorAutomation.Framework.Pages.AdminPages.Users;

namespace ATutorAutomation.Framework.Controls.Navigation.Admin
{
   [ControlDefinition("ul[@id='topnavlist']")]
```

```
    public class AdminMainPageMenu<TOwner> : Control<TOwner>
        where TOwner : PageObject<TOwner>
    {
        public LinkDelegate<AdminMainPage, TOwner> Home { get; private set; }

        public LinkDelegate<UsersPage, TOwner> Users { get; private set; }

        public LinkDelegate<AdminCoursesPage, TOwner> Courses { get; private set; }
    }
}
```

Код класу AdminCourseCategoriesSubMenu:

```
using Atata;
using ATutorAutomation.Framework.Pages;
using ATutorAutomation.Framework.Pages.AdminPages.Courses;

namespace ATutorAutomation.Framework.Controls.Navigation.Admin
{
    [ControlDefinition("ul[@id='subnavlist']")]
    public class AdminCourseCategoriesSubMenu<TOwner> : Control<TOwner>
        where TOwner : PageObject<TOwner>
    {
        public LinkDelegate<AdminCreateCateroryPage, TOwner> CreateCategory { get;
private set; }
    }
}
```

Код класу AdminCoursesSubMenu:

```
using Atata;
using ATutorAutomation.Framework.Pages;
using ATutorAutomation.Framework.Pages.AdminPages.Courses;
```

```
namespace ATutorAutomation.Framework.Controls.Navigation.Admin
{
   [ControlDefinition("ul[@id='subnavlist']")]
   public class AdminCoursesSubMenu<TOwner> : Control<TOwner>
      where TOwner : PageObject<TOwner>
   {
      public LinkDelegate<AdminCoursesPage, TOwner> Courses { get; private set; }
      public LinkDelegate<AdminCourseCategoriesPage, TOwner> Categories { get;
private set; }
   }
}
```

Код класу AdminUsersSubMenu:

```
using Atata;
using ATutorAutomation.Framework.Pages;
using ATutorAutomation.Framework.Pages.AdminPages.Users;

namespace ATutorAutomation.Framework.Controls.Navigation.Admin
{
   [ControlDefinition("ul[@id='subnavlist']")]
   public class AdminUsersSubMenu<TOwner> : Control<TOwner>
      where TOwner : PageObject<TOwner>
   {
      public LinkDelegate<UsersPage, TOwner> Users { get; private set; }
      public LinkDelegate<UserRegisterPage, TOwner> CreateUserAccount { get;
private set; }
   }
}
```

Код класу LoginPage:

```
using Atata;
using ATutorAutomation.Framework.Pages.AdminPages.Home;
```

```
using ATutorAutomation.Framework.Pages.NonAdminPages;

namespace ATutorAutomation.Framework.Pages
{
    using _ = LoginPage;

    public class LoginPage : Page<_>
    {
        [FindById("login")]
        public TextInput<_> LoginNameOrEmail { get; private set; }

        [FindById("pass")]
        public PasswordInput<_> Password { get; private set; }
        public Button<AdminMainPage, _> Login { get; private set; }
        public Button<UserRegisterPage, _> Register { get; private set; }
    }
}
```

Код класу UserRegisterPage:

```
using Atata;
using ATutorAutomation.Framework.Pages.NonAdminPages.MyCourses;

namespace ATutorAutomation.Framework.Pages
{
    using _ = UserRegisterPage;

    public class UserRegisterPage : Page<_>
    {
        public TextInput<_> LoginName { get; private set; }

        public PasswordInput<_> Password { get; private set; }
```

```
        public PasswordInput<_> PasswordAgain { get; private set; }

        [RandomizeStringSettings("{0}@mail.com")]
        public TextInput<_> EmailAddress { get; private set; }

        public TextInput<_> EmailAddressAgain { get; private set; }

        public TextInput<_> FirstName { get; private set; }

        public TextInput<_> LastName { get; private set; }

        [FindByXPath("div[text() = 'Account Status']")]
        public RadioButtonList<AccountStatus, _> AccountStatus { get; private set; }

        [ScrollDown]
        public Button<MyCoursesPage, _> Save { get; private set; }
    }

    public enum AccountStatus
    {
        Disabled,
        Student,
        Instructor
    }
}
```

Код класу MyCoursesPage:

```
using Atata;
using ATutorAutomation.Framework.Controls;
using ATutorAutomation.Framework.Controls.Navigation.NonAdmin;
using ATutorAutomation.Framework.Pages.CoursePages.CourseHome;
```

```
namespace ATutorAutomation.Framework.Pages.NonAdminPages.MyCourses
{
    using _ = MyCoursesPage;

    public class MyCoursesPage : Page<_>
    {
        [FindByXPath("//div[@id='feedback']//ul")]
        public UnorderedList<ListItem<_>, _> Feedback { get; private set; }
        public NonAdminMainPageMenu<_> MainMenu { get; private set; }
        public NonAdminMyCoursesSubMenu<_> SubMenu { get; private set; }
        public CustomTable<CourseRow, _> CoursesTable { get; private set; }

        public class CourseRow : CoursesTableRow<_>
        {
            public LinkDelegate<CourseHomePage, _> Course { get; private set; }
            public Text<_> Instructor { get; private set; }
            public Text<_> Status { get; private set; }
            [FindByContent]
            public LinkDelegate<CourseUnenrollConformationPage, _> Unenroll { get;
private set; }
        }
    }
}
```

Код класу BrowseCoursesPage:

```
using Atata;
using ATutorAutomation.Framework.Controls;
using ATutorAutomation.Framework.Controls.Navigation.NonAdmin;
using ATutorAutomation.Framework.Pages.CoursePages.CourseHome;

namespace ATutorAutomation.Framework.Pages.NonAdminPages.BrowseCourses
{
```

```
    using _ = BrowseCoursesPage;

    public class BrowseCoursesPage : Page<_>
    {
        [FindByXPath("//div[@id='feedback']//ul")]
        public UnorderedList<ListItem<_>, _> Feedback { get; private set; }
        public NonAdminMainPageMenu<_> MainMenu { get; private set; }
        public CustomTable<CourseRow, _> CoursesTable { get; private set; }

        public class CourseRow : CoursesTableRow<_>
        {
            public LinkDelegate<CourseHomePage, _> Title { get; private set; }
            public Text<_> Description { get; private set; }
            public Text<_> Category { get; private set; }
            public Text<_> Instructor { get; private set; }
            public Text<_> Access { get; private set; }
        }
    }
}
```

Код класу AddQuestionsToTestPage:

```
using Atata;
using ATutorAutomation.Framework.Controls;

namespace ATutorAutomation.Framework.Pages.CoursePages.Manage
{
    using _ = AddQuestionsToTestPage;

    public class AddQuestionsToTestPage : Page<_>
    {
        public Select<_> Category { get; private set; }
        public Button<_, _> Filter { get; private set; }
```

```
        [Term("Add to Test/Survey")]
        public Button<ConfirmAddingQuestionsToTestPage, _> AddToTestSurvey { get;
private set; }
        public CustomTable<QuestionRow, _> QuestionsTable { get; private set; }

        public class QuestionRow : QuestionsTableRow<_>
        {
            [FindByXPath("input[@type='checkbox']")]
            public CheckBox<_> Check { get; private set; }
            public Text<_> Question { get; private set; }
            public Text<_> Type { get; private set; }
        }
    }
}
```

Код класу CreateTestSurveyPage:

```
using Atata;
using ATutorAutomation.Framework.Pages.NonAdminPages.BrowseCourses;
using ATutorAutomation.Framework.Triggers;

namespace ATutorAutomation.Framework.Pages.CoursePages.Manage
{
    using _ = CreateTestSurveyPage;

    public class CreateTestSurveyPage : Page<_>
    {
        public TextInput<_> Title { get; private set; }
        public Select<_> AttemptsAllowed { get; private set; }
        [ScrollIntoElementCenter]
        [FindByXPath("//div[@class='row']")]
        public RadioButtonList<PassScoreEnum, _> PassScore { get; private set; }
        [ScrollIntoElementCenter]
```

```
        [FindById("passpercent")]
        public TextInput<_> PercentageScore { get; private set; }
        [ScrollIntoElementCenter]
        public TextArea<_> PassFeedback { get; private set; }
        [ScrollIntoElementCenter]
        public TextArea<_> FailFeedback { get; private set; }
        [ScrollDown]
        [FindByXPath("//div[@class='row']")]
        public RadioButtonList<ReleaseRusultsEnum, _> ReleaseResults { get; private
set; }
        [ScrollDown]
        [FindByName("day_start")]
        public Select<_> StartTestDay { get; private set; }
        [ScrollDown]
        [FindByName("day_end")]
        public Select<_> EndTestDay { get; private set; }
        [ScrollDown]
        public Button<TestsAndSurveysPage, _> Save { get; private set; }
    }

    public enum PassScoreEnum
    {
        [Term("No pass score")]
        NoPassScore,
        [Term("% percentage score")]
        PercentageScore,
        [Term("points score")]
        PointsScore
    }

    public enum ReleaseRusultsEnum
    {
```

```
        [Term("Once quiz has been submitted")]
        OnceQuizHasBeenSubmited,
        [Term("Once quiz has been submitted and all questions have been marked")]
        OnceQuizHasBeenSubmitedWithAllQuestions,
        [Term("Do not release results")]
        DoNotRelease
    }
}
```

Код класу TestsAndSurveysPage:

```
using Atata;
using ATutorAutomation.Framework.Controls.Navigation.NonAdmin;

namespace ATutorAutomation.Framework.Pages.CoursePages.Manage
{
    using _ = TestsAndSurveysPage;

    public class TestsAndSurveysPage : Page<_>
    {
        public NonAdminTestsAndSurveysSubMenu<_> SubMenu { get; private set; }
        [FindByClass("data")]
        public Table<TestRow, _> TestsTable { get; private set; }
        public Button<QuestionsForTestPage, _> Questions { get; private set; }

        public class TestRow : TableRow<_>
        {
            [FindByXPath("input[@type='radio']")]
            public RadioButton<_> Check { get; private set; }
            public Text<_> Title { get; private set; }
            public Text<_> Status { get; private set; }
            public Text<_> Availability { get; private set; }
            public Text<_> ReleaseResults { get; private set; }
            public Text<_> Submissions { get; private set; }
```

```
        public Text<_> AssignedTo { get; private set; }
    }
}}
```

Код класу MyTestsAndSurveysPage:

```
using Atata;
using ATutorAutomation.Framework.Controls;

namespace ATutorAutomation.Framework.Pages.CoursePages.CourseHome
{
    using _ = MyTestsAndSurveysPage;

    public class MyTestsAndSurveysPage : Page<_>
    {
        [FindByXPath("h4/preceding-sibling::table")]
        public MyTestsAndSubmissionsTable<TestRow, _> TestsTable { get; private set;
}
        public class TestRow : CustomTableRowBase<_>
        {
            public LinkDelegate<TestInformationPage, _> Title { get; private set; }
            public Text<_> Status { get; private set; }
        }

        [FindByXPath("h4/following-sibling::table")]
        public MyTestsAndSubmissionsTable<SubmissionRow, _> SubmissionsTable {
get; private set; }
        public class SubmissionRow : CustomTableRowBase<_>
        {
            public Text<_> Title { get; private set; }
            public Text<_> DateTaken { get; private set; }
        }
    }}
```

Код класу UITestFixture:

```
using Atata;
using NUnit.Framework;
using ATutorAutomation.Framework.Pages;
using ATutorAutomation.Framework.Pages.AdminPages.Home;
using ATutorAutomation.Framework.Pages.NonAdminPages.MyCourses;
namespace ATutorAutomation.Tests
{
   [TestFixture]
   public class UITestFixture
   {
      [SetUp]
      public void SetUp()
      {
         AtataContext.Configure().
            ApplyJsonConfig<AppConfig>().
            Build();
      }

      [TearDown]
      public void TearDown()
      {
         AtataContext.Current?.CleanUp();
      }

      public AdminMainPage LoginAsAdmin()
      {
         return Go.To<LoginPage>().
            LoginNameOrEmail.Set(AppConfig.Current.AdminAccount.Login).
            Password.Set(AppConfig.Current.AdminAccount.Password).
```

```
            Login.ClickAndGo<AdminMainPage>();
        }

        public MyCoursesPage LoginAsDefaultInstructor ()
        {
            return LoginAsNonAdminUser(
                AppConfig.Current.InstructorAccount.Login,
                AppConfig.Current.InstructorAccount.Password);
        }

        public MyCoursesPage LoginAsSimpleUser (string login, string password)
        {
            return Go.To<LoginPage>().
                LoginNameOrEmail.Set(login).
                Password.Set(password).
                Login.ClickAndGo<MyCoursesPage>();
        }
    }
}
```

Код класу SmokeTestSuite:

```
using Atata;
using NUnit.Framework;
using ATutorAutomation.Framework.Pages;
using ATutorAutomation.Framework.Pages.AdminPages.Users;
using ATutorAutomation.Framework.Pages.CoursePages.Manage;

namespace ATutorAutomation.Tests.Tests
{
    public class SmokeTestSuite : UITestFixture
```

```
    {
        private const string SuccessfulLogInMessage = "You have logged in
successfully.";
        private const string SuccessfulRegistrationMessage = "You have successfully
logged in and have been successfully enrolled in the following courses:";
        private const string _COURSE_CATEGORY_NAME = "Automation Category";
        private const string _COURSE_NAME = "Automation Course";
        private const string _STUDENT_LOGIN = "atutorstudent";
        private const string _QUESTION_CATEGORY_NAME = "Automation Question
Category";
        private const string _QUESTION_TEXT = "Mario - cool game?";
        private const string _TEST_NAME = "Automation Test";
        private const string _TEST_PASS_FEEDBACK = "Good job!";
        private const string _TEST_FAIL_FEEDBACK = "Not great :(";

        [Test, Order(1)]
        public void AdminLogIn()
        {
            LoginAsAdmin().
                Feedback.Items.
                Contents.Should.Contain(SuccessfulLogInMessage);
        }

        [Test, Order(2)]
        public void InstructorLogIn()
        {
            LoginAsDefaultInstructor().
                Feedback.Items.
                Contents.Should.Contain(SuccessfulLogInMessage);
        }

        [Test, Order(3)]
```

```
public void RegisterNewUser()
{
    Go.To<LoginPage>().
        Register.ClickAndGo().
            LoginName.SetRandom().
            Password.Set(AppConfig.Current.DefaultPassword).
            PasswordAgain.Set(AppConfig.Current.DefaultPassword).
            EmailAddress.SetRandom(out string email).
            EmailAddressAgain.Set(email).
            FirstName.SetRandom().
            LastName.SetRandom().
        Save.ClickAndGo().Feedback.Items.
        Contents.Should.Contain(SuccessfulRegistrationMessage);
}

[Test, Order(4)]
public void NewCourseCategoryCanBeAdded()
{
    LoginAsAdmin().
        MainMenu.Courses.ClickAndGo().
        SubMenu.Categories.ClickAndGo().
        SubMenu.CreateCategory.ClickAndGo().
            Title.Set(_COURSE_CATEGORY_NAME).
            Save.ClickAndGo().
        CategoriesTable.Rows
            [x => x.Name.Content == _COURSE_CATEGORY_NAME].
        Should.Exist();
}

[Test, Order(5)]
public void NewCourseCanBeAdded()
{
```

```
    string courseName = LoginAsDefaultInstructor().
        SubMenu.CreateCourse.ClickAndGo().
            Title.Set(_COURSE_NAME).
            Category.Set(_COURSE_CATEGORY_NAME).
            Save.ClickAndGo().
        CourseTitle.Content.Value.TrimEnd();
    Assert.That(courseName, Is.EqualTo(_COURSE_NAME));
}

[Test, Order(6)]
public void NewStudentCanBeRegistered()
{
    LoginAsAdmin().
        MainMenu.Users.ClickAndGo().
        SubMenu.CreateUserAccount.ClickAndGo().
            LoginName.Set(_STUDENT_LOGIN).
            Password.Set(AppConfig.Current.DefaultPassword).
            PasswordAgain.Set(AppConfig.Current.DefaultPassword).
            EmailAddress.SetRandom(out string email).
            EmailAddressAgain.Set(email).
            FirstName.SetRandom().
            LastName.SetRandom().
            AccountStatus.Set(AccountStatus.Student).
        Save.ClickAndGo<UsersPage>().
        UsersTable.Rows
            [x => x.LoginName.Content == _STUDENT_LOGIN].Should.Exist();
}

[Test, Order(7)]
public void StudentCanBeAssignedToTheCorse()
{
```

```
        var ff = LoginAsSimpleUser(_STUDENT_LOGIN,
AppConfig.Current.DefaultPassword).
            MainMenu.BrowseCourses.ClickAndGo<BrowseCoursesPage>().
            CoursesTable.Rows[x => x.Title.Content ==
_COURSE_NAME].Title.ClickAndGo().
                EnrollMe.ClickAndGo().
                EnrollMe.ClickAndGo().
                   CoursesTable.
                   Rows[x => x.Course.Content == _COURSE_NAME].Should.Exist();
    }

    [Test, Order(8)]
    public void NewQuestionCategoryCanBeCreated()
    {
       LoginAsDefaultInstructor().
            MainMenu.MyCourses.ClickAndGo().
            CoursesTable.Rows
               [x => x.Course.Content == _COURSE_NAME].Course.ClickAndGo().
               MainMenu.Manage.ClickAndGo().
               TestsAndSurveys.ClickAndGo().
               SubMenu.QuestionCategories.ClickAndGo().
               CreateCategory.ClickAndGo().
                  Title.Set(_QUESTION_CATEGORY_NAME).
                  Save.ClickAndGo().
               QuestionCategoriesList
                  [x => x.Content ==
_QUESTION_CATEGORY_NAME].Should.Exist();
    }

    [Test, Order(9)]
    public void NewQuestionCanBeCreated()
    {
```

```
    LoginAsDefaultInstructor().
        MainMenu.MyCourses.ClickAndGo().
        CoursesTable.Rows
            [x => x.Course.Content == _COURSE_NAME].Course.ClickAndGo().
        MainMenu.Manage.ClickAndGo().
        TestsAndSurveys.ClickAndGo().
        SubMenu.QuestionBank.ClickAndGo().
        CreateNewQuestion.Set("True or False").
        Create.ClickAndGo<NewTrueFalseQuestionPage>().
        Category.Set(_QUESTION_CATEGORY_NAME).
        Statement.Set(_QUESTION_TEXT).
        Answer.Set(true).
        Save.ClickAndGo().
        QuestionsTable.Rows
            [x => x.Question.Content == _QUESTION_TEXT].Should.BeVisible();
}


[Test, Order(10)]
public void NewTestCanBeCreated()
{
    CreateTestSurveyPage createTestPage = LoginAsDefaultInstructor().
      MainMenu.MyCourses.ClickAndGo().
      CoursesTable.Rows
      [x => x.Course.Content == _COURSE_NAME].Course.ClickAndGo().
      MainMenu.Manage.ClickAndGo().
      TestsAndSurveys.ClickAndGo().
      SubMenu.CreateTestSurvey.ClickAndGo();

    string currentDay = createTestPage.
      Title.Set(_TEST_NAME).
      AttemptsAllowed.Set("5").
```

```
            PassScore.Set(PassScoreEnum.PercentageScore).
            PercentageScore.Set("50").
            PassFeedback.Set(_TEST_PASS_FEEDBACK).
            FailFeedback.Set(_TEST_FAIL_FEEDBACK).
            ReleaseResults.Set(ReleaseRusultsEnum.OnceQuizHasBeenSubmited).
            EndTestDay.Value;

        string nextDay = (int.Parse(currentDay) + 1).ToString();
        string previousDay = (int.Parse(currentDay) - 1).ToString();

        createTestPage.EndTestDay.Set(nextDay).
            StartTestDay.Set(previousDay).
            Save.ClickAndGo().
            TestsTable.Rows[x => x.Title.Content == _TEST_NAME].Should.Exist();
    }

    [Test, Order(11)]
    public void QuestionCanBeAddedToTheTest()
    {
        LoginAsDefaultInstructor().
            MainMenu.MyCourses.ClickAndGo().
            CoursesTable.Rows
              [x => x.Course.Content == _COURSE_NAME].Course.ClickAndGo().
            MainMenu.Manage.ClickAndGo().
            TestsAndSurveys.ClickAndGo().
            TestsTable.Rows[x => x.Title.Content == _TEST_NAME].Check.Click().
                Questions.ClickAndGo().
                SubMenu.AddQuestions.ClickAndGo().
                Category.Set(_QUESTION_CATEGORY_NAME).
                Filter.ClickAndGo().
                QuestionsTable.Rows[x => x.Question ==
_QUESTION_TEXT].Check.Click().
```

```
            AddToTestSurvey.ClickAndGo().
            Yes.ClickAndGo().
            QuestionsTable.Rows
               [x => x.Question.Content == _QUESTION_TEXT].Should.Exist();
      }

      [Test, Order(12)]
      public void StudentCanCompleteTheTest()
      {
         LoginAsSimpleUser(_STUDENT_LOGIN,
AppConfig.Current.DefaultPassword).
            MainMenu.MyCourses.ClickAndGo().
            CoursesTable.Rows[x => x.Course.Content ==
_COURSE_NAME].Course.ClickAndGo().
            MyTestsAndSurveys.ClickAndGo().
            TestsTable.Rows[x => x.Title.Content ==
_TEST_NAME].Title.ClickAndGo().
            Begin.ClickAndGo().
            True.Click().
            PreSubmit.Click().
            Submit.ClickAndGo().
            GoToContent.ClickAndGo().
            MyTestsAndSurveys.ClickAndGo().
            SubmissionsTable.Rows[x => x.Title.Content ==
_TEST_NAME].Should.Exist();
      }

      [Test, Order(13)]
      public void StudentCanUnenrollFromTheCourse()
      {
         LoginAsSimpleUser(_STUDENT_LOGIN,
AppConfig.Current.DefaultPassword).
```

```
            MainMenu.MyCourses.ClickAndGo().
            CoursesTable.Rows
              [x => x.Course.Content == _COURSE_NAME].Unenroll.ClickAndGo().
            Yes.ClickAndGo().
            CoursesTable.Rows
              [x => x.Course.Content == _COURSE_NAME].Should.Not.Exist();
        }
    }
}
```