МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки
Факультет Комп'ютерних наук
Кафедра Програмної Інженерії

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження методів побудови програмних систем для управління вимогами в IT-проектах

Виконала:
студент 2 курсу, групи ІПЗм-21-1
Каменєв Р.В.
(прізвище, ініціали)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-наукова
Керівник доц. Голян Н.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. Кафедри _____ З.В. Дудар

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної Інженерії

Рівень вищої другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
« ____» _____20___р

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Каменєва Романа Віталійовича
(прізвище, ім'я, по батькові)

1.Тема роботи «Дослідження методів побудови програмних систем для управління вимогами в IT-проектах »

затверджена наказом по університету від 29 березня 2023 р. №302Ст

1. Термін подання студентом роботи до екзаменаційної комісії 22 травня 2023 р.

2. Вихідні дані до роботи Статичні та гнучкі методології, методи фільтрації вимог, характеристика програмного продукту

3. Перелік питань, що потрібно опрацювати в роботі Вступ, аналіз предметної галузі, постановка задачі, аналіз методів дослідження, порівняння результатів планування, планування розробки програмного забезпечення, розробка програмного забезпечення

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапівроботи	Примітка
1	Аналіз предметної області	25.02.2023	виконано
2	Постановка завдання	05.03.2023	виконано
3	Аналіз існуючих методів	13.04.2023	виконано
4	Планування дослідження	19.04.2023	виконано
5	Проведення дослідження	10.04.2023	виконано
6	Підготовка пояснювальної записки	30.04.2023	виконано
7	Підготовка презентації та доповіді	02.05.2023	виконано
8	Перевірка на плагіат	15.05.2023	виконано
9	Нормоконтроль	16.05.2023	виконано
10	Рецензування	16.05.2023	виконано
11	Занесення роботи в електронний архів	22.05.2023	виконано
12	Попередній захист	22.05.2023	виконано
13	Допуск до захисту у зав. кафедри	22.05.2023	виконано

Дата видачі завдання ____23 січня 2023 р._____

Студент _____

(підпис)

Керівник роботи _____ дой. Голян Н.В.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи, 57 сторінок, 12 рисунки, 31 джерело, 4 додатки.

АНАЛІЗ ДАНИХ, БІЗНЕС, ПРОГРАМНА СИСТЕМА, ПРОЕКТ, КОМПАНІЯ, AGILE, WATERFLOW, KANO, SMART.

Об’єктом дослідження є методи побудови програмних систем для управління вимогами в IT-проектах.

Метою роботи є розробка плану та проведення підготовчих робіт щодо дослідження ефективності методів побудови проектів у сфері інформаційних технологій та способів управління вимогами в проектах та дослідження методів створення програмних систем.

У результаті роботи була створена система з управління вимогами в IT-проектах та проаналізовані статичні та гнучкі методи розробки та планування програмного забезпечення.

DATA ANALYSIS, BUSINESS. SOFTWARE SYSTEM, PROJECT, COMPANY, AGILE, WATERFLOW, KANO, SMART.

The object of the research is methods of building software systems for requirements management in IT projects.

The purpose of the work is to develop a plan and carry out preparatory work on researching the effectiveness of methods of building projects in the field of information technologies and ways of managing requirements in projects and researching methods of creating software systems.

As a result of the work, a system with management requirements in IT projects was created, and static and flexible software development and planning methods were analyzed.

Я, Каменєв Роман Віталійович, студент гр. ІПЗм-21-1, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів побудови програмних систем для управління вимогами в ІТ-проектах», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих таелектронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Через великий вплив інформатизації на сучасних людей у 2022 році вже неможливо уявити людину без досвіду з будь-якою інформаційною системою. Для подальшого розповсюдження цих систем країни створюють фонди та заохочують молодь до участі в різного типу проектах. Через велику кількість бажаючих учбові заклади створюють освітні програми, які націлені на людей, які прагнуть отримати нову професію та бути на одній хвилі з розвитком технологій, а через те, що кожна людина сьогодні має зв'язок з будь-яким типом технологій – ще й мати змогу будувати сьогодення та спрощувати життя мільйонів людей.

Програмні системи створюються здебільшого за допомогою великої кількості людей, які мають технічні навички та мають змогу виконувати різного роду завдання в окремий проміжок часу. Зібрати людей з необхідними навичками досить складно, через брак кваліфікованих кадрів на ринку праці, які б з легкістю розуміли поставлену перед ними задачу та виконували її в найменш можливий проміжок часу. Здебільшого люди працюють за різного роду винагороду та намагаються уникнути зайвої втрати часу на обробку не потрібної інформації, або ж інформації, яку складно усвідомити через її неповноту або значно заплутану структуру.

На даний момент вже існує велика кількість товариств та компаній в яких є кваліфіковані та відповідальні співробітники. Саме тому люди, які мають різного роду ідеї для створення програмного забезпечення звертаються до таких компаній, як: Softserve [1], GlobalLogic [2], Epam [3]. Ці компанії протягом десятків років займаються розробкою та створенням IT-проектів та мають репутацію стабільних та відповідальних. Через позитивну репутацію перелічені компанії мають великий попит на свої послуги, а люди з необхідною кваліфікацією бажають доєднатися до команд з розробки програмного забезпечення.

Для досягнення мети в створенні замовленого продукту долучають інженерів програмного забезпечення, дизайнерів та менеджерів. Серед цих людей повинно бути єдине бачення кінцевої форми продукту а також шляху з розробки

та введення в експлуатацію.

Варто зазначити, що люди можуть мати різний погляд на реалізацію систем та вважати свою думку єдиною вірною, через це є загроза нечіткого або ж зовсім не достовірного бачення програмного продукту, що може призвести до небажаного результату у вигляді не працюючого або ж не відповідного до вимог замовника продукту.

Для вирішення проблеми з інтерпретацією даних від замовника наймають окремого спеціаліста, який ітеративно буде дізнаватися поставлені перед замовником цілі та що саме треба досягнути завдяки розробленому програмному забезпеченню.

Цей процес займає багато часу та потребує додаткових коштів, що збільшує вартість проекту для замовника та не дає повної гарантії в досягненні поставлених цілей. Саме через це були створені методи для визначення вимог та структурування потрібних даних для подальшого створення продукту.

Метою роботи є розробка плану та проведення підготовчих робіт щодо дослідження ефективності методів побудови проектів у сфері інформаційних технологій та способів управління вимогами в проектах та дослідження методів створення програмних систем.

З огляду на окреслену проблематику предметною галуззю даного дослідження є аналіз життєвого циклу проектів у сфері інформаційних технологій, а саме дослідження методів побудови та управління вимогами ІТ-продуктів. Через це ми маємо змогу дослідити також розробку одного з прототипів окресленого програмного забезпечення.

1 ОПИС ПРОБЛЕМНОЇ ГАЛУЗІ

1.1 Аналіз предметної області

Сегментом дослідження є проекти у сфері інформаційних технологій, які не мають жорсткої класифікації та можуть бути направлені на будь-який сегмент суспільства та переслідувати окремі цілі, що можуть не відповідати наповненню, або ідеології програмного продукту.

Поняття вимоги можна класифікувати як збірка правил, та цілей які повинен реалізовувати проект у рамках життєвого циклу. Ця збірка не є фіксованою та зберігає свій стан лише на проміжок часу за вичерпанням якого вимоги можуть бути доповнені новою інформацією або змінені таким чином, щоб відповідати новим реаліям ринку, для якого проводиться розробка програмного продукту. Проміжок часу в якому зміни вимог не відбувається не є фіксованим та може бути подовжений, або, навпаки, скорочений згідно зі згодою замовника.

З огляду на визначення поняття вимог до проектів у сфері інформаційних технологій, треба визначити систему завдяки якій внесення змін до списку правил та цілей може нести найменшу втрату часу на створення нового або вдосконалення вже існуючого функціоналу системи. Ці зміни несуть в собі додаткові фінансові зобов’язання зі сторони замовника, окрім випадків, де кількість та якість змін до вимог не була узгоджена раніше.

Для скорочення часу обробки нової інформації від замовника, а також прискоренню адаптації виконавців з компаній підрядника до нових завдань є потреба в адаптивній системі, яка матиме змогу формувати звіти щодо кількості внесених змін та відслідковувати кількість часу витраченого на реалізацію.

З огляду на існуючі варіанти можна виділити системи з прив’язкою до життєвого циклу програмного забезпечення.

Для дослідження було обрано найпопулярніші моделі життєвих циклів, Agile [4,5,6] та Waterflow [7,8]. Підходи до реалізації програмного продукту мають суттєві відмінності.

Agile методологія є гнучкою, через велику варіативність можливих підходів до опрацювання поставлених задач та базується на принципах маніфесту [9].

Також дана методологія є ітеративною (див. рис.1.1).

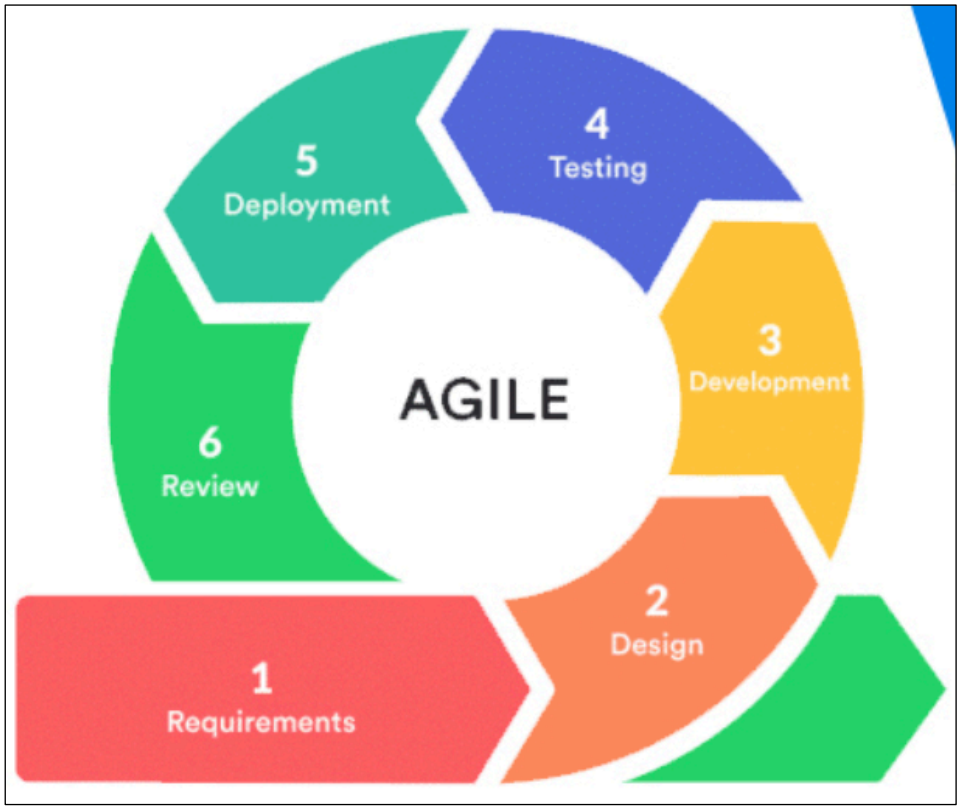

Рисунок 1.1 – Життєвий цикл проекту за методологією Agile [10]

Для реалізації встановленого плану першою ціллю є створення MVP [11]. Подальший розвиток продукту є ітеративним де додання нового функціоналу вписується у жорсткі межі, які у свою чергу поділені на сегменти під назвою спринти. Кожен спринт зазвичай ж довжиною від одного тижня до чотирьох. За цей проміжок часу команда інженерів програмного забезпечення повинна досягнути поставлених цілей, які були узгоджені до початку спринта.

Waterfall у свою чергу є більш стандартизованим та у кінці життєвого циклу розробки команда отримує вже готовий продукт з повним функціоналом. На відміну до Agile підходів, підхід Waterfall займає більше часу на всіх можливих етапах, але головною позитивною характеристикою цього підходу є чітко сформульовані вимоги, які не мають наміру на зміну у короткий проміжок часу. Саме через прогнозованість цей підхід зазвичай обирають великі корпорації для своїх програмних систем.

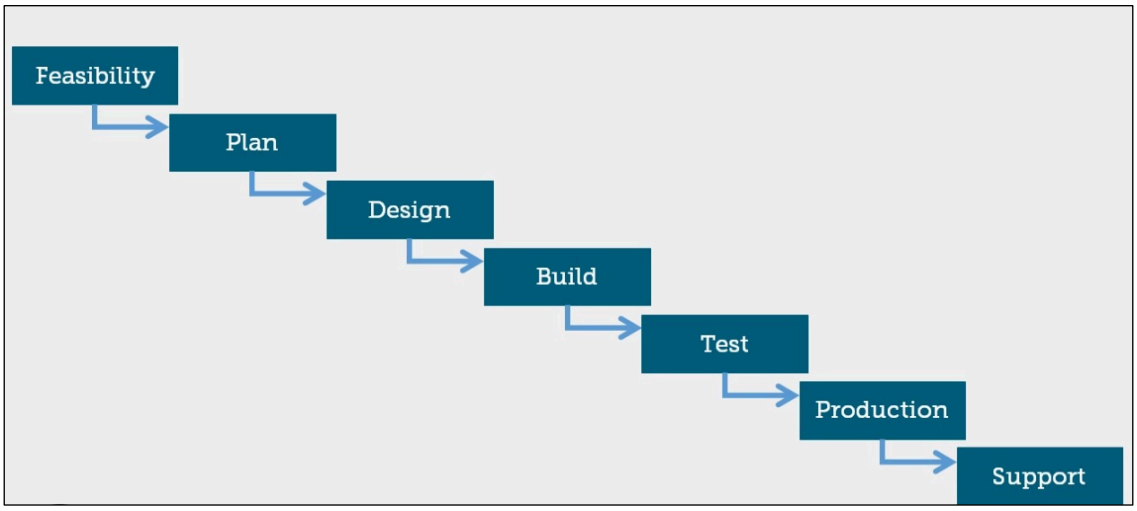

Рисунок 1.2 – Життєвий цикл проекту за методологією Waterfall [12]

Однак підхід Waterfall має суттєвий недолік, який обмежує його сферу використання, а саме вимоги які не мають наміру на зміну. Цей недолік на перший погляд має лише позитивні ознаки, але є винятки. Через довгий шлях від планування програмного проекту до його реалізації, категорія людей, яким потенційно цікава ця програмна система вже може почати користування аналогічними системами, хоча й вони не матимуть всього функціоналу проекту створеного за методологією Waterfall. З цього виникає потреба модернізувати систему таким чином, щоб додаткові або уточнені вимоги не створювали суттєвих проблем при інтерпретації її та трансформуванні до задач, які будуть виконувати інженери програмного забезпечення [13].

Однак проблема управління вимогами до проектів існує вже досить давно, і вже створено кілька варіантів комерційних систем, які мають змогу полегшити та структурувати управління вимогами.

Розглянемо платформу Visure [14] (див. рис. 1.3).

Дана платформа є одним з представників програмних продуктів, де намагалися вирішити питання вимог до проектів, їх незрозумілості та недостатньої обізнаності виконавців, через брак деталей опису вимоги.

Модель реалізації програмного продукту через веб застосунок є вдалою, через можливість використання на будь-якій платформі, яка підтримує веб-застосунки.

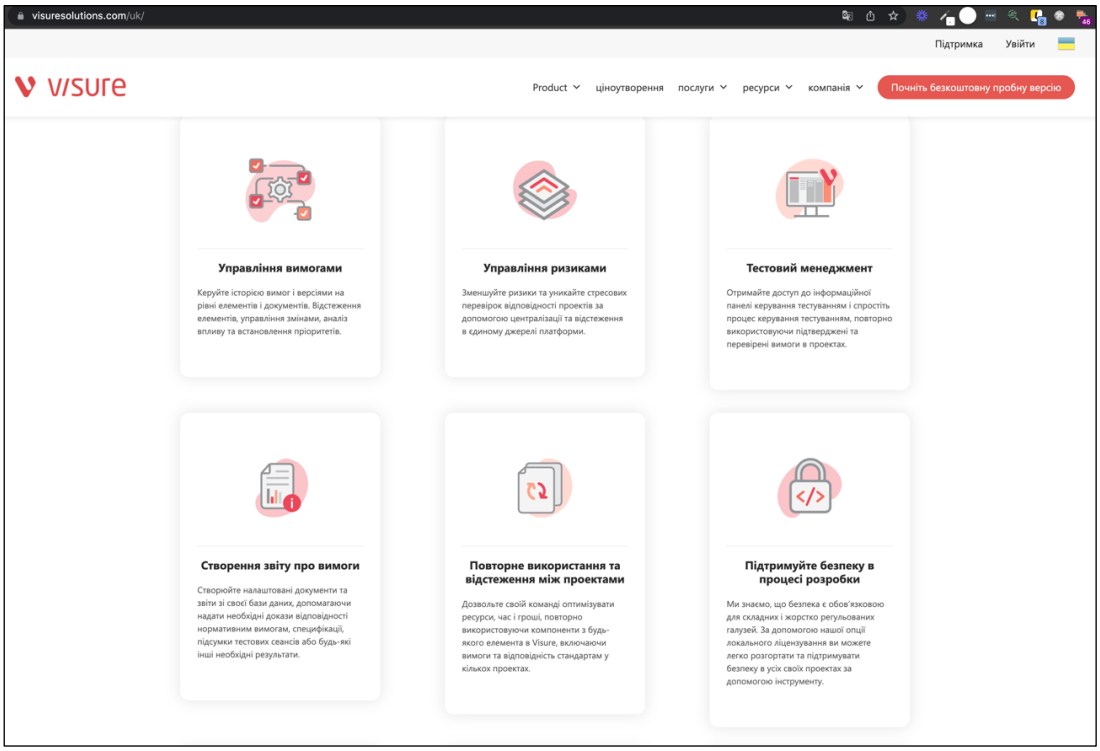

Рисунок 1.3 – Домашня сторінка «Visure» [14]

Данна система має досить різноманітний функціонал, але нас цікавить саме управління вимогами. Система має вигляд таблиці (див. рис. 1.2) та виділяє кольором зміни та позначає їх як критичні у разі необхідності.

Цей підхід до відстеження змін є одним з оптимальних та має візуальне відображення змін, також сервіс надає можливість побудувати різного роду графікі для модулювання та прогнозування результатів в залежності від змін.

Однак, дана система не є ідеальною та не пророблює концепт виявлення вимог до продукту, а лише працює з вже існуючими вимогами та дозволяє відстежувати зміни в них. Через це маємо проблему виявлення вимог до програмного продукту, а саме аналізу можливих варіантів створення вимог різного роду. Також систематизацію та фільтрацію вимог на предмет можливості виконання та з огляду на виділений час для розробки проекту.

Окремі вимоги потребують систематизації та групування, через їх схожість але відрізний функціонал. Саме це є причиною для детального документування вимоги та відокремлення основних критеріїв, позначання приоритезації і визнання складності поставленої задачі, що повинна корелювати з виділеним

часом.

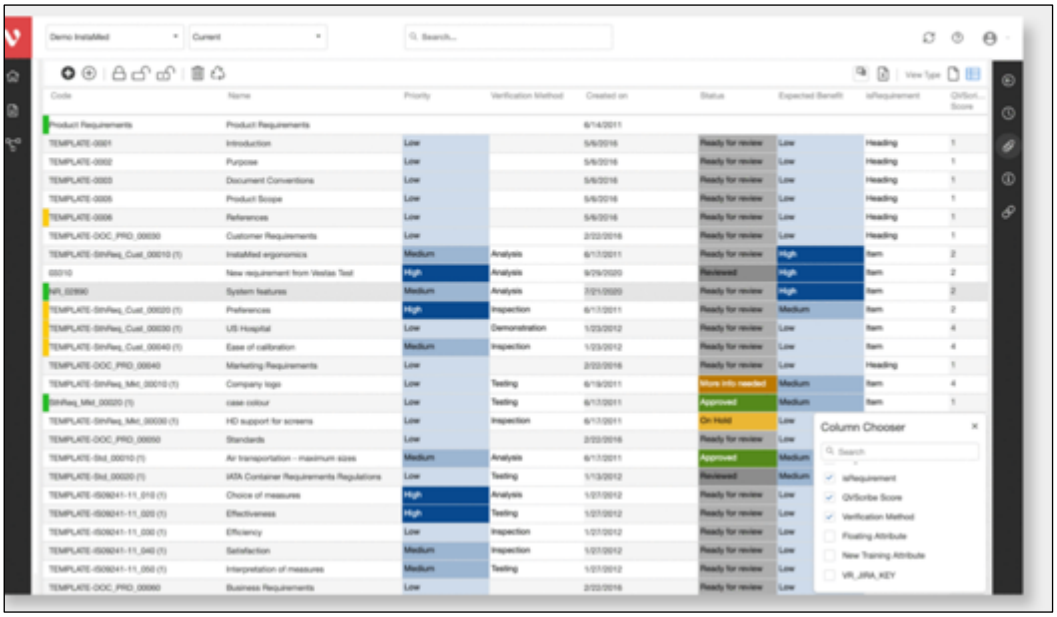

Рисунок 1.4 – Сторінка з відстежування змін у вимогах до продукту [14]

Вимоги в рамках IT-проектів можуть бути класифіковані як:

– функціональні потреби;

– не функціональні потреби.

Функціональні потреби у свою чергу можна розділити на потреби бізнесу та користувачів. Однак кожна з класифікацій вимог важлива для моделювання програмної системи, а головне для окреслення проблеми, яку вирішуватиме цей проект.

Фільтрація вимог на предмет актуальності та можливості реалізації є одним з головних етапів обробки, згенерованих різними методами, потреб для бізнесу та користувачів.

1.2 Постановка задачі

Інформація, наведена в попередньому пункті дає змогу сформувати задачу для опрацювання у цій роботі – дослідження та методів побудови програмних систем для управління вимогами в IT-проектах.

Саме проблема, яка постає перед користувачами є головним критерієм для створення потрібного продукту, який матиме популярність серед користувачів.

Під словом проблема можна виділити не ефективну систему чи спосіб взаємодії користувача з послугами або повну відсутність конкурентної реалізації програмного продукту.

Дана предметна область є однією з можливих сфер використання системи управління вимог до проектів, через це дане дослідження повинно бути актуально саме для проектів у сфері інформаційних технологій.

Для виконання поставленої задачі необхідно проаналізувати:

– наявні методи для створення ідей;
– системи критеріїв щодо реальності проблеми;
– наявні методи фільтрації нераціональних вимог.

Через опис варіантів для генерації обрати найліпшу за умови недовгострокового проекту, який можливо буде реалізувати. За обраною стратегією слід провести аналіз обраної проблеми на потрібність в вирішенні.

З зазначених критеріїв аналіз методів фільтрації вимог слід провести на прикладі створення вимог до системи, яка буде відслідковувати зміну та актуальність вимог у вигляді веб застосунку.

З цього виникає завдання на аналіз:

– описати можливу проблему, яка потребує рішення;
– перевірити проблему на життєздатність;
– виявити потенційних користувачів;
– сформувати вимоги;
– скоротити їх кількість за допомогою фільтрування;
– визначити критерії які матимуть вагу для фільтрації.

Дані вимоги є класичними для проектів у сфері інформаційних технологій та узагальненими, через що результати дослідження мають бути з оптимальними варіантами, які можна використовувати в IT-проектах.

2 ДОСЛІДЖЕННЯ СИСТЕМИ ГЕНЕРАЦІЇ ВИМОГ

2.1 Огляд існуючих моделей генерації вимог

Аналізуючи концепцію Six Sigma [15,16], розроблену в компанії Motorolla, бізнес вимоги – це критично важливі активності підприємства, які необхідно виконати задля досягнення мети корпорації незалежно від будь-якого рішення.

З наведеного вище треба виділити критично важливі активності, а саме критерії за якими треба відокремлювати їх від інших. Для визначення рангу критичності окремої активності слід проаналізувати її та отримати список характеристик за яким можна розподілити переваги та недоліки на фоні конкурентів. Сукупність цих характеристик має виражати істинну вагу активності та дати розуміння щодо подальшого менеджменту задля досягнення поставленої мети підприємства.

Отримавши окремі активності, які тепер треба проаналізувати на предмет ефективності можна розробити вимоги, щодо покращення роботи обраних сервісів або скорочення видатків на опрацювання вже налагодженого процесу, який не потребує термінових утручань. Однак слід приймати рішення щодо формування вимог розуміючи можливі ризики, наслідком яких може бути втрата цілого сектору підприємства або можливе зменшення ефективності. При виконанні усіх вимог щодо реструктуризації процесів повинна бути збережена та досягнута мета компанії, яка була поставлена на початку реорганізації активностей корпорації.

Інший, але концептуально схожий підхід у сфері інформаційних технологій був розроблений та описаний Карлом Вігерсом та Джойом Бітті під порядкуванням компанії Microsoft «Розробка вимог до програмного забезпечення» [17]. За цим виданням формування вимог до проектів можна розділити на 7 категорій: збір інформації, аналіз, специфікації, перевірка, управління вимогами, навчання та упавління проектами. У сукупності всі категорії мають приблизно 50 прийомів для концептуального визначення вимог, однак деякі з прийомів можна віднести до кількох категорій. З цього можна зазначити, що, кожний прийом може підходити для розуміння вимог на різних

етапах та частинах проекту.

Однак треба зазначити, що чіткої класифікації щодо використання конкретних методів наразі не існує, тому у виборі оптимальних критеріїв основну роль має саме аналітик, який повинен чітко розуміти ризики від неправильно обраного методу оцінки продукту.

З огляду на моделі створення вимог можна розділити їх на типи:

– послідовний;

– ітеративний.

Ітеративна модель має перевагу в меншому проміжку часу потрібного для початку розробки або вдосконалення вже існуючої активності, але аналіз даних та формування нових вимог повинно проходити постійно з розвитком продукту. Також до переваг можна віднести можливість коректування напрямку з розвитком продукту для мінімізації можливих втрат та виявлення нових потреб кінцевих користувачів, які були не розкриті на етапі початкового проектування. До мінусів даної моделі відносять можливість різкої зміни або повної відмови від напряму у зв’язку з новими можливими ризиками. Також тривалість розробки даного продукту займає більше часу через невідому кінцеву форму продукту.

Послідовна модель має перевагу в часі розробки, через повне планування вимог перед початком процесу створення продукту, але основним недоліком такої моделі є вкрай важка зміна напрямку розробки, через яку компанія може понести великі втрати.

Кожна з моделей має різного роду переваги та недоліки які треба враховувати при обранні одної з них.

2.2 Аналіз моделі перевірки реальності проблеми

Аналіз проблеми яку повинна вирішувати програмна система є одним з найголовніших етапів визначення вимог до проекту. Для інформації щодо зацікавленості в продукті, основною метою якого буде вирішення конкретної проблеми або де-кількох проблем користувача основним інструментом є опитування потенційних користувачів. Завдяки цьому підходу де-які проблеми

будуть відокремлені та не матимуть основного пріоритету під час визначення вимог до всього проекту або завдяки результатам опитування може бути прийняте рішення не починати розробку програмного продукту через низьку зацікавленість аудиторії в рішенні даної проблеми.

За бізнес-моделлю Kano [18,19] можна визначити ступінь привабливості продукту у потенційних користувачів.

Етапами для аналізу за моделлю Kano є:

– встановлення цілі опитування;
– формування питань та відповідей до користувачів;
– проведення опитування;
– аналіз результатів опитування та побудова моделі Kano.

Ціллю опитування може бути з’ясування реакції потенційних користувачів на можливі характеристики продукту та приоритизація питань для потенційного вирішення (див. рис. 3.1).

За моделлю Kano можна виділити такі характеристики продукту:

– очікувані або базові;
– основні або бажані;
– захоплюючі або маючі вплив.

Питання повинні мати таке емоційне забарвлення: подобається, ймовірно подобається, нейтрально, можна терпіти, не подобається. Кожне питання повинно мати два варіанти, які мають бути описані в діаметрально протилежному вигляді. Через це сформовані первинні питання мають два емоційних зображення для кожного користувача, що дає значно більше розуміння потреби у подібному функціоналі.

Характеристики продукту показують реальність проблеми і її критичність для потенційних користувачів, що може бути вирішальним фактором для отримання успішного продукту.

З отриманих відповідей на запитання формується таблиця з відповідями та підраховується кількість відповідей з різним емоційним забарвленням. Для основного питання критерієм для приймання рішення щодо його реалізації стає

позитивні реакції на запитання, а для протилежного обираються негативні реакції, і якщо співвідношення має більшість серед опитаних респондентів – то таке питання було поставлено правильно та проблема дійсно існує, і вирішення її буде доцільне. Якщо ж ситуація протилежна – це говорить про неактуальність проблеми для потенційних користувачів і скоріш за все програмне рішення не буде мати успіху.

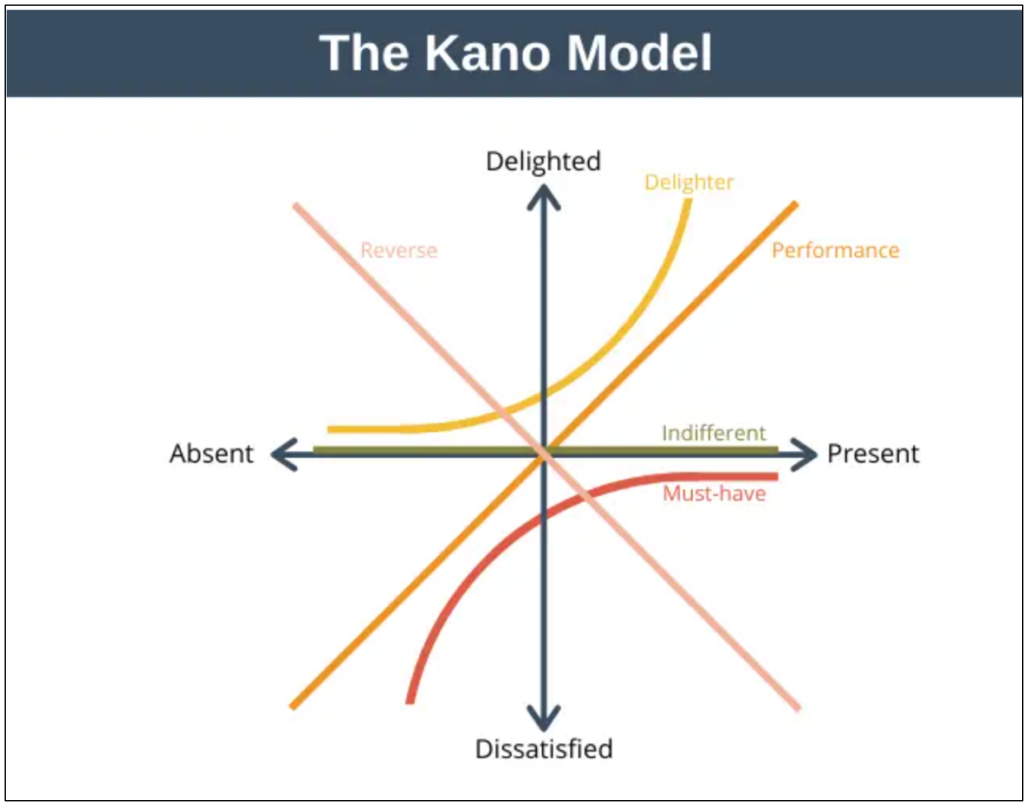

Рисунок 2.1 – Модель приоритизації Kano [20]

Бізнес модель Kano є репрезативною у випадку наявності великого обсягу вхідних даних, які були зібрані з окремих джерел і мають велику ступінь кореляції між собою. Це дає розлогу інформацію та показує, що модель матиме реальне підґрунтя і може бути основою для подальшого аналізу та обрання найважливіших вимог, які необхідні для успішної реалізації проекту, що призведе до популярності продукту і подальшому його розвитку.

2.3 Фільтрація вимог

Потреба в фільтрації вимог виникає через можливу їх неактуальність або

нереальності. Такі вимоги не нададуть позитивного результату під час або після їх виконання. З метою скорочення видатків таку процедуру рекомендовано проводити перед початком реалізації розробки програмного продукту або його ітерації [21].

Вимоги до проектів мають різні якісні характеристики, однією з яких є раціональність реалізації встановленої вимоги. Раціональними вимогами можна вважати вимоги створені за допомогою методу SMART [22].

Метод SMART має набір критеріїв за допомогою яких можна створити вимогу матиме такі характеристики:

– направлена;
– вимірюєма;
– досягаєма;
– значима;
– лімітована у часі.

Завдяки цим критеріям можливо одразу мінімізувати вплив однозначно негативних вимог на проект та опрацювати окремі критерії прийняття виконаної вимоги.

Таким чином ми можемо відхилити або переробити вимоги, які мають нечітку сферу впливу; неможливо виконати; мають низький пріоритет за моделлю Kano або не мають фіксованого часу на реалізацію.

Для виконавців проекту важливо передати вимоги, які будуть мати актуальність як мінімум для однієї ітерації життєвого циклу проекту. Завдяки чітко сформованим вимогам час на розробку програмного забезпечення буде мінімалізован, так як інженерам не потрібно буде робити повторний аналіз вимог за для успішного втілення їх у життя.

3 ПЛАНУВАННЯ ПОРІВНЯЛЬНОЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Планування за гнучкою методологією

Гнучка методологія має в своїй основі поняття невизначених вимог, або тих вимог, які можуть бути змінені протягом де-якого відрізку часу. Через це планування є ітеративним і розробка продукту є майже нескінченною. Але даний підхід дозволяє адаптувати програмний продукт під вимоги користувачів базуючись на відгуках та статистичних даних.

Для реалізації планування використовується документ “Vision and Scope” [23], головною метою якого є фіксування функціоналу та вимог які потрібно реалізувати протягом ітерації.

Документ має такі розділи:

- бізнес вимоги. Цей розділ має на меті описати проект та проблему, які треба вирішити за допомогою продукту, описує ризики та критерії успіху;
- бачення рішення. Розділ включає в собі опис головного функціоналу проекту, його залежності та припущення, отримані під час аналізу предметної галузі;
- обмеження та масштаб. Опис першої версії проекту та наступних можливих варіацій, обмеження та ліміти які має бізнес на створення мінімальної життєздатної версії та подальшого розвитку;
- бізнес контекст. Розділ описує пріоритети проекту, його основних діючих персон та навколишнє середовище.

Розроблений Vision and Scope документ (додаток г) має в собі опис реалізації MVP проекту. Де головною задачею є створення продукту, який буде вирішувати проблему відстеження змін до вимог в доступному за зрозумілому вигляді веб-застосунку, з таким мінімальним набором необхідних функцій, як:

- авторизація;
- створення вимог з докладним описом і критеріями до прийняття;

- створення проекту, та можливість додавати вимоги до існуючого проекту;
- верифікація вимоги (перенос статусу вимоги до фінальної версії, яка не повинна зазнавати змін);
- додавання коментарів до вимоги;
- можливість перегляду вимог проекту та редагування існуючих.

Після реалізації цього функціоналу проект можна вважати готовим, як для першої стадії. Подальша розробка буде мати залежність від статистичних даних користування та побажань користувачів.

Створення веб-додатку не є кінцевою метою створення продукту, через те, що проект треба презентувати до користувачів за допомогою релевантних рекламних заходів, які допоможуть заохотити велику кількість людей на використання продукту. Його масштабування з точки зору апаратної частини продукту буде проводитися в залежності від кількості одночасних користувачів та проектів.

Монетизація проекту передбачена після створення мінімальної версії продукту через додавання абонентської плати за використання додатку за яку користувач отримує збільшену кількість проектів, які можна створити в межах його організації, можливість менеджменту команди та більшу кількість функціоналу з графічним зображенням статистичних даних проектів та учасників команди.

3.2 Планування за статичною методологією

Створення застосунку за статичною методологією обумовлюється чіткими вимогами до проекту та відсутності можливості подальшого розвитку. Підтримка проекту дозволяє лише виявляти та знешкоджувати проблемні частини проекту без можливості переробки застарілих частин кодової бази.

Монетизація даного програмного застосунку буде обумовлена можливістю перманентної покупки вже готового додатку, в який одразу буде додано максимальну кількість функціоналу. Підтримка продукту можлива за допомогою

використання команди технічної підтримки, яка зможе допомагати вирішувати проблеми користувачів дистанційно.

Необхідна підготовка перед плануванням:

- виявлення проблеми та знаходження найкращого шляху вирішення її;
- обрання платформи, яка матиме всі можливості для реалізації всього функціоналу проекту;
- обрання платформи для технічної підтримки продукту.

Опис необхідного функціоналу:

- механізм авторизації через магазини додатків популярних систем, таких як MacOS [24], Windows [25];
- можливість входу в команду за унікальним кодом або створення нової команди;
- створення проектів та їх архівація;
- додавання нових вимог до проекту;
- фільтрація неверіфікованих вимог;
- сортування за алфавітом та за датою додавання або редагування;
- можливість відправити посилання на проект незареєстрованій людині;
- можливість відправляти запити до технічної підтримки;
- механізм верифікації умов за етапі створення вимоги.

Даний проект має такі обмеження:

- додаток має бути створений на платформі, яка буде підтримуватися популярними операційними системами;
- копія даних о проектах має зберігатися на пристрої користувача та віддалено на сервері;
- масштабування проекту можливо лише за допомогою додавання додаткового місця збереження даних на сервері.

3.3 Розробка діаграм Use Case до систем

Діаграма використання (use case diagram) [26] – це тип поведінкової діаграми у мові моделювання Unified Modeling Language (UML) [27], яка ілюструє

взаємодії між системою та її акторами, зосереджуючись на функціональних вимогах системи. Вона надає загальне уявлення про функціональність системи та допомагає визначити ролі або акторів, що беруть участь, а також конкретні взаємодії або використання, які вони здійснюють.

Виходячи з вимог до системи було розроблено діаграму Use Case для спланованої ітеративно (див. рис. 3.1) та статично (див. рис. 3.2) систем.

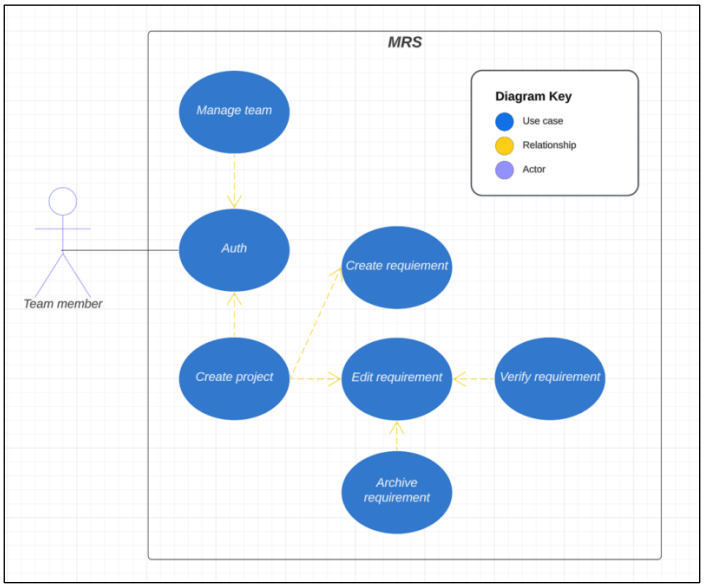

Рисунок 3.1 – Діаграма Use Case для проекту спланованого ітеративно

(Рисунок виконаний самостійно)

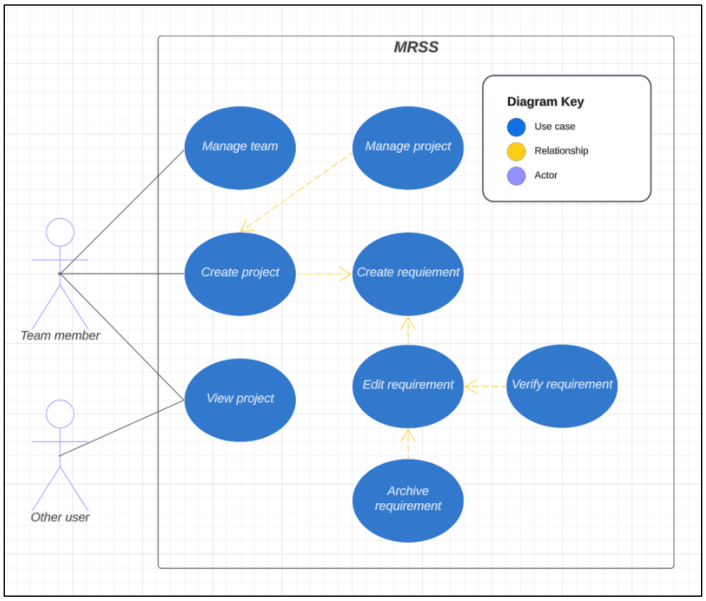

Рисунок 3.2 – Діаграма Use Case для проекту спланованого статично

(Рисунок виконаний самостійно)

Ітеративна система має включати одного актора, який одразу буде взаємодіяти з системою після авторизації. Дана система не передбачає перегляд проекту без можливості редагування, через це є обмеження, що кожен користувач має бути залакований в систему.

Система спланована статично має двох акторів, один із яких є опосередковано учасником проекту, а інший має лише доступ на перегляд його, без можливості редагування. Дана система налічує в собі можливість модифікації проекту та надання доступу до нього стороннім особам.

При порівнянні отриманих діаграм, можна визначити, що система, спланована статично має набагато більше функціоналу та потенційно має більшу привабливість для користувачів. Проте, зважаючи на те, що це є кінцевий вигляд системи, то подальша модифікація її не можлива, тому ітеративна система має перевагу у вигляді можливості адаптації під потреби користувачів та більший обсяг користувачів, через первинну безкоштовність продукту для користувача.

3.4 Планування за State Chart діаграмою та обрання бази даних

Діаграма станів (State Chart diagram) [28] – це тип діаграми у мові моделювання Unified Modeling Language (UML), яка використовується для опису поведінки об'єкта або системи та її залежності від різних станів. Вона ілюструє, як об'єкт або система реагує на події та переходить між різними станами.

Програмна система має основний функціонал, який є однаковим для обох спланованих систем (див. рис. 3.3).

Програмна система має точку входу на головну сторінку, де у користувача є вибір, або зареєструватися, або використати вже наявний аккаунт. Далі можна обрати тип дії, це може бути або операції з додавання або видалення членів команди, або перегляд проектів. На сторінці перегляду проектів можна створити новий проект або передивитися вже існуючий. Створення нових вимог зафіксовано за окремим проектом. Створити вимогу можна лише після проходження повного цикну верифікації вимоги, що є кінцевим етапом використання програмного продукту.

Створення діаграми станів є етапом планування, на якому можна визначаються дії користувачів та як вони пов’язані між собою. Іллюстрація взаємодії компонентів у системі є доказом того, що елементи поєднані у реальній та можливій системі.

Нереляційні бази даних розроблені для роботи з великими обсягами структурованої або напівструктурованої інформації. Вони пропонують гнучкі схеми дизайну та можливість горизонтального масштабування, що робить їх підходящими для певних випадків використання, де традиційні реляційні бази даних можуть бути менш ефективними або менш гнучкими [29].

Застосування нереляційної бази даних зменшує ризик при видаленні елементів, через відсутність необхідності каскадного видалення елементів, яке необхідно для коректної роботи реляційної бази даних у випадку видалення одного з ключових елементів структури.

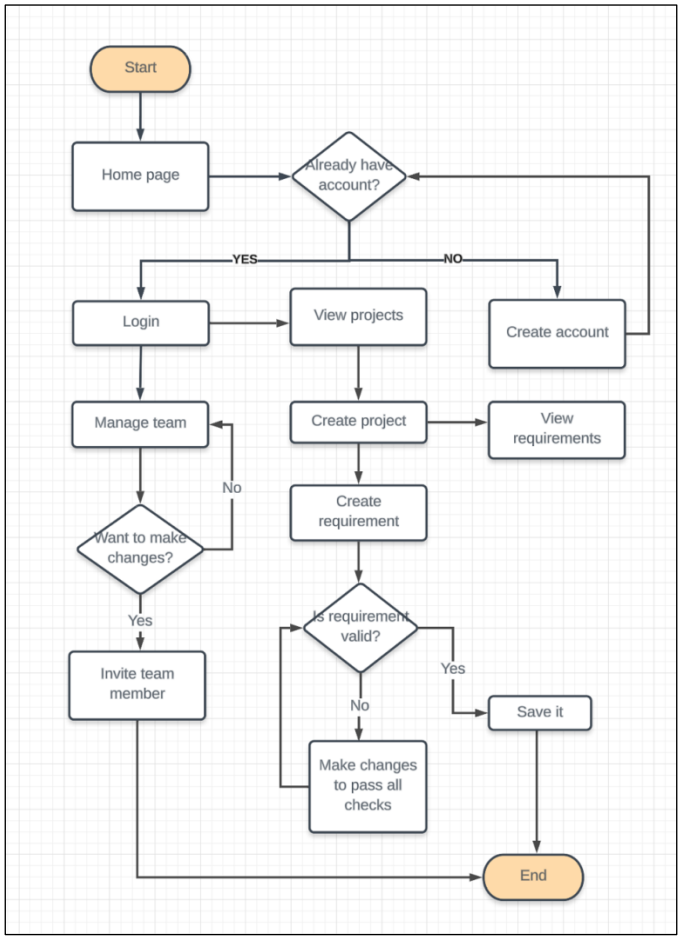

Рисунок 3.3 – Діаграма State Chart

(Рисунок виконаний самостійно)

Основні переваги нереляційних баз даних:

– продуктивність;
– гнучкість;
– відсутність схем;
– розподілена архітектура;
– великі дані та аналітика.

Основним критерієм вибору у цій роботі була гнучкість нереляційної бази даних, що дозволяє розробити застосунок, захищений від несподіваних проблем зі структурою або під час розробки.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вибір та планування бази даних

Аналізуючи вхідні дані до проекту, можна прийти до висновку, що використання реляційної бази даних є недоцільним через невеликий обсяг даних в окремій сутності та відсутності чіткого зв’язку між сутностями.

Для подальшої розробки було обрано нереляційну базу даних [30] від Firebase [31]. Функціонал даної бази даних та супутніх з нею сервісів є достатнім для реалізації проекту з менеджменту вимог в рамках IT-проектів.

Виходячи з існуючих вимог до функціоналу проекту, потребують планування сутності Вимоги, Проекту, Користувача. Зв’язок між ними є за унікальним ідентифікатором проекту, та вимоги.

Користувач, або User (див. рис. 4.1) повинен мати поля, які відображають його належність до команди та його позицію, яка буде відповідати за можливості редагування складу команди та проектів.

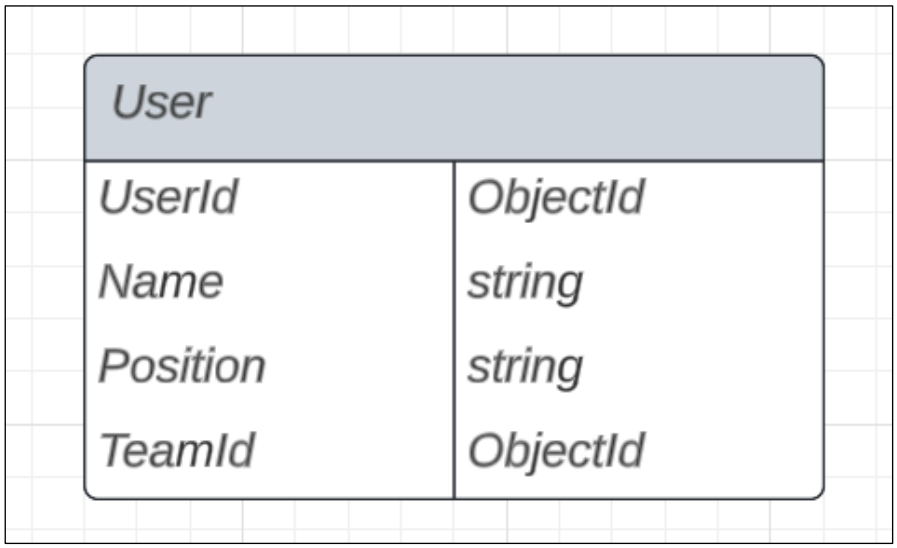

Рисунок 4.1 – Схема об’єкту «користувач»

(Рисунок виконаний самостійно)

Наступною важливою сутністю є проект (див. рис. 4.2). Ця сутність є батьківською до сутності вимоги (див. рис 4.3), вона має на меті об’єднати всі вимоги, які належать до окремого проекту та надати змогу оперувати з доступами до проекту.

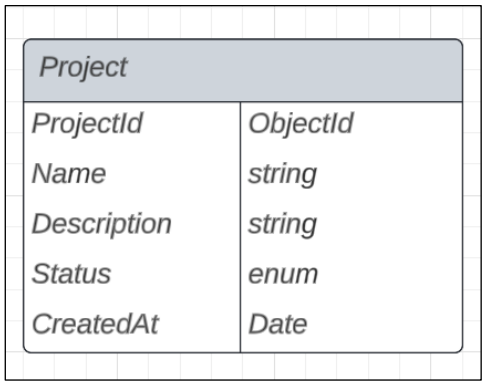

Рисунок 4.2 – Схема об’єкту «проект»

(Рисунок виконаний самостійно)

Сутність вимоги є головною, бо вона в собі має інформацію про версію, час створення та контекст функціоналу, який треба розробити. Через це вона має такі додаткові поля, як version та createdAt. Репрезентацію версій можна буде побачити за допомогою інтерфейсу користувача.

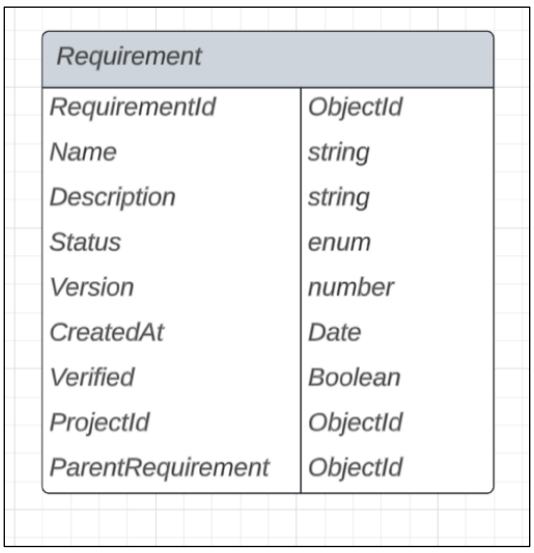

Рисунок 4.3 – Схема об’єкту «вимога»

(Рисунок виконаний самостійно)

Фільтрація та обрання окремої показує лише останній статус її. Щоб передивитися всі статуси цієї вимоги слід зробити запит до бази даних з унікальним ідентифікатором вимоги батьківської. Через те, що вимога не редагується, а лише створюється кожен раз нова, але з додатковою прив’язкою до батьківської вимоги користувач зможе отримати всю історію змін вимоги, та

матиме можливість показати їх при врегулюванні спірних запитань у якості або фінальному результаті продукту стосовно задокументованої вимоги.

Наведені вище схеми мають на меті показати вигляд окремих елементів бази даних та їх сценарії використання. Ролі окремих компонентів визначають критичність правильної реалізації даних елементів системи для її належного функціонування.

4.2 Створення інтерфейсу

Серед основних елементів програмного забезпечення є його інтерфейс. Поєднання основного функціоналу зі зрозумілою навігацією застосунком покращує UX. Через це користувачі матимуть чіткий та зрозумілий список дій, які необхідно виконати заради досягнення поставленої мети. Приклад отриманого інтерфейсу наведено на рисунку 4.4

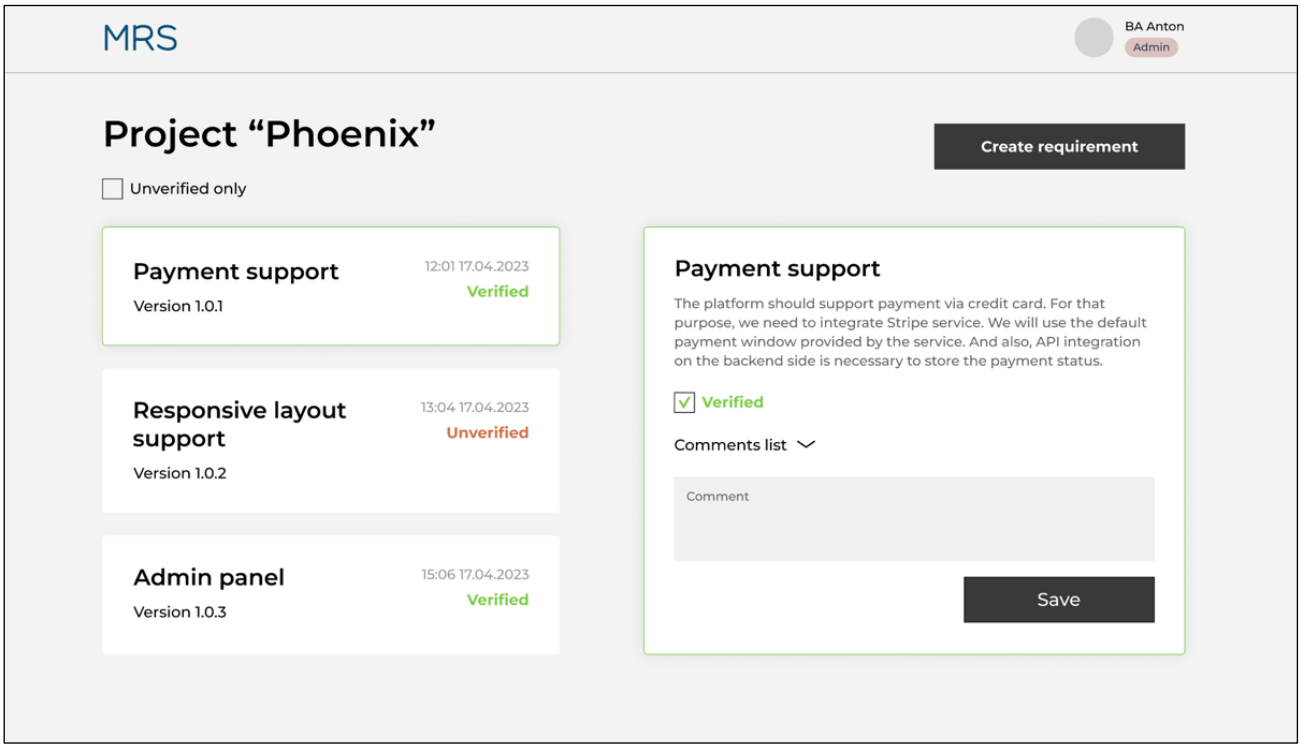

Рисунок 4.4 – Приклад сторінки проекту з вимогами

(Рисунок виконаний самостійно)

Інтерфейс застосунку виконано за допомогою мови JavaScript з використанням бібліотеки React.

4.3 Розробка логіки верифікації вимог

Основним завданням програмного продукту є створення системи, яка буде інструментом для реалізації IT-проектів з можливістю опрацювання вимог на етапі створення. Через це, послідовність дій, які треба виконати перед створенням вимоги є важливою. Відповідність до реальності, досяжності та інших SMART критеріїв має основне значення. Через це створення системи передбачає наявність перевірки. Реалізація механізму перевірки виконана у два етапи.

Алгоритмічний. Після створення вимоги, є перевірка, яка буде вважатися пройденим, якщо час на виконання завдання, і закладена складність є раціональними, а деталізація опису налічує як мінімум 150 символів.

Користувальницький. Користувач має змогу перевірити всі свої записи та після перевірки натиснути кнопку верифікації. Цей етап може завершити будь-хто з команди користувача, окрім людей, які мають доступ лише на перегляд інформації.

За допомогою даного підходу є можливість уникнути вимог, які не матимуть реальної цінності для подальшого розвитку продукту. Допоможе мати шаблонної структури вимоги, які легко зчитуються та не потребують додаткового роз’яснення.

ВИСНОВКИ

За результатом виконання курсової роботи було проаналізовано методи генерації вимог, їх позитивні та негативні якості. Було з’ясовано основні етапи роботи з програмним продуктом, а саме його плануванням та початком реалізації.

Основними етапами планування є:

– визначення проблеми, яку потенційно треба вирішити;
– з’ясування актуальності проблеми до потенційних користувачів;
– фільтрація не актуального функціоналу;
– створення мети розробки програмного продукту;
– створення функціональних та нефункціональних вимог;
– фільтрація нераціональних вимог.

Визначено, що найважливішим етапом планування є визначення актуальності проблеми для користувачів, тому що через неправильно сформовану проблему та бачення її вирішення – компанія може понести великі видатки або втратити значну частину свого бізнесу.

Було встановлено необхідність створення мети для компанії, а саме того, чого треба досягнути завдяки розробці або ітерації життєвого циклу програмного продуту. Після остаточного формування мети компанії формування вимог до програмного продукту буде відображати саме інтереси компанії, які можуть також стосуватися до побажань потенційних клієнтів.

Вирішення основної проблеми трекінгу зміни вимог вирішується за допомогою додатку. Однак доданий етап у вигляді верифікації готових вимог вирішує проблему нерелевантної зміни контексту, а версіанування створених вимог вирішує проблему не фіксованого статусу вимоги, через який можуть бути проблеми на етапі розробки та релізу проекту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Головна сторінка Softserve. URL: https://www.softserveinc.com/en-us (дата звернення: 02.03.2022)

2. Головна сторінка Globallogic. URL: https://www.globallogic.com/ (дата звернення: 02.03.2022)

3. Головна сторінка Epam. URL: https://www.epam.com/ (дата звернення: 02.03.2022)

4. Project Management Institute. Agile Practice Guide. 2017. С 50-70

5. Markus H. Agile Project Management: Scrum for Beginners. 2019. С 10-50

6. Robert M. Agile: What You Need to Know About Agile Project Management, the Kanban Process, Lean Thinking, and Scrum. 2021. С. 35-40

7. Gerardus B, Waterfall model A Complete Guide - 2019 Edition. 2019. С. 20-40.

8. Andrew S. Introducing Agile Project Management With Scrum: Why You Need To Use Scrum And How To Make It Work In Your Individual Situation. 2020. С. 100-120

9. Agile manifest. URL: https://agilemanifesto.org/iso/uk/principles.html (дата звернення 25.03.2023)

10. Agile methodology: https://indevlab.com/blog/what-is-agile-development/ (дата звернення 25.03.2023)

11. From An Idea To A Minimum Viable Product. URL: https://www.scribd.com/document/400018851/From-an-idea-to-a-Minimum-Viable-Product (дата звернення: 25.03.2022)

12. Waterfall methodology: https://manifesto.co.uk/agile-vs-waterfall-comparing-project-management-methodologies/ (дата звернення 25.03.2023)

13. Kameniev R., Golian N. A comprehensive comparison of development methodologies on a system example with a fixed cost and unapproved requirements // International scientific journal "Internauka". — 2023. — №7.

14. Головна сторінка Visure URL: https://visuresolutions.com/ (дата

звернення 5.12.2022)

15. Thomas P. The Six Sigma Project Planner: A Step-by-Step Guide to Leading a Six Sigma Project Through DMAIC. 2003. C. 50-100.

16. Michael L. , John M. , David T. ,Malcolm U . The Lean Six Sigma Pocket Toolbook: A Quick Reference Guide to Nearly 100 Tools for Improving Quality and Speed. 2004. C. 80-120

17. Karl. W, Joy B. Software Requirements (Developer Best Practices). 2019. T3. C. 30-60

18. Kano Model. URL: https://www.scribd.com/document/42615367/Kano-Model# (дата звернення: 12.12.2022)

19. Lance B. The Customer-Driven Organization: Employing the Kano Model. 2017. C. 20-70

20. Kano model image: https://expertprogrammanagement.com/2020/11/the-kano-model/(дата звернення 25.03.2023)

21. Dudar, Z., Medovoy, A., Olga, V.G. Internet-projects assessment criteria validity, problems and perspectives for proceedings of international conference CADSM 2007.The Experience of Designing and Application of CAD Systems in Microelectronics - Proceedings of the 9th International Conference, CADSM 2007, 2007, pp. 477–478, 4297623

22. How to write SMART goals. It’s easier to succeed when you have clearly defined objectives that are based in reality. URL: https://www.atlassian.com/blog/productivity/how-to-write-smart-goals. (дата звернення: 13.03.2022)

23. Teresa B. Defining the Vision and Scope for Software Projects: Effortless Business Analysis Kindle Edition

24. MacOS URL: https://www.apple.com/macos/ventura/ (дата звернення 12.04.2022)

25. Windows URL: https://www.microsoft.com/en-us/windows (дата

звернення 12.04.2022)

26. Martin F., UML Distilled: A Brief Guide to the Standard Object Modeling Language" 3rd edition p 105-108

27. Craig L, "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process" p 107

28. Jim A, Ila N. "UML 2.0 and the Unified Process: Practical Object-Oriented Analysis and Design" p 193-222

29. Bezsmertnyi O., Golian N., Golian V., Afanasieva I., “Behavior Driven Development Approach in the Modern Quality Control Process,” 2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020 – Proceedings, 2021, pp. 217–220, 9467891

30. Kuzochkina, A., Shirokopetleva, M., Dudar, Z. Analyzing and Comparison of NoSQL DBMS 2018 International Scientific-Practical Conference on Problems of Infocommunications Science and Technology, PIC S and T 2018 - Proceedings, 2019, pp. 560–564, 8632133

31. Firebase URL: https://firebase.google.com/ (дата звернення: 24.03.2023)

ПЕРЕЛІК ДЖЕРЕЛЬ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

21 Dudar, Z., Medovoy, A., Olga, V.G. Internet-projects assessment criteria validity, problems and perspectives for proceedings of international conference CADSM 2007.The Experience of Designing and Application of CAD Systems in Microelectronics - Proceedings of the 9th International Conference, CADSM 2007, 2007, pp. 477–478, 4297623

28 Bezsmertnyi O., Golian N., Golian V., Afanasieva I., “Behavior Driven Development Approach in the Modern Quality Control Process,” 2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020 – Proceedings, 2021, pp. 217–220, 9467891

29 Kuzochkina, A., Shirokopetleva, M., Dudar, Z. Analyzing and Comparison of NoSQL DBMS 2018 International Scientific-Practical Conference on Problems of Infocommunications Science and Technology, PIC S and T 2018 - Proceedings, 2019, pp. 560–564, 8632133