Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Факультет інформатики
Кафедра інформатики

Магістерська робота

Освітній ступінь: магістр

На тему: «ХМАРНА СИСТЕМА MATHLEARNING: ІМПЛЕМЕНТАЦІЯ ПРОДУКТОВИХ ВИМОГ ТА ДОКУМЕНТАЦІЇ»

Виконав: студент 2 року навчання
Спеціальності
121 Інженерія програмного забезпечення
Скуратівський Максим Юрійович
Керівник
Малашонок Г. І.
доктор фіз.-мат. наук

Рецензент О. П. Жежерун

Магістерська робота захищена
з оцінкою_____
Секретар ЕК _____
«___» _____2023

Київ 2023

ЗМІСТ

АНОТАЦІЯ

Дана робота присвячена темі збору та документації вимог до програмного забезпечення, як важливого етапу розробки інформаційних систем. Розглянуто історію та визначення продуктових вимог, їх класифікацію, а також окреслено компетенції та якості бізнес-аналітика, як спеціаліста, який працює над вимогами до програмного забезпечення у командах розробки. Описано кожен етап життєвого циклу продуктових вимог, а також техніки та інструменти, які використовує бізнес-аналітик для роботи з продуктовими вимогами у проектах. Результатом практичної частини даної роботи є продуктова документація та опис вимог до хмарної системи MathLearning двома мовами: українською та англійською.

ВСТУП

Сьогодні робота над продуктовими вимогами є фундаментом кожного проекту з розробки програмного забезпечення. Багато компаній, попри роки досвіду роботи з IT- продуктами, все ще мають проблеми із розумінням, документуванням та управлінням вимог до IT-систем. Часто це призводить до розробки неактуальних та непрацюючих рішень, що супроводжується втратою часу, ресурсів та можливостей для компаній. Все більше замовників та фахівців з розробки IT-рішень розуміють, що комунікація – це ключ до актуальності та успішності продукту. Саме тому сьогодні часто у командах розробки окреме місце посідає фахівець, який тісно співпрацює із замовником, користувачами та командою, аби гарантувати, що продукт, над яким ведеться робота, є необхідним, функціональним та приноситиме додану вартість для кінцевого користувача.

Мінливість кон'юнктури ринку, глобалізація, війни та пандемії роблять свій внесок у вектор розвитку людства та призводять до змін у підходах до освітнього процесу. У відповідь, все частіше на ринку з'являються EdTech продукти, мета яких швидко пристосуватись до змін та пропонувати нові рішення для освітньої сфери. Вдалий підхід до збору, аналізу, документування та валідації вимог до інформаційних систем у EdTech сфері підвищує шанси на отримання якісного кінцевого продукту.

Відповідно до аналізу наукових робіт та публікацій, питанням імплементації продуктових вимог та документації для програмного забезпечення займалися такі науковці: М. Шоу, К. Вігерс, С. Робертсон, Б. Боем, І. Соммервіль, Д, Джексон, Л.М. Божуха, Ю. І. Грицюк, О. А. Нємова та інші.

Метою даної наукової роботи є створення продуктових вимог та документації до EdTech платформи MathLearning. Відповідно до мети, можна виділити наступні завдання: дослідити історію та роль продуктових вимог як елементу програмного забезпечення; проаналізувати сучасні техніки та

інструменти для збору, аналізу, документування та валідації продуктових вимог; зібрати, проаналізувати та задокументувати продуктові вимоги до хмарної системи MathLearning, використовуючи техніку User Story в рамках інструментів Jira та Confluence; проаналізувати місце освітніх IT-продуктів та можливості, які вони надають для навчання; провести порівняльний аналіз хмарної платформи MathLearning із іншими популярними освітніми продуктами.

Об’єктом дослідження є вимоги до програмного забезпечення та хмарна система MathLearning.

Предметом дослідження є продуктові вимоги та документація до хмарної системи MathLearning.

РОЗДІЛ 1. ПРОДУКТОВІ ВИМОГИ ЯК ЕЛЕМЕНТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Історія та визначення продуктових вимог

Аби краще зрозуміти, що означає поняття “продуктові вимоги” або “вимоги до програмного забезпечення”, варто спочатку звернутись до академічного тлумачного словника української мови, щоб дати визначення слова “Вимоги”. Серед доступних визначень цього терміну можна виокремити наступні: “Побажання, прохання, висловлене так, що не припускає заперечень”, “Норми, правила, яким хто-, що-небудь повинні підлягати”, та “Потреби, запити, які хто-, що-небудь має або ставить до когось, чогось”. Ці визначення чітко вказують на три дійові елементи: щось, що має підпадати під вимоги, хтось, хто має запит на вимоги і власне самі вимоги. Якщо об’єднати ці три елементи, то можна сказати, що вимоги допомагають встановити якісні параметри та характеристики для продуктів, систем чи сервісів, щоб забезпечити їхню відповідність потребам користувачів [1].

Сьогодні, певні види вимог використовуються в різних галузях, від програмного забезпечення до харчової промисловості. Вони виступають гарантом того, що результат праці буде реалізований таким чином, щоб задовольняти потребу кінцевого споживача або норми контролюючої інституції. Серед прикладів сфер та вимог чи норм, які поширені у цих сферах, можна виділити наступні:

- Інформаційні технології та програмне забезпечення: у цій галузі дуже поширене використання вимог до програмного забезпечення або продуктових вимог. Вони допомагають забезпечити, що розроблене програмне забезпечення відповідає потребам користувачів та бізнесу.
- Інженерія та будівництво: вимоги в інженерії використовуються для визначення технічних характеристик і вимог до конструкцій, будівель, систем енергопостачання, та інших об'єктів.

- Медицина та фармацевтика: у цих галузях вимоги визначають стандарти безпеки, якості та ефективності медичних препаратів, медичного обладнання, процедур та протоколів лікування.
- Фінанси та банківська справа: вимоги в цій галузі можуть включати правила та регуляторні вимоги до фінансової звітності, безпеки платіжних систем, захисту конфіденційної інформації клієнтів.
- Автомобільна промисловість: вимоги в автомобільній галузі можуть стосуватися екологічних стандартів, безпеки автомобілів, енергоефективності, функціональності електронних систем та інших аспектів автомобільного виробництва.
- Логістика та постачання: вимоги у цій галузі можуть стосуватися ефективності логістичних процесів, систем відстеження товарів, якості упаковки та транспортування, термінів доставки, тощо.
- Харчова промисловість: у цій галузі вимоги стосуються якості продуктів харчування, стандартів безпеки їжі, виробничих процесів, упаковки та маркування.
- Публічна безпека: у галузі публічної безпеки вимоги можуть визначати стандарти та протоколи для пожежної безпеки, безпеки громадських місць, систем відеоспостереження, тощо

У сфері розробки програмного забезпечення документування вимог завжди було важливим елементом. Продуктові вимоги у програмному забезпеченні - це опис того, що має включати продукт, який розробляється або планується до розробки. Це може бути документ, який містить детальні характеристики продукту, які можуть включати функціональні вимоги (тобто те, що продукт повинен робити) та нефункціональні вимоги - якісні характеристики програмного забезпечення (як от безпека, надійність та інші характеристики) [2].

У контексті програмного забезпечення, вимоги почали активно розглядатись у 1950-1960-х роках з появою комп'ютерних систем. В цей період розробники програмного забезпечення стали усвідомлювати необхідність систематичного підходу до встановлення та документування вимог для

успішного розвитку та управління проектами. Вимоги до програмного забезпечення стали ключовим етапом у процесі розробки програм, оскільки допомагали зрозуміти та формалізувати очікування та потреби користувачів [4].

Процес встановлення вимог у програмному забезпеченні пройшов шлях еволюції від неструктурованих описів до формалізованих специфікацій. Раніше, вимоги можна було виражати у формі нотаток чи через усну комунікацію. Проте, з плином часу та розширенням галузі стало очевидним, що необхідно певним чином стандартизувати процес встановлення вимог для досягнення більшої ясності та точності. Одним з ранніх підходів до специфікації та структуризації вимог була методика структурованого аналізу (Structured Analysis), яка була розроблена в 1970-х роках. Цей підхід використовувався для аналізу та документування функцій та потоків даних у системі. Застосування структурованого аналізу допомагало інженерам розібратись у потребах користувачів та створити формальну специфікацію вимог[4].

Інший важливий підхід, що з'явився у цей період - це методика прототипування (Prototyping). Цей підхід полягав у створенні прототипів програмного забезпечення, що дозволяли користувачам тестувати та зрозуміти функціональні вимоги. Це дало можливість розробникам отримувати більш ранній зворотній зв'язок від користувачів та коригувати вимоги згідно з їхніми потребами [4].

У 1970-х роках були розроблені формальні мови та методики, такі як VDM (Vienna Development Method) та інші, які дозволяли формалізувати та математично описувати вимоги до програмного забезпечення. Ці підходи були корисними для верифікації вимог та забезпечення точності і якості програм [4].

Згодом, у 1990-х роках з'явилися методики, які акцентували на взаємодії з користувачами, такі як Crystal, Scrum, екстремальне програмування (Extreme Programming - XP) та інші. Ці методики звертали увагу на активну участь користувачів у визначенні вимог та забезпеченні взаєморозуміння між розробниками та клієнтами [4].

Сьогодні з розвитком Agile-методологій та ітеративних підходів до розробки програмного забезпечення, акцент змістився на постійне оновлення та документування вимог протягом всього процесу розробки. За останні роки з'явилися нові підходи до документування вимог, такі як використання "User Stories", "Use Cases" та інші. Ці підходи спрямовані на більш гнучке та комунікативне визначення та документування вимог з урахуванням мінливості потреб користувачів.

Продуктові вимоги є важливим інструментом для забезпечення якості продукту, так як вони дозволяють визначити, що саме очікується від продукту, перед тим, як він буде розроблений. Вони дозволяють забезпечити однакове розуміння між замовником продукту та командою розробників щодо цілей функціоналу в рамках продукту [3]. Серед аспектів, на які впливають продуктові вимоги під час розробки програмного забезпечення, можна виділити наступні:

1. Керування проектом: вимоги визначають, що повинен робити продукт, тому вони є основою для планування та керування проектом розробки програмного забезпечення. Якщо базові вимоги неякісно визначені або не задокументовані належним чином, це може призвести до затримок у графіку розробки, витрат на виправлення вже написаних частин проекту та незадоволеності з боку клієнта.
2. Взаємодія з клієнтом: правильно складені продуктові вимоги допомагають встановити зв'язок з клієнтом та зрозуміти його потреби. Вони дозволяють розробникам зібрати потрібну інформацію та визначити, яким чином продукт має бути розроблений, щоб задовольнити потреби клієнта.
3. Якість: продуктові вимоги допомагають визначити критерії, за якими можна оцінювати якість розробленого продукту, що дозволяє оцінювати ефективність та результативність команди розробки.
4. Керування ризиками: вимоги допомагають ідентифікувати ризики та визначити заходи для їх зменшення на ранніх стадіях.
5. Комунікація: вимоги допомагають забезпечити ефективну комунікацію всередині команди, між розробниками тестувальниками і менеджерами

проекту. Це дозволяє уникнути непорозумінь та помилок, що в результаті можуть призвести до збільшення витрат на розробку продукту [3].

З появою Agile-методологій розробки програмного забезпечення, продуктові вимоги почали розглядатись як динамічний процес, що може змінюватися в залежності від змін у потребах користувачів та кон'юнктури ринку.

1.2 Класифікація вимог до програмного забезпечення

Класифікація вимог до програмного забезпечення - це процес групування та категоризації вимог з метою легшого розуміння, управління та виконання процесу розробки програмного продукту. Правильно зібрані та задокументовані вимоги відповідно до класифікації допомагають визначити специфічні характеристики та властивості, які повинні бути відображені у програмному забезпеченні. Згідно з Карлом Вігерсом, вимоги до програмного забезпечення можна поділити на наступні: бізнес-вимоги, користувацькі вимоги, та програмні вимоги [2].

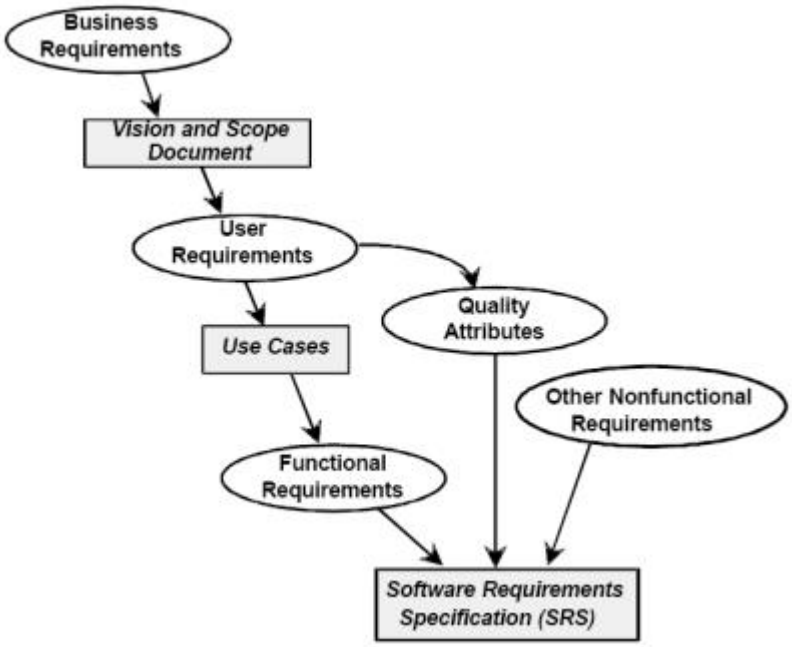

Рисунок 1. Класифікація вимог до програмного забезпечення [2]

Бізнес-вимоги - це вимоги, які визначають потреби та очікування організації або бізнес-середовища, в якому буде використовуватися програмне забезпечення. Ці вимоги визначаються з метою покращення бізнес-процесів, досягнення стратегічних цілей та вирішення проблем організації [3].

Бізнес-вимоги є важливою основою для розробки програмного забезпечення, оскільки вони допомагають зорієнтуватись у потребах бізнесу, визначити основні функції та можливості програми, а також оцінити успішність проекту з точки зору бізнес-цілей. Розуміння бізнес-вимог і їх належна реалізація у програмному забезпеченні можуть сприяти досягненню успіху та задоволення потреб клієнта або організації. Бізнес-вимоги визначають потреби проекту та вимоги до його успіху, зважаючи на цільову аудиторію. [5]

Користувацькі вимоги, які також відомі як user requirements – це вимоги, які описують очікування, потреби та бажання користувачів щодо функціональності та характеристик програмного забезпечення. Це вимоги, які встановлюються на основі взаємодії користувача з продуктом та його потреб до виконанні власних завдань [3].

Вимоги, отримані від користувачів, грають важливу роль у процесі розробки програмного забезпечення, оскільки вони допомагають розробникам зрозуміти, як програма повинна взаємодіяти зі своїми користувачами та задовольняти їх потреби. Вони служать основою для проектування і розробки інтерфейсу користувача, функціональності та інших аспектів програмного продукту. Користувацькі вимоги описують, як програмне забезпечення повинно взаємодіяти зі своїми користувачами та яким чином задовольняти їх потреби. Вони визначають, що програма має робити, які функції мають бути доступні користувачам, які дії вона має підтримувати, а також, які характеристики та властивості повинні бути присутні для задоволення очікувань користувачів [24].

Користувацькі вимоги допомагають розробникам глибше зрозуміти потреби, очікування і цілі користувачів програмного забезпечення. Вони дозволяють визначити, яким чином будуть оцінюватись успішність та ефективність програмного продукту з точки зору користувачів. Вони

допомагають визначити, які функції та можливості має включати продукт, а також які можливості є опціональними або поза обсягом проекту. Концепція користувацьких вимог ставить користувачів у центр розробки програмного забезпечення. Вона сприяє створенню інтуїтивно зрозумілого і зручного для продукту, що підтримує потреби користувачів, дозволяє їм досягати своїх цілей та має зручний та інтуїтивний інтерфейс [12; 13].

Окремим видом вимог до програмного забезпечення відзначають програмні вимоги, які також відомі як Solution Requirements. Програмні вимоги - це тип вимог, які визначають специфічні вимоги щодо конкретного рішення або системи, яка буде розроблене для вирішення конкретної проблеми або задачі. Вони концентруються на деталях і технічних характеристиках рішення. Програмні вимоги визначаються після або впродовж аналізу бізнес-вимог та користувацьких вимог, і відображають технічні, функціональні та нефункціональні характеристики самого рішення. Вони описують, яким чином система повинна бути розроблена, впроваджена та підтримувана для виконання потреб бізнесу та користувачів. Програмні вимоги можна поділити на функціональні та нефункціональні вимоги [2].

Функціональні вимоги визначають, які функції та задачі повинна виконувати система або програмне забезпечення. У даному типі вимог можуть бути описані різні аспекти функціональності, такі як вхідні та вихідні дані, обробка даних, логіка бізнес-процесів, взаємодія з користувачем та інші функціональні елементи системи. Функціональні вимоги визначають основні архітектурні рішення програмного забезпечення і служать основою для розробки та тестування системи [14].

Як приклад, можна навести функціональні вимоги до електронної платформи для навчання на віддаленій основі:

1. Вчителі можуть створювати та публікувати навчальні матеріали, включаючи уроки, завдання та тести.
2. Учні можуть реєструватись та входити в систему для доступу до навчальних ресурсів.

3. Система може надавати календар занять та розклад.
4. Учні можуть переглядати відеоуроки, читати електронні підручники та виконувати завдання.
5. Вчителі можуть оцінювати завдання та давати зворотний зв'язок учням.
6. Система може надавати звіти про активність учнів та їх прогрес у навчанні.
7. Система може надавати сертифікати та звіти про виконання курсів.

Як було зазначено, іншим видом програмних вимог є нефункціональні вимоги. Нефункціональні вимоги є важливою складовою частиною специфікації програмного забезпечення і визначають аспекти якості, вимоги до продуктивності та інші характеристики системи, які не стосуються безпосередньо функцій програми. Основна відмінність між функціональними і нефункціональними вимогами полягає в тому, що нефункціональні вимоги описують як система має працювати, а не що саме вона має робити [2].

Основні категорії нефункціональних вимог включають:

- Ефективність - вимоги до продуктивності та швидкості відповіді системи, такі як час відгуку, швидкість обробки даних, обсяг пам'яті, оптимізація алгоритмів тощо.
- Надійність - вимоги до стабільності, доступності та надійності системи, такі як час безвідмовної роботи, відновлення після збоїв, захист від помилок та відновлення даних.
- Безпека - вимоги до захисту даних, обмеження доступу до системи, аутентифікації та авторизації користувачів, шифрування тощо.
- Сумісність - вимоги до взаємодії з іншими системами, стандартами та протоколами [15].

Отже, вимоги до програмного забезпечення можна класифікувати за різними аспектами, що допомагає систематизувати і структурувати вимоги. У свою чергу класифікація сприяє кращому розумінню та керуванню процесом розробки програмного забезпечення.

1.3 Бізнес-аналітик та його роль у розробці програмного забезпечення

У проектах з розробки програмного забезпечення беруть участь різні спеціалісти з відповідними навичками та експертизою. Це обумовлено складністю самого процесу розробки програмного забезпечення та необхідністю врахування різноманітних аспектів проекту для його успішної реалізації. Сьогодні у більшості проектів з розробки програмного забезпечення, для визначення та оформлення вимог передбачена роль бізнес-аналітика. Вони вивчають потреби бізнесу та користувачів, проводять аналіз вимог і визначають цілі проекту, аналізують ринок і конкурентну ситуацію, визначають функціональні і нефункціональні вимоги до програмного забезпечення. Бізнес-аналітики виступають посередниками між бізнесом і розробниками, чим допомагають поєднати потреби і точки зору на процес з обох сторін.

Роль бізнес-аналітиків в командах розробки програмного забезпечення почала активно розвиватись і набирати популярність з початку 2000-х років. До цього часу, в розробці програмного забезпечення, багато уваги приділялось технічній стороні проекту, а менше - розумінню бізнес-потреб та комунікації із замовниками. Це часто призводило до непорозумінь, помилок та непридатного програмного забезпечення, яке не відповідало потребам бізнесу. З появою бізнес-аналітиків у командах розробки програмного забезпечення, ця роль стала виконувати функцію посередника між бізнес-замовниками та розробниками. Бізнес-аналітики почали спеціалізуватись на розумінні бізнес-процесів та потреб, а також на перекладі цих потреб у зрозумілі для розробників вимоги до програмного забезпечення [21].

Для ефективного виконання обов’язків, які покладають на бізнес-аналітиків, дані спеціалісти мають володіти відповідним набором знань та навичок. Бізнес-аналітик є певною мірою “мостом” між замовниками та командою розробки, що вимагає від нього розуміння обставин, мотивів та стану справ з обох сторін [22].

Насамперед, бізнес-аналітик повинен мати розуміння принципів функціонування бізнесу, галузевих особливостей та бізнес-процесів, в якому працює компанія або проект. Дані навички допомагають їм визначати потреби бізнесу та перетворювати їх на вимоги до програмного забезпечення [21].

Ще однією компетенцією, яка важлива для бізнес-аналітика – є сильні аналітичні навички, зокрема здатність аналізувати бізнес-процеси, дані та запити стейкхолдерів, виявляти тенденції та залежності, аби пропонувати відповідні течнічні- та бізнес-рішення [12].

Бізнес-аналітик повинен мати відмінні комунікаційні навички, які дозволяють їм ефективно спілкуватися з бізнес-замовниками, розробниками та іншими зацікавленими сторонами проекту. До таких навичок можна віднести: вміння слухати, задавати запитання, пояснювати складні концепції для різних аудиторій та узгоджувати різні точки зору [2].

Хоча бізнес-аналітик не є безпосередньою технічною роллю, даний спеціаліст повинен мати базові технічні знання, що стосуються процесу розробки програмного забезпечення. У бізнес-аналітика повинне бути розуміння щодо обмежень та можливостей технологій, які будуть використовуватись у реалізації програмного продукту [2].

Так як бізнес-аналітики тісно співпрацюють з проектними менеджерами, такі навички, як планування, оцінка ризиків, контроль термінів та ресурсів є також важливими для повсякденної роботи [2].

Отже, роль бізнес-аналітиків є критично важливою в процесі розробки програмного забезпечення, оскільки вони допомагають зрозуміти потреби бізнесу, встановити правильні пріоритети, уточнити вимоги та забезпечити взаєморозуміння між різними сторонами проекту. Це сприяє покращенню якості та реалізації програмного забезпечення, що відповідає потребам бізнесу та користувачів. Водночас важливо, щоб всі члени команди проекту розуміли продуктові вимоги і прагнули до їх виконання для успішного завершення проекту [6].

Висновки до розділу 1

У першому розділі було розглянуто поняття вимог у програмному забезпеченні, їх роль та місце у процесі розробки. Було наведено структуру класифікації вимог до програмного забезпечення та опис кожного виду продуктових вимог: бізнес-вимоги, користувацькі та програмні вимоги. Було визначено спеціаліста, який займається збором та документацією продуктових вимог, а також наведено компетенції та навички, якими повинен володіти даний спеціаліст.

РОЗДІЛ 2. ПРОДУКТОВІ ВИМОГИ ЯК СКЛАДОВА ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Життєвий цикл та управління вимогами до програмного забезпечення

Розробка програмного забезпечення вимагає чіткого розуміння вимог, які ставляться до системи. Успішне визначення, документування та управління вимогами є критично важливими для розробки високоякісного програмного забезпечення, яке відповідає потребам користувачів та замовників. Для опису процесу управління вимогами впродовж усієї тривалості проекту використовується таке поняття як "життєвий цикл вимог до програмного забезпечення" [2]. Життєвий цикл вимог охоплює такі етапи:

1. *Збір вимог* передбачає виявлення, осмислення та запис вимог від зацікавлених сторін. Вимоги можуть бути зібрані шляхом спілкування з клієнтом, аналізу доступної документації, проведення опитувань та інтерв’ю потенційних користувачів [18].
2. На етапі *аналізу вимог* проводиться детальний аналіз і розуміння зібраних вимог. Під час аналізу вимоги класифікуються, структуруються і перевіряються на взаємозв'язок та суперечності [15].
3. Під час *документування вимог,* бізнес-аналітик описує їх у визначеному форматі у відповідно структурованому документі [20].
4. Останнім етапом є *валідація вимог*. На цьому етапі перевіряється, чи відповідають зібрані вимоги потребам та очікуванням зацікавлених сторін, а також виконується перевірка правильності і повноти вимог [26].

Життєвий цикл вимог є важливим процесом для забезпечення якості та успішної розробки програмних продуктів, оскільки він дозволяє системно створювати вимоги - від загальної концепції того, що має бути зроблено і до реалізації у продукті.

Поряд із життєвим циклом продуктових вимог, важливим елементом також є менеджмент вимог до програмного забезпечення. Управління вимогами (або менеджмент вимог) - це процес планування та контролю реалізації вимог протягом усього процесу розробки програмного продукту [23]. На рис. 2 зображений перелік основних активностей у розрізі управління вимогами, які розподілені за чотирма категоріями: управління змінами, контроль версій, відстеження статусу та трасування вимог [14].

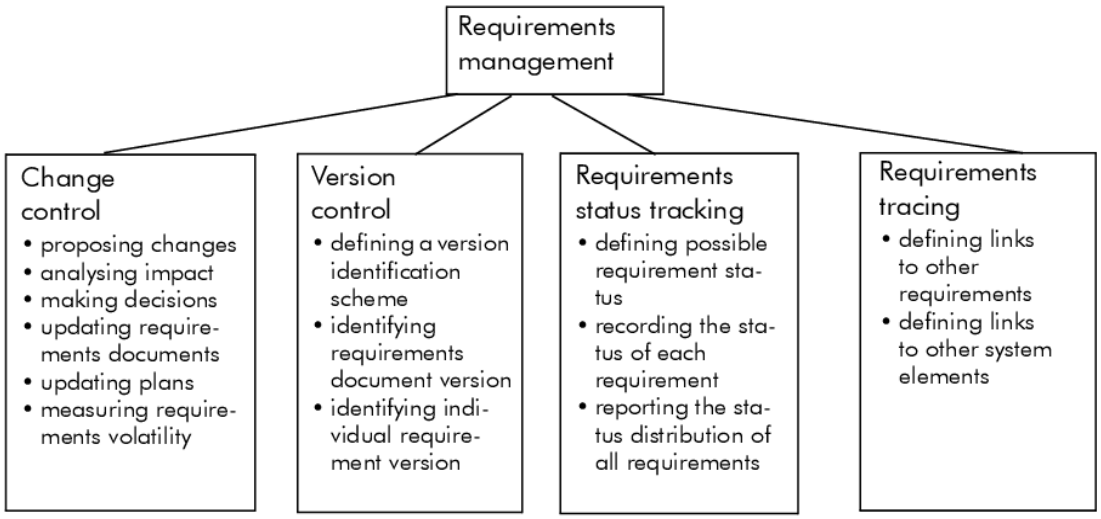

Рисунок 2. Категорії та активності управління вимогами [2]

Якщо детальніше розглядати кожну із наведених категорій, то можна виділити наступне:

- Управління змінами - це процес контролю та керування змінами, що виникають у вимогах протягом розробки програмного забезпечення. Цей процес включає ідентифікацію зміни, аналіз впливу на проект, затвердження з клієнтом, оновлення проектної документації та оцінка волатильності вимог.
- Контроль версій вимог - ця категорія охоплює активності зі зберігання, відстеження та керування різними версіями вимог або наборів вимог.
- Відстеження статусу - включає у себе визначення можливих статусів вимог, призначення статусу кожній вимозі та аналіз розподілу статусів вимог. Статуси допомагають забезпечити прозорість і контроль над

реалізацією вимог, а також виявленням частин процесу, які потребують коригування ресурсів.

- Трасування вимог - складається із пошуку зв’язків із іншими вимогами, а також із іншими елементами системи. Трасування дозволяє встановити, від яких джерел походять вимоги, як вони впливають на інші вимоги чи елементи системи. Правильне трасування допомагає забезпечити цілісність та узгодженість вимог, а також спрощує їх зміни [2].

Загалом, управління вимогами є необхідним елементом у розробці програмного забезпечення, оскільки воно “страхує” проект від того, що зусилля, витрачені на розробку вимог не будуть марними. Ефективне управління вимогами зменшує прірву очікувань між клієнтом та командою розробки завдяки легкій можливості у інформуванні клієнта про статус вимог протягом усього процесу розробки.

Протягом усього часу реалізації проекту, не лише бізнес-аналітик, а й інші члени проектної команди беруть активну участь у роботі над і з вимогами.

Розробники програмного забезпечення виконують важливу роль у реалізації вимог. Використовуючи описані вимоги, розробники розробляють архітектуру та дизайн програмного продукту.

Тестувальники перевіряють, чи відповідає програмне забезпечення вимогам та чи воно працює належним чином. Тестувальники розробляють тестові сценарії на основі вимог, виявляють можливі відхилення від вимог та взаємодіють із командою для узгодження функціоналу із вимогами.

Менеджери проекту координують роботу з вимогами, беруть безпосередню участь у їх пріоритезації та управлінням ресурсів для реалізації наявних вимог.

2.2 Техніки та інструменти які використовуються при роботі з продуктовими вимогами

На кожному етапі роботи з продуктовими вимогами (збір, аналіз, опис та валідація), бізнес-аналітики використовують відповідний набір технік, що допомагають їм ефективно структурувати та опрацьовувати отриману інформацію. Конкретний набір технік, які використовуються на кожному етапі, може залежати від методології розробки, контексту проекту та потреб команди розробки чи замовника [13]. Далі буде розглянутий перелік технік по кожному етапу роботи з вимогами.

Збір продуктових вимог - на даному етапі зазвичай використовуються техніки, які допомагають бізнес-аналітику виявляти та збирати різноманітну інформацію та дані для формування повної картини щодо потреб та очікувань зацікавлених сторін. Найбільш поширені та ефективні техніки включають наступні:

- Інтерв'ю з зацікавленими сторонами. Бізнес-аналітик проводить співбесіди з різними зацікавленими сторонами проекту, такими як клієнти, користувачі, менеджери, експерти та інші. Це дозволяє отримати важливу інформацію про їх потреби, очікування, проблеми та бажання.
- Спостереження. Часто, спостерігаючи роботу користувачів в реальному часі, бізнес-аналітик може проаналізувати як саме побудовані бізнес-процеси.
- Фокус-групи. Відбуваються шляхом організації групових дискусій з представниками різних зацікавлених сторін для обговорення вимог, спільного пошуку рішень та збирання цінного фідбеку.
- Анкетування та опитування. Це методи, за допомогою яких бізнес-аналітик може зібрати вимоги шляхом створення структурованих опитувальних анкет або проведення інтерв'ю великої кількості людей.
- Прототипування. Створення прототипів або демонстраційних моделей допомагає виробити краще розуміння вимог і сприяє активній участі

зацікавлених сторін у процесі збору вимог. Прототипи можуть служити вихідним матеріалом для обговорення та валідації вимог [19].

Аналіз продуктових вимог - етап у роботі з вимогами, який спрямований на виявлення, уточнення та узгодження вимог, а також на визначення внутрішньої структури системи та зв'язків між її компонентами. Процес аналізу вимог складається із набору різних підходів, але загальноприйнятий процес містить у собі перелік сталих практик. Насамперед, бізнес-аналітик розбиває зібрані вимоги на групи та класифікує відповідно до їх характеристик та призначення. Це допомагає зрозуміти загальну картину вимог та виявити потенційні залежності та конфлікти між ними. Після цього, часто використовується метод моделювання бізнес-процесів, який супроводжується використанням EPC-діаграм або BPMN-діаграм, для зображення поточних та цільових бізнес-процесів. Це допомагає виявити вимоги до функціональності системи та зрозуміти взаємозв'язки між компонентами та акторами у системі. За умови комплексності системи, бізнес-аналітик може приходити до методу функціональної декомпозиції, розбиваючи складні підсистеми на більш прості елементи. Він також об'єднує схожі вимоги за приналежністю до схожих функціональних модулів. Фінальним етапом зазвичай є проведення перевірки вимог на наявність суперечностей, неузгодженостей або неповностей. Якісна перевірка на суперечності забезпечує якість вимог та сприяє зменшенню кількості помилок вже під час розробки програмного забезпечення [25].

Документування вимог - це процес опису зібраних та проаналізованих вимог у чітко визначеному форматі. Основна мета документування - створення зрозумілого і однозначного опису вимог, який буде використовуватись командою розробки як вихідний матеріал для подальшого проектування, розробки та тестування системи. Окрім того, вихідна документація зазвичай використовується при ознайомленні нових членів команди із проектом, а також при розширенні системи та розробки нових складових [20]. Найбільш поширеними техніками, за допомогою яких описують вимоги, є наступні:

- User Stories (Історії користувачів) - це формат опису вимог, який широко використовується у методології Agile. User Stories зосереджуються на потребах та цілях користувачів системи. Основна форма запису User Story має наступний формат: "Як [тип користувача], я хочу [функціональність], щоб [користь або ціль]". Де "тип користувача" може бути або роллю в системі, або конкретною особою "функціональність" описує очікувану дію або функцію системи, а "користь або ціль" пояснює, які переваги чи результати очікуються від цієї функціональності. Історії користувачів зазвичай супроводжуються так званими Acceptance Criteria (критеріями прийняття), які розкривають більше технічних деталей необхідного функціоналу [9].
- Use case (сценарій використання) - це техніка опису вимог, яка використовується для моделювання взаємодії між системою та її акторами (користувачами або зовнішніми системами). Use case описує послідовність подій, які відбуваються з користувачем та системою при їх взаємодії [11]. Структура Use case включає включає такі елементи:
    - *Назва*: ідентифікує конкретний сценарій використання.
    - *Актори*: описують особу або сутність, які взаємодіють з системою.
    - *Опис*: детальний опис сценарію використання, включаючи контекст, послідовність дій та взаємодію між акторами та системою.
    - *Передумови*: умови або стан системи, які повинні бути задоволені перед початком сценарію.
    - *Результат*: очікуваний результат або стан системи після успішного виконання сценарію.
    - *Виключення*: вказує на можливі виключні ситуації або помилки, які можуть виникнути під час виконання сценарію.

Кожна з цих технік опису має свої переваги та недоліки і може бути використана в різних ситуаціях залежно від потреб бізнесу та характеристик проекту. Наприклад, техніка Use Case добре підходить для великих та складних систем, тоді як User Stories краще пасує для менших проектів з короткими

ітераціями розробки. Ці техніки можуть використовуватись окремо або в комбінації одна з одною. Вибір певної техніки залежить від вимог клієнта, проекту та вимог розробника продукту [7].

Наступний етап роботи бізнес-аналітика після опису вимог є їх валідація. Валідація вимог - це процес перевірки та підтвердження, що вимоги до системи є правильними, повними, зрозумілими та відповідають потребам і очікуванням зацікавлених сторін. Валідація вимог включає у себе наступні елементи:

1. Перевірка на відповідність бізнес-цілям: Оцінка того, наскільки вимоги відповідають бізнес-цілям та стратегії компанії.
2. Перевірка на реалізованість: Оцінка того, наскільки вимоги можуть бути реалізовані в контексті проекту, яка проводиться за допомогою аналізу технічної можливості втілення вимог та оцінку ризиків, пов'язаних з їх реалізацією.
3. Затвердження та погодження: Забезпечення отримання згоди всіх зацікавлених сторін на валідність вимог. На практиці це відбувається шляхом затвердження документів вимог та підписання угод [26].

Підсумовуючи, бізнес-аналітики використовують різноманітні техніки при роботі з вимогами. Кожна з цих технік відіграє свою роль у процесі роботи з вимогами, допомагаючи бізнес-аналітику розуміти потреби стейкхолдерів, структурувати та документувати вимоги, валідувати їх на відповідність та забезпечувати їх виконання. Використання відповідних технік є важливим аспектом успішної роботи бізнес-аналітика та сприяє якісному та ефективній розробці програмного забезпечення [7].

Важливою складовою у роботі з продуктовими вимогами є інструменти, які використовує бізнес-аналітик для документування та роботи з вимогами. Вдалий вибір інструментів сприяє покращенню у комунікації між бізнес-аналітиком, командою розробки та замовником. Інструменти, що відповідають масштабу та складності проекту, допомагають забезпечити точність, зрозумілість та прозорість вимог, що у свою чергу слугує підвищенню ефективності роботи команди проекту. Інструменти для документування та

управління вимогами можуть бути досить різноманітними і залежать від потреб та можливостей конкретної команди розробників [28]. Однак, серед найбільш популярних інструментів можна виділити наступні:

1. Microsoft Excel або Google Sheets - це табличні процесори, які можуть бути використані для створення простих таблиць з вимогами та їх атрибутами, такими як пріоритет, статус, тощо.
2. Microsoft Word або Google Docs - це текстові процесори, які можуть бути використані для детального опису вимог та їх атрибутів у форматі документу [27].
3. Atlassian Jira - це популярна платформа для управління проектами, яка має функціонал для документування та відстеження статусу вимог [33].
4. IBM Rational DOORS - це інструмент для управління вимогами, який дозволяє створювати, організовувати та аналізувати вимоги великих проектів [32].
5. Confluence - це хмарний інструмент, який має у собі зручний текстовий процесор, що може використовуватися для створення та редагування документів з вимогами [33].
6. Trello - це інструмент для управління проектами, який дозволяє створювати картки з вимогами та переміщувати їх між колонками в залежності від стану вимог [34].
7. ReqView - це інструмент для управління вимогами, який дозволяє створювати, організовувати та відстежувати вимоги, а також генерувати звіти та документацію [35].

Ці інструменти можуть бути використані як окремо, так і в поєднанні з іншими інструментами для роботи з продуктовими вимогами протягом повного циклу розробки програмного забезпечення. Важливо вибирати інструмент, який найкраще підходить для потреб проекту та забезпечує ефективне управління вимогами.

Зважаючи на те, що інструменти для документування та управління

вимогами можуть відрізнятися за своїм функціоналом, можна провести порівняльний аналіз, виходячи з наступних критеріїв:

1. Функціональність: кількість та якість функцій, що надає інструмент, включаючи можливість додавати вимоги, встановлювати пріоритети, відстежувати стан вимог, редагувати та експортувати вимоги.
2. Ціна: вартість користування інструментом або його планів підписки.
3. Складність навчання: складність освоєння та користування інструментом.
4. Інтеграція: можливість інтеграції з іншими інструментами та сервісами.
5. Підтримка: якість та швидкість підтримки користувачів.

Нижче наведена порівняльна таблиця за критеріями функціональності, ціни, складності навчання, інтеграції з іншими сервісами та підтримки користувачів:

Інструмент	**Функціональність**	**Ціна**	**Складність**	**Інтеграція**	**Підтримка**
Microsoft Excel/ Google Sheets	Середня	Безкоштовно	Низька	Обмежена	Середня
Microsoft Word/ Google Docs	Висока	Безкоштовно	Низька	Обмежена	Середня
Atlassian Jira	Висока	Середня	Середня	Широка	Висока
Confluence	Висока	Середня	Середня	Широка	Висока
Trello	Середня	Безкоштовно	Низька	Широка	Середня
IBM DOORS	Висока	Висока	Складна	Широка	Висока
ReqView	Висока	Середня	Середня	Широка	Висока

Таблиця 1. Порівняльна характеристика інструментів для роботи з вимогами [32; 33; 34; 35].

Як можна побачити, якості кожного із наведених інструментів різняться відповідно до виділених критеріїв. Кожен проект може мати свої унікальні умови, команду розробки та особливості спілкування із зацікавленими сторонами, тому важливо вибрати інструменти, які найкраще відповідають контексту та специфічним потребам проекту.

2.3 Робота з продуктовими вимогами в Agile-проектах

Сучасний світ характеризується швидким темпом змін та постійною еволюцією. Технології, бізнес-процеси, ринки та потреби споживачів надзвичайно мінливі та турбулентні. У такому середовищі, успішність людей і організацій залежить від їх здатності швидко адаптуватись та пристосовуватись до нових реалій. Так, і в сфері розробки програмного забезпечення дійшли до розуміння, що підхід до проектів має бути гнучким та пристосовуватись до змін. Результатом цього розуміння стала поява Agile-підходу [8]. Agile - це набір принципів та практик, спрямованих на гнучку та ітеративну розробку продуктів, яка дозволяє ефективно відповідати на зміни у потребах клієнтів і швидко адаптуватись до мінливої кон'юнктури ринку. Поява Agile-методології стала реакцією на недоліки традиційного waterfall-підходу, який передбачав чіткі рамки для кожного етапу у циклі розробки програмного забезпечення і обмежену можливість змінювати вимоги після початку фази розробки [10]. Далі будуть наведені особливості роботи з продуктовими вимогами у Agile-проектах.

Насамперед потрібно виділити такий аспект, як залученість клієнтів. Тісна співпраця з клієнтами у проектах з розробки програмного забезпечення завжди підвищує шанси на успіх кінцевого продукту. Це стосується як Agile, так і Waterfall-проектів, але основна відмінність між двома підходами полягає в термінах залучення клієнта. У Waterfall-проектах замовники зазвичай беруть активну участь на етапі збору, документування та валідації вимог, а також під час тестування, коли функціонал вже реалізований. Однак на етапі розробки зазвичай клієнти не є залученими у верифікацію функціоналу, що ускладнює адаптацію проекту до мінливих потреб користувачів. У Agile-проектах клієнти є постійно залученими протягом усієї тривалості проекту і на кожному його етапі. Так як у Agile-проектах планування скоупу робиться для кожної ітерації, то у клієнта або інших членів команди є можливість внести зміни у скоуп ітерації, враховуючи усі нові аспекти ринку або вимог, які могли з’явитись протягом

попередньої ітерації. Відповідно у проектної команди також з'являється вікно можливостей для ретроспективи попереднього циклу розробки і комунікації з клієнтом щодо потенційних змін у вимогах. Також вимоги, які у Agile-проектах часто описані методом User Story та зазвичай менш детальні, ніж традиційні функціональні вимоги, потребують залученості клієнта під час проектування та розробки рішень. [16].

Іншим важливим аспектом, який має свої особливості у Agile-проектах - це підхід до документації. Оскільки у waterfall-проектах розробники мало взаємодіють із замовниками після початку девелопменту, вимоги та документація мають дуже детальною та чітко визначати поведінку системи, потоки даних, інтерфейс користувача та інші складові продукту. Натомість проекти із гнучким підходом до розробки, дозволяють документувати вимоги менш детально, так як тісна співпраця клієнтів із командою розробки відкриває можливість постійно валідувати результат через комунікацію та презентації. Це не лише дозволяє більш надійно валідувати актуальність розробленого функціоналу, але й зменшувати витрати ресурсів на документування на початкових етапах проекту [17].

Важливий аспект роботи з вимогами, який присутній у Agile-методологіях, на противагу іншим - це беклог (Backlog) продукту. Беклог - це увесь набір нереалізованих задач у форматі User Story або дефектів продукту (багів), які наявні у проекті на певний момент часу. Беклог є динамічним документом, який змінюється та розширюється протягом усього проекту. Загальноприйнято, що проект повинен мати лише один беклог, який виступає інструментом для планування, пріоритезації та управління вимогами продукту. Він допомагає команді розробки пріоритезовувати вимоги і обирати, які задачі будуть присутні у наступних ітераціях, а які можна відкласти. Пріоритизація вимог, використовуючи беклог, допомагає команді та замовнику зосередитися на найважливіших вимогах та забезпечує прозорість та спільне розуміння між усіма учасниками проекту щодо того, що потрібно реалізувати [2].

Agile-проекти вимагають тих самих типів вимог та зусиль щодо їх створення та реалізації, що й інші підходи до розробки програмного забезпечення. Бізнес-аналітику так само потрібно збирати вимоги від замовника та потенційних користувачів, аналізувати зібраний матеріал, документувати вимоги з відповідним рівнем деталізації та валідувати їх щодо бізнес-цілей проекту. Однак важлива відмінність полягає у розподілі термінів цієї роботи. Як правило, на початку agile-проектів створюються високорівневі вимоги у формі історій користувачів, що дозволяє почати планування, пріорітезування та розробку функціоналу ще до повного завершення роботи з продуктовими вимогами. Цей аспект часто може бути вирішальним для деяких бізнес-вимог, коли потрібно показати результат зацікавленим особам якомога швидше для отримання додаткового фінансування. Як показано на малюнку 3, робота з вимогами у Agile-проектах розподіляється на певні ітерації і деталі кожної User Story можуть додатково уточнюватись навіть під час розробки вимог відповідної ітерації. У такому форматі, робота з вимогами проводиться протягом усього проекту і може вестись навіть незадовго до випуску продукту. Водночас важливо виявити ключові нефункціональні вимоги на ранніх стадіях розробки, аби архітектуру системи можна було спроектувати для досягнення критичних показників продуктивності, зручності використання та доступності [2].

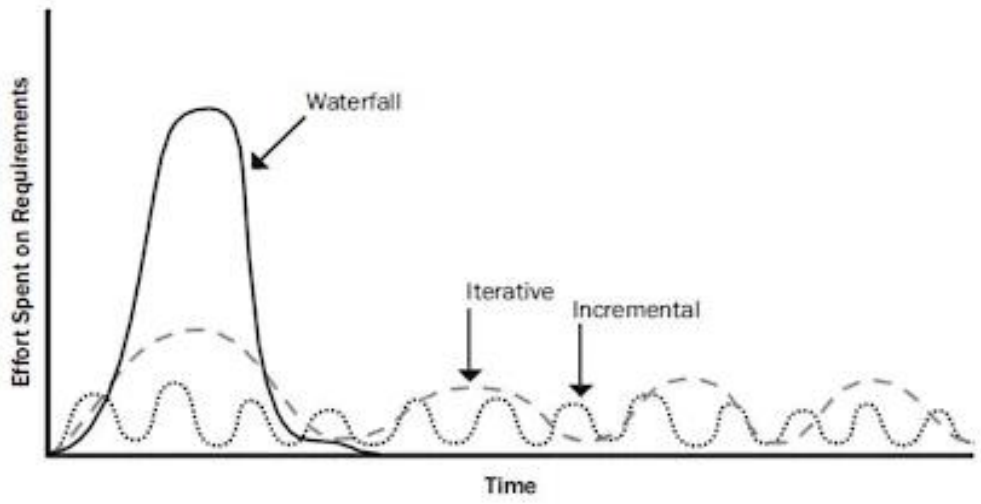

Рисунок 3. Розподіл витрати ресурсів на роботу з вимогами відповідно до типу проекту [2]

Висновки до розділу 2

Продуктові вимоги відіграють ключову роль у процесі розробки програмного забезпечення. У кожному проекті вимоги проходять через чітко визначений життєвий цикл, який включає в себе збір, аналіз, опис та документування і валідацію вимог. Правильна та послідовна робота з вимогами дозволяє забезпечити високу якість і відповідність продукту до потреб користувачів у майбутньому. На кожному етапі життєвого циклу продуктових вимог, бізнес-аналітики використовують відповідний набір технік та інструментів, аби виконувати свою роботу ефективно. Набір технік та інструментів має відповідати контексту та потребам кожного окремого проекту. Коректно обрані техніки та інструменти допомагають бізнес-аналітику чітко структуровувати та описувати вимоги, цим самим забезпечуючи зрозумілість та узгодженість між командою та клієнтом [24].

Робота з вимогами в Agile-проектах передбачає гнучкий підхід до розробки, де вимоги можуть додаватись та оновлюватись протягом усього проекту. Основні особливості включають гнучкість у плануванні та пріоритезації, ітеративність та активна комунікація між командою проекту, замовником та користувачами. Agile набув популярності через свою здатність швидко пристосовуватись до змін на ринку через можливість швидкого перегляду вимог і цим самим почав створювати більше доданої вартості та зниження кількості ризиків для замовників.

РОЗДІЛ 3. ІМПЛЕМЕНТАЦІЯ ПРОДУКТОВИХ ВИМОГ ТА ДОКУМЕНТАЦІЇ ДО ХМАРНОЇ СИСТЕМИ “MATHLEARNING”

3.1 Хмарна система “MathLearning” як приклад перспективного EdTech проекту

Сьогодні освіта є однією із найважливіших складових сталого розвитку суспільства. Високоякісна та актуальна освіта забезпечує людей необхідними знаннями, навичками та компетенціями для конфронтації викликів, які постають перед людьми у наш час. Із розвитком технологій все більше спеціалістів у сфері розробки програмного забезпечення почали докладати зусиль для пошуку варіантів використання технологічного прогресу для розвитку освіти.

Освіта і технології добре поєднуються з кількох причин. По-перше, технології надають доступ до безмежного обсягу інформації та ресурсів, що робить навчання більш доступним та гнучким. Студенти можуть швидко отримувати необхідні знання з використанням електронних матеріалів, відеоуроків, інтерактивних платформ та інших освітніх ресурсів, які доступні онлайн [29].

По-друге, технології дозволяють створювати інноваційні методи навчання, які залучають і мотивують студентів. Використання віртуальної реальності, розширеної реальності, ігор та інтерактивних додатків дозволяє створити заняття, які є цікавими, змістовними та забезпечують активну участь студентів [29].

Крім того, використання технологій у навчальному процесі сприяє розвитку навичок, які є необхідними у сучасному цифровому світі: інформаційна грамотність, критичне мислення, проблемне та творче мислення, комунікаційні навички [31]. Таким чином, поєднання освіти і технологій допомагає створити сучасне та ефективне навчальне середовище, яке підтримує розвиток студентів і підготовку до викликів сучасного світу. Саме тому протягом кількох останніх десятиліть, EdTech проекти, які допомагають людям навчатися, стали

популярним у всьому світі [29]. EdTech (освітні технології) - це проекти, які поєднують освітні процеси з використанням сучасних технологій. EdTech проекти спрямовані на покращення якості освіти, підвищення доступності навчання та розвиток інноваційних методів навчання. Такі проекти можуть зробити навчання більш доступним, ефективним та зручним для студентів та вчителів у будь-якому місці світу [30].

В рамках даної магістерської роботи, були зібрані, проаналізовані та описані вимоги для EdTech продукту MathLearning. MathLearning - це продукт у сфері EdTech, спрямований на школи, які прагнуть забезпечити якісну освіту для своїх учнів. Ця платформа надаватиме можливість школам створити власну віртуальну школу, де можуть бути зареєстровані всі адміністративні працівники, вчителі та учні.

Однією з ключових особливостей платформи MathLearning буде можливість користувачів створювати власні навчальні матеріали, такі як лекції, практичні завдання та іспити. Це дозволятиме індивідуалізувати навчальний процес і формувати навчальні плани різного рівня складності, відповідно до потреб учнів. Крім того, кожен вчитель зможе сформувати власну навчальну групу залежно від рівня складності навчального плану, враховуючи здібності та навички студентів групи.

Дана платформа забезпечуватиме доступ до розширеного набору навчальних ресурсів та інструментів, рекомендованих офіційними інституціями, які підтримують ефективність навчання і сприяють розвитку критичного мислення та творчих навичок учнів. Платформа буде побудована на потужному математичному модулі MathPartner, що дозволятиме учням виконувати різноманітні математичні обчислювання. Крім того, продукт буде надавати школам зручні інструменти для моніторингу прогресу учнів, що дозволить швидко реагувати на індивідуальні потреби та забезпечити якісну освіту.

Аби краще розглянути функціонал, який пропонуватиме платформа MathLearning, у таблиці 2 наведена порівняльна характеристика із двома іншими відомими EdTech продуктами: Coursera та Moodle. Coursera - це онлайн-

платформа, що надає доступ до курсів від провідних університетів та організацій по всьому світу з можливістю самостійного навчання та отримання сертифікатів. Moodle - це система управління навчання, яка дозволяє створювати та управляти навчальними курсами з можливістю спілкування, оцінювання та моніторингу прогресу студентів.

Характеристика	MathLearning	Coursera	Moodle
Основна функція	Система для створення віртуальних шкіл та навчальних планів	Надання доступу до онлайн-курсів та навчальних матеріалів	Платформа для відеоконференцій та дистанційного навчання
Функціонал для організацій/шкіл	Можливість створити власну школу, додавати адміністрацію, вчителів та учнів, формувати учнів у класи та формувати програми навчання під відповідні класи	Можливість створити власну організацію та наповнити її курсами	Можливість створити власну організацію, запросити студентів та наповнити її навчальними матеріалами
Функціонал для вчителів/викладачів	Можливість створювати власні навчальні матеріали та формувати навчальні плани	Можливість завдавати та оцінювати домашні завдання, вести журнал успішності учнів	Інструменти для проведення інтерактивних уроків, обміну матеріалами та спілкування з учнями
Функціонал для учнів/студентів	Можливість вступати до навчальних груп, проходити навчальні матеріали різного рівня складності	Доступ до відеолекцій, вправ та тестів, можливість взаємодії з вчителями та іншими учнями	Можливість долучатись до дискусій, отримувати повідомлення про домашні завдання та оцінки
Підтримка інтерактивного навчання	Змога працювати із задачами без участі сторонньої допомоги, завдяки потужному математичному ядру, яке використовує платформа	Наявність ігрових елементів, квестів та групових проектів	Інтерактивні інструменти для розвитку критичного мислення та творчості
Аналітика	Моніторинг прогресу учнів, можливість швидко аналізувати результати виконання навчальних планів	Збір та аналіз даних про прогрес учнів, відслідковування присутності та активності	Генерація звітів про успішність, статистика використання платформи

Таблиця 2. Порівняльна характеристика двох EdTech продуктів із платформою MathLearning [36; 37].

Як можна побачити із порівняння з іншими популярними продуктами у сфері EdTech, завдяки своїм особливостям, MathLearning може підтримати школи у наданні інноваційної та якісної освіти своїм учням. Дана платформа має усі шанси стати ефективним інструментом для покращення навчання та створення стимулюючого середовища для розвитку учнів. Такий продукт допомагатиме школам забезпечувати найкращі освітні послуги та готувати учнів до успішної майбутньої кар'єри у сучасному світі.

3.2 Продуктові вимоги та документація до хмарної системи “MathLearning”

У розділі 3.2 описані продуктові вимоги та наведено елементи документації до хмарної системи “MathLearning”. Для організації роботи над практичною частиною було використано наступні програмні продукти:

- **Jira** - для написання історій користувачів та їх розподілу по відповідних епіках.
- **Confluence** - для ведення проектної документації: Структури Розподілу Робіт, Матриці керування доступом на основі ролей, нотаток із зустрічей, системних повідомлень та опису інших елементів системи
- **Figma** - для створення вайрфреймів у випадках, коли історія користувача потребує візуального доповнення.
- **FigJam** - для проведення воркшопів спільно із зацікавленими особами.

3.2.1 Опис Вимог

У даному розділі наведений загальний опис функціоналу, детальний опис наведений в окремому файлі: “Продуктові_Вимоги_MathLearning_УКР”.

Модуль:	Портал Школи
Епік:	Авторизація та Автентифікація
Перелік функціоналу:	● *Сторінка лендінгу* ● *Реєстрація директора зі створенням школи* ● *Вхід з використанням електронної пошти* ● *Відновлення паролю*
Всього історій користувача:	4

Опис функціоналу: в рамках епіку Авторизація та Автентифікація, мають бути реалізовані наступні функції/сторінки:

- Сторінка Лендінгу - користувач повинен мати можливість ввести доменне ім’я платформи MathLearning та відкрити лендінг сторінку із можливістю зареєструватись або увійти на платформу MathLearning.
- Реєстрація директора зі створенням школи - користувач повинен мати можливість самостійно зареєструватись на платформі. Користувачі, які реєструються самостійно, мають також заповнити інформацію про школу. По завершенню реєстрації, користувач має отримати роль “Директор” і нова школа має бути створена в системі.
- Вхід з використанням електронної пошти - зареєстровані в системі користувачі повинні мати можливість увійти на платформу, використовуючи електронну пошту та пароль.
- Відновлення паролю - користувачі повинні мати можливість відновити пароль. Для відновлення паролю користувач має ввести електронну пошту, перейти за посиланням з листа та оновити пароль.

Модуль:	Портал Школи
Епік:	Домашня сторінка
Перелік функціоналу:	● *Перегляд домашньої сторінки* ● *Перегляд особистого профілю* ● *Редагування особистого профілю* ● *Перегляд деталей школи* ● *Редагування деталей школи*
Всього історій користувача:	5

Опис функціоналу: в рамках епіку Домашня сторінка, мають бути реалізовані наступні функції/сторінки:

- Перегляд домашньої сторінки - після успішного входу користувач повинен бути автоматично перенаправлений на домашню сторінку платформи. Користувач повинен мати можливість використовувати бічне меню для навігації по платформі, а також перейти на сторінку особистого профілю або деталей школи, використовуючи верхню панель.
- Перегляд особистого профілю - користувач повинен мати можливість переглянути особистий профіль у системі. З даної сторінки користувач повинен мати можливість редагувати особистий профіль або вийти із системи.
- Редагування особистого профілю - користувач повинен мати можливість редагувати особисті дані на платформі: Ім’я, Прізвище.
- Перегляд деталей школи - користувач повинен мати можливість переглянути деталі школи у системі. З даної сторінки Директор повинен мати можливість редагувати деталі школи.
- Редагування деталей школи - Директор повинен мати можливість редагувати деталі школи на платформі: Назва школи, Тип школи, контактна електронна адреса, контактний номер телефону.

Модуль:	Портал Школи
Епік:	Налаштування
Перелік функціоналу:	● *Список навчальних класів* ● *Створення навчального класу* ● *Перегляд навчального класу* ● *Редагування навчального класу* ● *Список дисциплін* ● *Додавання дисципліни* ● *Редагування дисципліни*
Всього історій користувача:	7

Опис функціоналу: в рамках епіку Налаштування, мають бути реалізовані наступні функції/сторінки:

- Список навчальних класів - сторінка зі списком навчальних класів у системі. Список повинен мати наступні колонки: Назва класу, Класний кабінет, Опис класу, Кількість учнів. Користувач повинен мати можливість використовувати функції сортування та фільтрування списку, перейти на деталі навчального класу, редагувати деталі навчального класу або створити новий клас.
- Створення навчального класу - користувачі повинні мати можливість створити навчальний клас, заповнивши наступну інформацію: Паралель класу, Літера класу, Класний кабінет, Опис класу. При створенні, новий запис має бути доданий у список навчальних класів.
- Перегляд навчального класу - користувачі повинні мати можливість переглянути інформацію про навчальний клас, яка була додана під час створення, а також список учнів навчального класу.
- Редагування навчального класу - користувач повинен мати можливість редагувати інформацію про навчальний клас, яка була додана під час створення.
- Список дисциплін - сторінка зі списком дисциплін у системі. Список повинен мати наступні колонки: Назва, Паралелі. Користувач повинен мати можливість використовувати функції сортування списку, редагування конкретної дисципліни або додавання нової дисципліни.
- Додавання дисципліни - користувачі повинні мати можливість створити дисципліну, заповнивши наступну інформацію: Назва дисципліни, паралелі. При створенні, новий запис має бути доданий у список дисциплін.
- Редагування дисципліни - користувачі повинні мати можливість редагувати Назву або належність до Паралелей дисципліни

Модуль:	Портал Школи
Епік:	Школа: Користувачі

Перелік функціоналу:	● *Список шкільної адміністрації* ● *Додавання шкільного адміністратора* ● *Підтвердження запрошення (для усіх ролей)* ● *Перегляд і редагування деталей адміністратора* ● *Список вчителів* ● *Додавання вчителя* ● *Перегляд і редагування деталей вчителя* ● *Список учнів* ● *Додавання учня* ● *Перегляд і редагування деталей учня*
Всього історій користувача:	10

Опис функціоналу: в рамках епіку Школа: Користувачі, мають бути реалізовані наступні функції/сторінки:

- Список шкільної адміністрації - сторінка зі списком шкільних адміністраторів у системі. Список повинен мати наступні колонки: Ім’я, Прізвище, Електронна пошта, Статус, Посада. Користувач повинен мати можливість використовувати функції сортування та фільтрування списку, перейти на сторінку деталей адміністратора, редагувати деталі адміністратора або додати нового адміністратора. Адміністратор повинен мати можливість деактивувати, активувати іншого адміністратора або повторно надіслати запрошення. Директор повинен належати до списку адміністраторів.
- Додавання шкільного адміністратора - користувачі повинні мати можливість додати нового адміністратора в школу, ввівши значення електронної пошти та посади. На вказану електронну адресу має бути надісланий лист із запрошенням і новий адміністратор має з'явитись у списку адміністрації зі статусом “Очікується”
- Підтвердження запрошення (для усіх ролей) - При отриманні запрошення у певну школу, користувач повинен мати можливість підтвердити електронну пошту та заповнити Ім’я, Прізвище та Пароль, перейшовши за посиланням із запрошенням. Після успішного підтвердження користувач повинен отримати статус "Активний" у списку та мати можливість авторизуватися.

- Перегляд і редагування деталей адміністратора - користувачі повинні мати можливість переглянути інформацію про адміністратора, а також редагувати її за наявності відповідних прав.
- Список вчителів - сторінка зі списком шкільних вчителів у системі. Список повинен мати наступні колонки: Ім’я, Прізвище, Електронна пошта, Статус, Класи. Користувач повинен мати можливість використовувати функції сортування та фільтрування списку, перейти на сторінку деталей вчителя, редагувати деталі вчителя або додати нового вчителя. Адміністратор повинен мати можливість деактивувати, активувати вчителя або повторно надіслати запрошення.
- Додавання вчителя - користувачі повинні мати можливість додати нового вчителя в школу, ввівши значення електронної пошти, дисциплін, які викладатиме вчитель та вибірку класів, які будуть доступні вчител.. На вказану електронну адресу має бути надісланий лист із запрошенням і новий вчитель має з'явитись у списку вчителів зі статусом “Очікується”.
- Перегляд і редагування деталей вчителя - користувачі повинні мати можливість переглянути інформацію про вчителя, а також редагувати її за наявності відповідних прав.
- Список учнів - сторінка зі списком учнів у системі школи. Список повинен мати наступні колонки: Ім’я, Прізвище, Електронна пошта, Статус, Клас. Користувач повинен мати можливість використовувати функції сортування та фільтрування списку, перейти на сторінку деталей учня, редагувати деталі учня або додати нового учня. Адміністратор повинен мати можливість деактивувати, активувати учня або повторно надіслати запрошення.
- Додавання учня - користувачі повинні мати можливість додати нового вчителя в школу, ввівши значення електронної пошти та класу, до якого належить учень. На вказану електронну адресу має бути надісланий лист із запрошенням і новий вчитель має з'явитись у списку учнів зі статусом “Очікується”.

- Перегляд і редагування деталей учня - користувачі повинні мати можливість переглянути інформацію про учня, а також редагувати її за наявності відповідних прав.

Модуль:	Портал Школи
Епік:	Школа: База навчальних матеріалів
Перелік функціоналу:	● *Список навчальних матеріалів (школа)* ● *Додавання навчального матеріалу (школа)* ● *Перегляд навчального матеріалу (школа)* ● *Редагування навчального матеріалу (школа)*
Всього історій користувача:	4

Опис функціоналу: функціонал, який має бути доступний користувачам із роллю Вчитель або Адміністратор. В рамках епіку Школа: База навчальних матеріалів, мають бути реалізовані наступні функції/сторінки:

- Список навчальних матеріалів (школа) - сторінка зі списком навчальних матеріалів у системі школи. Список повинен мати наступні колонки: Предмет, Складність, Тип, Підручник, Тема. Користувач повинен мати можливість використовувати функції сортування та фільтрування списку, перейти на сторінку навчального матеріалу, редагувати деталі навчального матеріалу або додати новий навчальний матеріал.
- Додавання навчального матеріалу (школа) - користувачі повинні мати можливість створити новий навчальний матеріал зі списку навчальних матеріалів, натиснувши кнопку “Додати”. Створення навчального матеріалу складається із двох блоків: інформаційний та контентний. Інформаційний блок складається із наступних полів, які обов’язкові до введення: Предмет, Паралель, Складність, Тип, Підручник, Тема, Розділ, Параграф. Контентний блок являє собою поле для вводу тексту у форматі LaTeX, куди Вчитель або Адміністратор матиме можливість ввести текст навчального матеріалу (лекцію, задачі, тощо). Користувач повинен мати можливість побачити скомпільований версію тексту LaTeX, натиснувши

кнопку "Виконати".

- Перегляд навчального матеріалу (школа) - користувачі повинні мати можливість переглядати інформаційний та контентний блоки навчального матеріалу або перейти до редагування.
- Редагування навчального матеріалу (школа) - користувачі повинні мати можливість редагувати інформаційний та контентний блоки навчального матеріалу.

Модуль:	Портал Школи
Епік:	Навчальні плани
Перелік функціоналу:	● *Список навчальних планів школи* ● *Створення навчального плану* ● *Перегляд навчальних планів* ● *Редагування навчального плану*
Всього історій користувача:	4

Опис функціоналу: функціонал, який має бути доступний користувачам із роллю Вчитель або Адміністратор. В рамках епіку Навчальні плани, мають бути реалізовані наступні функції/сторінки:

- Список навчальних планів школи - сторінка зі списком навчальних планів у системі школи. Список повинен мати наступні колонки: Предмет, Назва плану, Складність, К-сть тижнів, К-сть занять. Користувач повинен мати можливість використовувати функції сортування та фільтрування списку, перейти на сторінку навчального плану, редагувати деталі навчального плану або додати новий навчальний план.
- Створення навчального плану - користувачі повинні мати можливість створити новий навчальний план зі списку навчальних планів, натиснувши кнопку “Додати”. Під час додавання користувач повинен мати змогу користуватись двома блоками: списком навчальних матеріалів та конструктором навчального плану. Список навчальних матеріалів повинен складатись із попередньо доданих матеріалів та мати наступну ієрархічну структуру: Предмети, Паралелі, Складність, Тип. У конструкторі

навчального плану користувач повинен мати можливість створювати тижні та заняття в рамках тижня, а також заповнити інформацію про план: Предмет, Складність та назва плану. Шляхом перетягування навчальних матеріалів у конструктор плану, користувач повинен мати можливість наповнювати навчальний план. Під час створення плану, користувач повинен мати можливість переглянути контент навчального матеріалу у модальному вікні.

- Перегляд навчальних планів - користувачі повинні мати можливість переглядати інформацію про навчальний план: Предмет, Назва, Складність, К-сть тижнів, К-сть занять, а також структуру тижнів та занять і додані навчальні матеріали. Користувач повинен мати можливість перейти до редагування.
- Редагування навчального плану - користувачі повинні мати можливість редагувати інформацію про навчальний план, додавати/видаляти тижні або заняття і наповнювати план новими навчальними матеріалами. Редагування навчального плану повинне бути реалізоване у такому ж форматі, як і створення.

Модуль:	Портал Школи
Епік:	Навчальні групи
Перелік функціоналу:	● *Список навчальних груп* ● *Створення навчальної групи* ● *Перегляд навчальної групи* ● *Редагування навчальної групи*
Всього історій користувача:	4

Опис функціоналу: функціонал, який має бути доступний користувачам із роллю Вчитель або Адміністратор. В рамках епіку Навчальні групи, мають бути реалізовані наступні функції/сторінки:

- Список навчальних груп - сторінка зі списком навчальних груп у системі

школи. Кожна група має зазначати такі характеристики: назва, предмет, рівень, кількість учнів. Користувач повинен мати можливість використовувати функції сортування та фільтрування списку, перейти на сторінку групи редагувати деталі групи або створити нову групу.

- Створення навчальної групи - вчителі повинні мати можливість створити навчальну групу зі списку груп. Сторінка створення групи повинна складатись із інформаційних полів: Предмет, Клас, Рівень, а також зі списку учнів паралелі, які ще не були додані у жодну групу по визначеному предмету. Вчитель повинен мати можливість зазначити навчальний план, який буде використовуватись для даної групи.
- Перегляд навчальної групи - користувач повинен мати можливість переглянути інформацію про навчальну групу: Назва, Паралель, Рівень, Навчальний план, а також список учнів, які додані у дану навчальну групу. Зі списку учнів користувач повинен мати можливість перейти на деталі учня, редагувати інформацію про нього, або перейти на Щоденник учня.
- Редагування навчальної групи - користувач повинен мати можливість редагувати інформацію про групу або додати/вилучити учня зі списку учнів групи.

Модуль:	Портал Школи
Епік:	Зошит учня
Перелік функціоналу:	● *Виконання лекції у зошиті* ● *Виконання практичної у зошиті* ● *Виконання іспиту у зошиті*
Всього історій користувача:	1

Опис функціоналу: функціонал, який має бути доступний користувачам із роллю Учень. В рамках епіку Зошит учня, мають бути реалізовані наступні функції/сторінки:

- Перегляд зошита - зошит учня повинен містити у собі перелік предметів,

які вивчає учень. При переході на предмет, учень повинен побачити план зі списком навчальних матеріалів та можливість перейти до їх виконання. В залежності від типу навчального матеріалу (лекція/практична/іспит), хід виконання повинен відрізнятись, але кожен навчальний матеріал повинен містити наступну інформацію: дисципліна, клас, складність, тип, підручник, тема, розділ, параграф

- Виконання лекції у зошиті - У межах навчального матеріалу з типом "Лекція" учні повинні мати можливість переглянути лекцію у скомпільованому LaTeX форматі. Кількість переглядів має бути зафіксована у щоденнику.
- Виконання практичної у зошиті - У навчальному матеріалі типу "Практика" учні повинні бачити завдання та мати можливість їх виконувати в окремих віконцях, отримуючи результат на дію в кожному віконці. Під кожним завданням повинна бути можливість ввести відповідь у віконце або можливість додати/видалити віконце для введення відповіді. Користувач повинен мати можливість скомпілювати введений текст. Результат останнього віконця має вважатись рішенням завдання. Кожне завдання повинне мати функцію “Підглянути відповідь”. При натисканні учень має побачити перелік віконець із покроковим виконанням завдання. Кожне завдання повинне мати функцію “Перевірити”. При натисканні кнопку “Перевірити” користувач має побачити відповідь системи: “правильно” або “неправильно”. Завдання мають бути зібрані на одній сторінці від першого до останнього. При натисканні на кнопку “Завершити”, результати виконання завдань мають потрапити у Щоденник у вигляді шифру.
- Виконання іспиту у зошиті - У навчальному матеріалі типу "Іспит" учні повинні мати можливість виконувати ті ж функції, як і у навчальному матеріалі з типом “Практика”, але без функцій “Перевірити відповідь” та “Підглянути відповідь”. Вчитель повинен мати можливість регулювати час відкриття іспиту, а також його тривалість. При завершенні виконання

іспиту учнем, результат виконання має потрапити у щоденник у вигляді шифру.

Модуль:	Портал Школи
Епік:	Щоденник учня
Перелік функціоналу:	● *Перегляд щоденника учня* ● *Кодування виконання лекції у щоденнику* ● *Кодування виконання практичної роботи у щоденнику* ● *Кодування виконання іспиту у щоденнику*
Всього історій користувача:	2

Опис функціоналу: функціонал, який має бути доступний користувачам із роллю Адміністратор та Вчитель. Учні повинні мати можливість переглядати лише власні щоденники. В рамках епіку Зошит учня, мають бути реалізовані наступні функції/сторінки:

- Перегляд щоденника учня - щоденник повинен містити перелік предметів, які вивчає учень. При переході на предмет, користувач повинен побачити план зі списком навчальних матеріалів та результати їх виконання у вигляді коду. Кодування повинне мати формат відповідно до типу навчального матеріалу. Кодування автоматичне і базується безпосередньо на ході виконання завдання учнем, що забезпечує об’єктивність у оцінюванні і є особливістю платформи MathLearning.
- Кодування виконання лекції - в щоденнику повинно бути відображено, скільки разів лекція була відкрита учнем (напр. “3 перегляди”)
- Кодування виконання практичної роботи - у щоденнику повинно бути зображено від однієї до кількох стрічок. Одна стрічка повинна відображати факт початку виконання практичної роботи (напр. 03dB034512; 01aC001312). Один символ у рядку представляє одну задачу в практичній роботі, тому кількість символів у рядку (рядках) має дорівнювати кількості задач у практичній роботі. Кожне значення задачі може містити один із

таких символів:

- розв'язання задачі не починалось (символ - 0)
- задачу було вирішено з n спроб (n може бути від 1 до 9)
- задачу не було вирішено з n спроб (n може бути від a до i)
- задачу не було розв’язано з n спроб, а потім учень переглянув розв’язок (n може бути від A до I)

- Кодування виконання іспиту - у щоденнику зберігається один рядок, який складається із символів "0" і "1". Один символ повинен позначати одне завдання в іспиті. Якщо завдання розв’язано правильно, то така задача має бути позначена знаком "1", якщо відповідь неправильна, то позначка має бути "0". Вкінці рядка користувач повинен бачити відсоток правильних відповідей. (приклад: 0101001101, 50)

Модуль:	Адмін Панель
Епік:	Адмін Панель: Автентифікація
Перелік функціоналу:	● *Автентифікація в адмін панель* ● *Логін під конкретним користувачем*
Всього історій користувача:	2

Опис функціоналу: функціонал, який направлений на створення адмін панелі для платформи MathLearning. В рамках епіку Адмін Панель: Автентифікація, мають бути реалізовані наступні функції/сторінки:

- Автентифікація в адмін панель - відкривши сторінку браузера із доменним іменем admin.mathlearning.com, користувач повинен мати можливість ввести електронну пошту та пароль та увійти в адмін панель.
- Логін під конкретним користувачем - Адміністратор MathLearning повинен мати можливість увійти в систему від імені користувача для підтримки користувача або виконання адміністративних функцій. Система повинна відстежувати та реєструвати всі дії адміністратора та застосовувати заходи безпеки, такі як хешування паролів і безпечне керування сеансом, щоб захистити дані користувачів.

Модуль:	Адмін Панель
Епік:	Адмін Панель: Сутності
Перелік функціоналу:	● *Список шкіл* ● *Перегляд деталей школи* ● *Список користувачів* ● *Перегляд деталей користувача*
Всього історій користувача:	4

Опис функціоналу: в рамках епіку Адмін Панель: Сутності, мають бути реалізовані наступні функції/сторінки:

- Список шкіл - сторінка зі списком усіх шкіл у системі MathLearning. Список повинен мати наступні колонки: ID, Назва школи, К-сть користувачів. Користувач повинен мати можливість використовувати функції сортування та пошуку по списку, а також перейти на сторінку деталей школи.
- Перегляд деталей школи - користувач повинен мати можливість побачити наступну інформацію про школу: ID, Назва, Тип, Контактна електронна адреса та контактний телефон, а також Ім’я, Прізвище та Електронна пошта директора.
- Список користувачів - сторінка зі списком усіх користувачів у системі MathLearning. Список повинен мати наступні колонки: ID, Ім’я, Прізвище та Електронна пошта, Назва школи, Роль. Адміністратор MathLearning повинен мати можливість використовувати функції сортування та пошуку по списку, а також перейти на сторінку деталей користувача.
- Перегляд деталей користувача - користувач повинен мати можливість побачити наступну інформацію про користувача школи: : ID, Ім’я, Прізвище та Електронна пошта, Назва школи, Роль. Адміністратор MathLearning повинен мати можливість увійти в школу під конкретним користувачем.

Модуль:	Адмін Панель
Епік:	Адмін Панель: База навчальних матеріалів
Перелік функціоналу:	• *Список навчальних матеріалів (адмін панель)* • *Додавання навчального матеріалу (адмін панель)* • *Перегляд навчального матеріалу (адмін панель)* • *Редагування навчального матеріалу (адмін панель)*
Всього історій користувача:	4

Опис функціоналу: в рамках епіку Адмін Панель: База навчальних матеріалів, мають бути реалізовані такі самі функції та сторінки, як і для епіку Школа: База навчальних матеріалів, але Навчальні матеріали, які створюються в Адмін Панелі, повинні автоматично додаватися до списку навчальних матеріалів кожної школи. У переліку навчальних матеріалів школи має бути додаткова колонка з назвою "Походження". Якщо навчальний матеріал створено з Адмін Панелі, джерелом має бути "МОН", інакше джерелом має бути "Школа".

3.2.2 Структура Розподілу Робіт

В рамках розробки документації, на основі описаних вимог була складена Структура Розподілу Робіт (Work Breakdown Structure, WBS). Структура Розподілу Робіт - це інструмент проектного управління, що допомагає розбити проект на більш менш дрібні, керовані елементи. WBS є важливим елементом документації для розуміння обсягу робіт, визначення послідовності виконання робіт, та встановлення пріоритетів. Він допомагає забезпечити систематичний підхід до управління проектом та забезпечити виконання всіх потрібних робіт для досягнення мети проекту. WBS для проекту MathLearning зображена у таблиці 3

Платформа/Функціонал	***Послідовність***	***Пріоритет (1-5)***	***Задача в Jira***
Портал Школи			

Авторизація та Автентифікація			
Лендінг	1	5	MAT-4
Реєстрація зі створенням школи	2	5	MAT-2
Вхід використовуючи електронну пошту	3	5	MAT-3
Відновлення паролю	4	2	MAT-12
Домашня сторінка			
Перегляд домашньої сторінки	5	5	MAT-64
Перегляд особистого профілю	6	3	MAT-14
Редагування особистого профілю	7	3	MAT-15
Перегляд деталей школи	8	3	MAT-7
Редагування деталей школи	9	3	MAT-8
Налаштування			
Список навчальних класів	10	5	MAT-20
Створення навчального класу	11	5	MAT-19
Перегляд навчального класу	12	5	MAT-69
Редагування навчального класу	13	5	MAT-21
Список дисциплін	14	5	MAT-16
Додавання дисципліни	15	5	MAT-17
Редагування дисципліни	16	5	MAT-18
Школа: Користувачі			
Список Шкільної адміністрації	17	5	MAT-9
Додавання Шкільного адміністратора	18	5	MAT-10
Підтвердження Запрошення (для усіх ролей)	19	5	MAT-66
Перегляд і редагування деталей адміністратора	20	3	MAT-25
Список Вчителів	21	5	MAT-11
Додавання Вчителя	22	5	MAT-22
Перегляд і редагування деталей вчителя	23	3	MAT-67
Список Учнів	24	5	MAT-23
Додавання Учня	25	5	MAT-24
Перегляд і редагування деталей учня	26	3	MAT-26
Школа: База навчальних матеріалів			

Список навчальних матеріалів (школа)	28	5	MAT-41
Додавання навчального матеріалу (школа)	29	5	MAT-43
Перегляд навчального матеріалу (школа)	30	5	MAT-42
Редагування навчального матеріалу (школа)	31	5	MAT-44
Навчальні плани			
Список навчальних планів школи	32	5	MAT-38
Створення навчального плану	33	5	MAT-37
Перегляд навчальних планів	34	5	MAT-70
Редагування навчального плану	35	5	MAT-39
Навчальні групи			
Список навчальних груп	36	5	MAT-33
Створення навчальної групи	37	5	MAT-32
Перегляд навчальної групи	38	5	MAT-71
Редагування навчальної групи	39	5	MAT-34
Зошит учня			
Виконання лекції/практичної/іспиту у зошиті	40	5	MAT-48
Щоденник учня			
Перегляд щоденника учня	41	5	MAT-28
Кодування виконання навчального матеріалу у щоденнику (лекція/практична/іспит)	42	5	MAT-30
Адмін Панель			
Адмін Панель: Автентифікація			
Автентифікація в адмін панель	43	4	MAT-50
Логін під конкретним користувачем	44	4	MAT-62
Адмін Панель: Сутності			
Список шкіл	45	4	MAT-60
Перегляд деталей школи	46	2	MAT-61
Список користувачів	47	3	MAT-58
Перегляд деталей користувача	48	2	MAT-59
Адмін Панель: База навчальних матеріалів			
Список навчальних матеріалів (адмін панель)	49	5	MAT-55

Додавання навчального матеріалу (адмін панель)	50	5	MAT-54
Перегляд навчального матеріалу (адмін панель)	51	5	MAT-56
Редагування навчального матеріалу (адмін панель)	52	5	MAT-57

Таблиця 3. Структура Розподілу Робіт для продукту MathLearning

3.2.3 Матриця керування доступом на основі ролей

Матриця контролю доступу на основі ролей (Role-Based Access Control, RBAC) - це модель управління доступом, в якій доступ до функціоналу системи для користувача конфігурується на основі ролі, яка була йому призначена. У RBAC матриці функціонал відображаються по рядках, а ролі по колонках. Кожна комірка матриці показує, які ролі мають доступ до відповідного функціоналу. Це дозволяє легко визначати та керувати правами доступу для кожної ролі і забезпечує гнучкий та ефективний контроль доступу до функціоналу. У таблиці 5 наведена RBAC матриця для продукту MathLearning:

Функціонал	**Учень**	**Вчитель**	**Адміністратор**	**Директор**
Створення школи	-	-	-	+
Перегляд деталей школи	+	+	+	+
Редагування деталей школи	-	-	+	+
Перегляд особистого профілю	+	+	+	+
Редагування особистого профілю	+	+	+	+
Перегляд списку Шкільної адміністрації	+	+	+	+
Додавання Шкільного адміністратора	-	-	+	+
Перегляд і редагування деталей адміністратора	-	-	+	+
Перегляд списку Вчителів	+	+	+	+
Додавання Вчителя	-	-	+	+
Перегляд і редагування деталей вчителя	-	-	+	+
Перегляд списку Учнів	+	+	+	+
Додавання Учня	-	-	+	+

Перегляд і редагування деталей учня	-	-	+	+
Перегляд списку навчальних класів	-	-	+	+
Створення навчального класу	-	-	+	+
Редагування навчального класу	-	-	+	+
Перегляд списку дисциплін	-	-	+	+
Додавання дисципліни	-	-	+	+
Редагування дисципліни	-	-	+	+
Перегляд списку навчальних матеріалів (школа)	-	+	+	+
Додавання навчального матеріалу (школа)	-	+	+	+
Перегляд навчального матеріалу (школа)	-	+	+	+
Редагування навчального матеріалу (школа)	-	+	+	+
Перегляд списку навчальних планів школи	-	+	+	+
Створення навчального плану	-	+	+	+
Перегляд навчальних планів	-	+	+	+
Редагування навчального плану	-	+	+	+
Перегляд списку навчальних груп	-	+	+	+
Створення навчальної групи	-	+	+	+
Перегляд навчальної групи	-	+	+	+
Редагування навчальної групи	-	+	+	+
Виконання лекції/практичної/іспиту у зошиті	+	-	-	-
Перегляд щоденника учня	власний	+	+	+

Таблиця 4. Матриця керування доступом на основі ролей для продукту MathLearning.

Висновки до розділу 3

У третьому розділі було наведено визначення та місце EdTech проектів у сфері розробки програмного забезпечення, розкрито суть хмарної системи MathLearning та проведено порівняння із іншими освітніми платформами. Також у даному розділі було описано вимоги до основних модулів системи

MathLearning: портал школи та адмін панель, а також зазначено інші елементи документації для платформи MathLearning.

ВИСНОВКИ

У рамках даної роботи було розкрито поняття та суть вимог до програмного забезпечення. Було розглянуто класифікацію продуктовим вимог, а також компетенції та навички, якими має володіти спеціаліст зі збору вимог.

Практична частина даної роботи полягала у зборі, аналізі, описі та валідації продуктових вимог до хмарної системи MathLearning, що складається із двох модулів: порталу школи та панелі адміністратора MathLearning. Для збору вимог було проведено 24 зустрічі із зацікавленими особами у форматі воркшоп та інтерв'ю. Результатом стала підготовка документу з вимогами до даної системи, викладена двома мовами: англійською та українською, що розширює потенційну аудиторію команди розробки. Вимоги до продукту були описані, використовуючи формат User Story (історій користувача) з відповідним описами Acceptance Criteria (критеріїв приймання) до кожної історії користувача. Історії користувачів з підвищеною складністю також доповнювались вайрфреймами для кращого розуміння опису функціоналу. Разом із вимогами також була складена Work Breakdown Structure (Структура Розподілу Робіт) та Role Based Access Control Matrix (Матриця керування доступом на основі ролей).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Академічний тлумачний словник української мови. Онлайн версія академічного тлумачного «Словника української мови» в 11 томах (1970–1980). Том 1. 1970. 434 с.
2. Wiegers K. Software Requirements / K. Wiegers, J. Beatty. – Redmond: Microsoft Press, 2013. – 637 с. – (3 edition).
3. A Guide to the Business Analysis Body of Knowledge (BABOK Guide) – Toronto: International Institute of Business Analysis, 2015. – 500 с. – (version 3).
4. Young R. R. Effective Requirements Practices / Ralph R. Young. – Boston: Addison-Wesley Professional, 2001. – 394 с. – (1st edition).
5. Waqar H. What are business requirements? / Hassaan Waqar // educative.io. – 2023. – Режим доступу до ресурсу: https://www.educative.io/answers/what-are-business-requirements.
6. Loconsole A. Theoretical Validation and Case Study of Requirements Management Measures / A. Loconsole, J. Börstler // UMEÅ UNIVERSITY. – 2013. – Режим доступу до ресурсу: https://www.researchgate.net/publication/238355117_Theoretical_Validation_and_Case_Study_of_Requirements_Management_Measures.
7. Robertson S. Mastering the Requirements Process: Getting Requirements Right / S. Robertson, J. Robertson. – Boston: Addison-Wesley Professional, 2012. – 576 с. – (3rd edition).
8. Cohn M. Agile Estimating and Planning / Mike Cohn. – London: Pearson, 2005. – 368 с. – (1st edition).
9. Cohn M. User Stories Applied: For Agile Software Development / Mike Cohn. – Redwood City: Addison-Wesley Professional, 2004. – 304 с. – (1st edition).
10. Shore J. The Art of Agile Development / J. Shore, S. Warden. – Sebastopol, California: O'Reilly Media, 2007. – 438 с. – (1st edition).

11. Cockburn A. Writing Effective Use Cases / Alistair Cockburn. – Boston: Addison-Wesley Professional, 2000. – 304 с. – (1st edition).
12. Wiegers K. Software Requirements 2 / Karl Wiegers. – Redmond: Microsoft Press, 2003. – 544 с. – (2nd edition).
13. Pohl K. Requirements Engineering Fundamentals, Principles, and Techniques / Klaus Pohl. – Heidelberg: Springer Berlin, 2016. – 813 с. – (1st edition).
14. Podeswa H. Why Your Agile Organization Needs a High-Performing Requirements Engineering Competency / Howard Podeswa // Requirements Engineering Magazine. – 2023. – Режим доступу до ресурсу: https://re-magazine.ireb.org/articles/why-your-agile-organization-needs-a-high-performing-requirements-engineering-competency.
15. Kassab M. The changing landscape of requirements engineering practices over the past decade / Mohamad Kassab. – Ottawa: IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE), 2015. pp. 1-8, doi: 10.1109/EmpiRE.2015.7431299.
16. Research Methods: Scenarios and User Stories // Nightingale Design Research – Режим доступу до ресурсу: https://nightingaledesignresearch.com/insights/research-methods-scenarios-and-user-stories/.
17. Kaley A. Mapping User Stories in Agile / Anna Kaley // Nielsen Norman Group. – 2021. – Режим доступу до ресурсу: https://www.nngroup.com/articles/user-story-mapping/.
18. What Is Elicitation: Top Requirement Elicitation Techniques for 2023 // Simplilearn. – 2023. – Режим доступу до ресурсу: https://www.simplilearn.com/what-is-elicitation-article.
19. Top 10 Most Common Requirements Elicitation Techniques // Software Testing Help. – 2023. – Режим доступу до ресурсу: https://www.softwaretestinghelp.com/requirements-elicitation-techniques/.
20. Babinski B. Software Product Requirements: The Complete Guide How to Write Decent High-Level Project Specifications / Bartosz Babinski // STXnext. – 2023.

– Режим доступу до ресурсу: https://www.stxnext.com/blog/software-product-requirements-high-level-project-specifications-guide.

21. McVey M. Article: The Business Analyst Career Road Map / Maureen McVey // IIBA. – 2023. – Режим доступу до ресурсу: https://www.iiba.org/professional-development/knowledge-centre/articles/roles-ba-career-road-map/?gad=1&gclid=Cj0KCQjw7PCjBhDwARIsANo7CgnpHWby0GWB0bCzXc2Q2wNka_Omlsm0gxqYxzNRINSLaVEsTYNkwiQaAu86EALw_wcB.
22. Pratt M. What is a business analyst? A key role for business-IT efficiency / M. Pratt, S. White // CIO. – 2019. – Режим доступу до ресурсу: https://www.cio.com/article/276798/project-management-what-do-business-analysts-actually-do-for-software-implementation-projects.html.
23. Jiao J. Customer Requirement Management in Product Development: A Review of Research Issues. – Concurrent Engineering, 2006. – pp.173-183 – (vol. 14; issue 3).
24. Aziz R. The Impacts of Requirements Relationships Knowledge on Requirements Quality and Software Development Project Success / R. Aziz, B. Wong // CEUR Workshop Proceedings. – 2021. – Режим доступу до ресурсу: https://ceur-ws.org/Vol-3062/Paper07_QuASoQ.pdf.
25. Saavedra R. Software Requirements Quality Evaluation: State of the art and research challenges / R. Saavedra, L. Ballejos, M. Ale // Argentine Symposium on Software Engineering. – 2013. – Режим доступу до ресурсу: https://42jaiio.sadio.org.ar/proceedings/simposios/Trabajos/ASSE/18.pdf.
26. Software Engineering | Requirements Validation Techniques // Geeksforgeeks. – 2023. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/software-engineering-requirements-validation-techniques/.
27. Sharma R. Top 7 Best Business Analytics Tools Recommended for every Business Analyst / Rohit Sharma // upGrad. – 2022. – Режим доступу до ресурсу: https://www.upgrad.com/blog/best-business-analytics-tools-recommended-for-every-business-analyst/.

28. What Tools Do Business Analysts Use? // BrainStation. – 2023. – Режим доступу до ресурсу: https://brainstation.io/career-guides/what-tools-do-business-analysts-use.
29. Dahya N. Education in Conflict and Crisis: How Can Technology Make a Difference? / Negin Dahya // Deutsche Gesellschaft für Internationale Zusammenarbeit. – 2016. – Режим доступу до ресурсу: https://www.edu-links.org/sites/default/files/media/file/GIZ%20InDesign-Vorlage%20fu%CC%88r%20Publikationen%20%E2%80%93%20DIN%20A4%20hoch.pdf.
30. Haßler B. Perspectives on Technology, Resources and Learning: Productive Classroom Practices, Effective Teacher Professional Development / B. Haßler, L. Major, S. Watson // Creative commons. – 2014. – Режим доступу до ресурсу: https://www.educ.cam.ac.uk/people/staff/watson/Hassler%20et%20al%202016%20-%20Perspectives%20on%20Technology,%20Resources%20and%20Learning%20(Full).pdf.
31. Selwyn N. Technology and education - why it's crucial to be critical / Neil Selwyn // Academia. – 2014. – Режим доступу до ресурсу: https://www.academia.edu/7771394/Technology_and_education_-_why_its_crucial_to_be_critical.
32. Overview of Rational DOORS // IBM Corporation. – 2023. – Режим доступу до ресурсу: https://www.ibm.com/docs/en/elms/ermd/9.6.0?topic=overview-rational-doors.
33. Tutorial: using Confluence and Jira Service Management together // Atlassian Режим доступу до ресурсу: https://www.atlassian.com/software/confluence/resources/guides/extend-functionality/confluence-jsm#overview.
34. Finnegan M. What is Trello? A guide to Atlassian’s collaboration and work management tool / Matthew Finnegan // Computerworld. – 2021. – Режим доступу до ресурсу: https://www.computerworld.com/article/3226447/what-is-trello-a-guide-to-atlassians-collaboration-and-work-management-tool.html.

35. Requirements Management Tool for HW/SW Systems // ReqView – Режим доступу до ресурсу: https://www.reqview.com/.
36. Leighton M. Coursera is one of the top online learning platforms / M. Leighton, J. Pugachevsky // Insider Reviews. – 2022. – Режим доступу до ресурсу: https://www.businessinsider.com/guides/learning/what-is-coursera.
37. Alvelos H. The Use of Moodle e-learning Platform: A Study in a Portuguese University / H. Alvelos, T. Leonor // ScienceDirect. – 2012. – Режим доступу до ресурсу: https://doi.org/10.1016/j.protcy.2012.09.037.