Хмельницький національний університет

**АНАЛІЗ ВИМОГ**

**ТА ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

*Методичні рекомендації та настанови до лабораторних робіт*

*для здобувачів першого* (*бакалаврського*) *рівня вищої освіти спеціальності 121* «*Інженерія програмного забезпечення*»

*Затверджено на засіданні кафедри*

*інженерії програмного забезпечення.*

*Протокол № 7 від 29.05.2023*

Хмельницький 2023

Аналіз вимог та якість програмного забезпечення :методичні рекомендації та настанови до лабораторних робіт для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 121 «Програмна інженерія»/ О. Г. Онишко. Хмельницький : ХНУ, 2023. 61 с.

Укладачі:Онишко О. Г., канд. пед. наук, доц.

Відповідальний

за випуск: Бедратюк Л. П., д-р фіз.-мат. наук, проф.

Випусковий редактор: Яремчук В. С.

Технічне редагування і верстка: Зварич Д. В.

Макетування та друк здійснено редакційно-видавничим відділом Хмель­ницького національного університету (м. Хмельницький, вул. Інститутська, 7/1). Підп. до друку 15.10.2023. Зам. № /23, тир. 50 прим., 2023.

© ХНУ, 2023

**ЗМІСТ**

|  |
| --- |
|  |

***Вступ*** 3

***Лабораторна робота 1***

Аналіз проблеми. Постановка завдання 6

***Лабораторна робота 2***

Розробка глосарію 10

***Лабораторна робота 3***

Пошук акторів (Actors) і варіантів використання розроблюваної системи 13

***Лабораторна робота 4***

Детальний опис ключових варіантів використання 19

***Лабораторна робота 5***

Аналіз і специфікація спеціальних вимог 22

***Лабораторна робота 6***

Документування вимог 28

***Лабораторна робота 7***

Тестування програмного забезпечення 38

***Лабораторна 8***

Технічне завдання 42

***Література***  48

***Додаток А*** 49

***Додаток Б***  50

**ВСТУП**

|  |
| --- |
|  |

Навчальна дисципліна «Аналіз вимог та якість програмного забезпечення» є однією із обов’язкових складових професійної підготовки і займає провідне місце у підготовці здобувачів першого (бакалаврського) рівня вищої освіти, які навчаються за освітньо-професійними програмами у межах спеціальності 121 «Інженерія програмного забезпечення».

***Мета дисципліни*** –полягає в забезпеченні теоретичної та практичної підготовки здбувачів, що має забезпечити отримання студентами основних знань у галузі сучасних технологій проектування, інженерії вимог до програмного забезпечення, отримання ними практичних навичок реалізації програмних систем, основ моделювання і аналізу програмних систем, аналізу розробки, специфікації та управління вимогами

***Завдання лабораторного практикуму з дисципліни*** полягає в по­глибленні знаннь про розробку та аналіз вимог, які висуваються до програмного продукту. Вміти проводити класифікацію вимог, аналізувати властивості вимог, розглядати методології, стандарти, нотації роботи з вимогами. Проводити узагальнення складових аналізу вимог: його виявлення, специфікацю та документування, верифікацію. Вміти визначити роль моделей, інструментальних засобів та процесів керування вимогами.

Відповідно до освітньої програми, вивчення здобувачем дисципліни у повному обсязі має сприяти здатності ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення, формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами замовника, технічним завданням, стандартами., дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу, накопичувати, обробляти та системати­зувати професійні знання щодо створення і супроводження програмного забезпечення та визнання важливості навчання протягом всього життя, реа­лізовувати фази та ітерації життєвого циклу програмних систем та інформа­ційних технологій на основі відповідних моделей і підходів розробки програмного забезпечення, здійснювати процес інтеграції системи, застосову­вати стандарти і процедури управління змінами для підтримки цілісності, загальної функціональності і надійності програмного забезпечення.

***Загальні методичні поради.*** Глибоке вивчення здобувачем вищої освіти теоретичного матеріалу є головною передумовою виконання лабора­торної роботи. Ґрунтовно опанувавши теоретичні відомості з теми роботи, він зможе проаналізувати, зрозуміти та пояснити отримані результати. Ступінь підготовки до виконання лабораторної роботи перевіряється шляхом усного опитування на початку заняття. Здобувач, який не засвоїв необхідного теоре­тичного матеріалу, до виконання лабораторної роботи не допускається. За ре­зультатами виконання лабораторної роботи складається звіт.

Звіт з лабораторної роботи має містити: назву, мету роботи, вихідні дані, детальний розрахунок необхідних показників, результати та висновки з виконаної роботи. Здобувач вищої освіти, який не підготував звіт з лабора­торної роботи, до її захисту не допускається.

Захист лабораторної роботи здійснюється на наступному занятті у формі усного опитування, що дає змогу найбільш об’єктивно виявити якість засвоєних знань та здатність до критичного мислення та аналізу. Для само­контролю знань до кожної лабораторної роботи розроблено перелік питань. Здобувачі, які успішно виконали всі заплановані лабораторні роботи та захистили їх, допускаються до підсумкового семестрового контролю з дис­ципліни. Пропущене з поважної причини лабораторне заняття здобувач пови­нен відпрацювати у встановлений викладачем термін.

Для виконання лабораторних робіт аудиторія має бути оснащена ком­п'ютерами, які під'єднані до інтернету, з встановленим необхідним програм­ним забезпеченням.

Оцінювання лабораторних робіт здійснюється за інституційною чо­тирибальною шкалою. Оцінки («відмінно», «добре», «задовільно», «незадо­вільно») виставляються викладачем в електронному журналі групи в елек­тронній мережі університету. Оцінка за лабораторний практикум є складовою підсумкової семестрової оцінки студента з дисципліни.

Оцінку ***«відмінно»*** отримує здобувач за грамотні, логічні і повні від­повіді на контрольні питання та якісне оформлення роботи (робота виконана правильно і повністю, без прогалин і помилок в логічних міркуваннях і обґрунтуванні рішення, з правильним використанням UML діаграм для розв'язування задач та демонстрацією практичних навичок розв'язування задач з аналізу вимог та якості програмного забезпеяення.

Оцінку ***«добре»*** отримує здобувач за повні відповіді на контрольні запитання з деякими неточностями або похибками та якісне оформлення роботи (здатність в основному правильно застосувати та аналізувати ориману інформацію. Вміти розбивати на функціональні та нефункціональні вимоги. Та робити аналіз програми відповідно до її якості згідно тих тестових випадків які заплановані)).

Оцінку ***«задовільно»*** отримує здобувач за неповні відповіді на конт­рольні запитання з неточностями, похибками та якісне оформлення роботи (здатність частково застосувати знання для побудови UML діаграм та вияв­лення вимог).

Оцінку ***«незадовільно»*** отримує здобувач за абсолютно невірні відпо­віді на контрольні питання та неякісне оформлення роботи (допущені істотні помилки при розв’язанні завдань, неосвоєна методика побудови UML діаграм та виявлення вимог).

Перед допуском до виконання лабораторного практикуму здобувачі проходять інструктаж із техніки безпеки та охорони праці з фіксацією їх під­писів у відповідному журналі, який зберігається за лабораторію.

**Лабораторна робота 1.**

**Аналіз проблеми. Постановка завдання**

|  |
| --- |
|  |

***Мета роботи***: сформувати навички: роботи з реальними замовни­ками програмних систем; навчитись ідентифікувати зацікавлених осіб і провоити інтерв'ю з ними та аналізувати отриманий матеріал; набути знань для формулювання проблеми, її актуальності і потреб зацікавлених осіб.

**1.1** **Завдання до лабораторної роботи**

1. Скласти перелік зацікавлених осіб.

2. Провести інтерв’ю та/або анкетування з кожною зацікавленою особою.

3. Проаналізувати отриману інформацію та сформулювати актуаль­ність проблеми і потреби зацікавлених осіб.

**1.2** **Теоретичні відомості і методичні рекомендації**

На етапі аналізу проблеми проводиться аналіз предметної області, для якої розробляється ПЗ.

Цілі етапу:

1) визначення меж, або контуру системи;

2) опис об'єктів автоматизації і формалізації знань про ці об'єкти;

3) виявлення або визначення потреб замовника ПЗ.

Аналіз предметної області можна проводити, наприклад, ґрунтуючись на теорії системного аналізу і використовувати запропоновані в ній методи.

Початковими даними для етапу системного аналізу є:

1) регламенти роботи відділів і посадові інструкції співробітників цих відділів;

2) анкети опитування зацікавлених осіб;

3) записи інтерв'ю із зацікавленими особами;

4) інші документи, що мають відношення до досліджуваного об'єкту.

Вихідними даними або результатом етапу системного аналізу є:

1) перелік зацікавлених осіб;

2) список потреб зацікавлених осіб в ПЗ, що розробляється;

3) опис об'єктів автоматизації;

4) модель об'єктів автоматизації або предметної області.

***Опис прикладу.*** Тут формується завдання, рішенням якого є роз­робка програмного забезпечення.

В результаті вступу України в Болонский процес в нашій дер­жаві була ініційована реформа вищої професійної освіти, у відповідності з якою Міністерством освіти і науки України була розроблена програма пере­ходу традиційної системи оцінки успішності здобувачів у систему залікових одиниць (кредитів). Це пояснюється необхідністю уніфікації систем вищої освіти з метою створення єдиного освітнього простору в тих країнах, які вже вступили у Болонский процес.

У рамках цієї програми усі ВНЗ країни повинні були до встановленої дати перейти на нову систему.. Одним із завдань переходу на нову систему в була автоматизація обліку поточної успішності і проміжних атестацій здобувачів в цілях уніфікації цього процесу на усіх кафедрах і факультетах ВНЗ, реалізації можливості автоматизованого формування звітів, публікації на сайті ВНЗ рейтингів успішності студентів.

***Складання списку зацікавлених осіб.*** Зацікавлені особи – це всі ті, хто має пряме або непряме відношення до процесу, автоматизації та особи які будуть користувачами системи. Для виявлення зацікавлених осіб необхідно відповісти на такі питання:

* хто є користувачем системи?
* хто є замовником (покупцем) системи?
* на кого ще вплинуть результати роботи системи?
* хто оцінюватиме і прийматиме систему, коли вона буде представ­лена і розгорнута?
* чи існують інші внутрішні або зовнішні користувачі системи, чиї потреби необхідно врахувати?
* хто займатиметься супроводом нової системи?
* чи не забули ми кого-небудь?

У нашому прикладі визначимо майбутніх користувачів системи – це викладачі, секретарі кафедр і деканатів, завідуючі кафедрами, системний ад­міністратор і співробітники навчального управління. Замовником нашої сис­теми є ВНЗ в особі першого проректора.

Тепер спробуємо з'ясувати, на кого ще чинитимуть вплив результати роботи нашої системи. По-перше, на здобувачів, адже рейтинги успішності усіх здобувачів будуть доступні на сайті ВНЗ. По-друге, з цієї ж причини, на батьків. По-третє, на деканів факультетів і заступників деканів по навчальній роботі, оскільки будь-які зміни в навчальному процесі торкаються їх профе­сійної діяльності.

Отже, розглянувши перші три питання, ми практично охопили усіх зацікавлених осіб. Оскільки супроводжувати систему буде розробник цієї сис­теми, то в список зацікавлених осіб ми його не включаємо. Таким чином, отримуємо наступний список зацікавлених осіб для нашої системи: викла­дачі; секретарі кафедр; секретарі деканатів; завідувачі кафедрами; навчальний відділ (в особі начальника); перший проректор; системний адміністратор (той, хто адмініструватиме нашу систему); здобувачі; батьки студентів; за­ступники деканів по учбовій роботі; декани факультетів.

***Анкетування і проведення інтерв’ю****.* Для виявлення потреб замов­ника і опису об'єктів автоматизації можна проводити як анкетування, так і інтерв'ю. Але найбільший ефект можливий тільки при проведенні і того і іншого.

Приклади анкети і переліку питань для інтерв'ю наведені в додатку А.

***Оцінка проблеми****.* Для яких проблем (прикладного типу) Ви відчу­ваєте нестачу хороших рішень? Назвіть їх.

З кожної проблемипотрібно з'ясувати наступне:

* чому існує ця проблема;
* як вона вирішується в даний час;
* як замовник (користувач) хотів би її вирішувати?

***Розуміння призначеного для користувача середовища*:**

* + які ваші навички в комп'ютерній області;
  + з якими типами додатків ви маєте досвід роботи;
  + яка платформа використовується;
  + які ваші плани відносно майбутніх платформ;
  + чи використовується ПЗ, яке має відношення до цієї проблеми (якщо так, то нехай про нього трохи розкажуть);
  + які ваші очікування відносно практичності продукту;
  + у якому виді має бути представлена довідкова інформація для ко­ристувача (у інтерактивному або друкарському)?

***Резюме*** (перераховуються основні пункти, щоб перевірити, чи усе правильно ви зрозуміли), наприклад:

* отже, ви сказали мені (перераховуєте описані замовником проб­леми своїми словами);
* чи адекватно цей список представляє проблеми, які є при існую­чому рішенні;
* які ще проблеми ви зазнаєте?

***Висновок аналітика.*** Після інтерв'ю, поки його дані ще свіжі у вашій пам'яті, зафіксуйте не менше трьох потреб або проблем з найвищими пріоритетами, виявлених вами у бесіді з цим замовником (користувачем).

Після проведення анкетування і інтерв'ювання необхідно обробити зібрану інформацію. На основі цих даних треба сформулювати перелік пот­реб замовників, побудувати модель предметної області і описати об'єкт/ об'єкти автоматизації. Усі ці результати надалі будуть використані при на­писанні технічного завдання (ТЗ) на систему, що розробляється.

Як приклад проведемо анкетування і інтерв'ю з начальником навчаль­ного відділу. Для цього скористаємося одним переліком питань, поєднавши дві ці процедури воєдино. По суті, ми проведемо тільки інтерв'ю. Приклад результату інтерв'ю (питання і відповіді), наведений в таблиці 1.1.

**Таблиця 1.1 – Інтерв’ю з начальником навчального управління**

|  |  |
| --- | --- |
| Питання | Відповідь |
| 1. Ім’я | Іванов Петро Володимирович |
| 2. Назва структурного підрозділу | Навчальне управління |
| 3. Посада | Начальник навчального управління |
| 4. Кому безпосередньо ви підпорядковуєтесь? | Першому проректору |
| 5. Які Ваші основні  обов’язки? | Забезпечення загального управління навчальною діяльністю. Забезпечення якості своєчасного вико­нання покладених на навчальне управління задач та функцій та забезпечення виконання мір по за­хисту конфіденційної та інформаційної безпеки в підрозділах управління. |

**Продовження таблиці 1.1**

|  |  |
| --- | --- |
| Питання | Відповідь |
| 6. Що ви в основному виробляєте? | Функції, що виконуються навчальним управлінням: 1. Створення єдиного нормативного-довідкового поля навчального процесу. 2. Організаційний супровід на­вчального процесу. 3. Планування ресурсів навчаль­ного процесу (об’ємів робіт кафедр, площ, чисель­ності контингенту, навантаження на умовах погодин­ної оплати, бланкової документації тощо). 4. Конт­роль за ходом організації та аналіз показників на­вчального процесу (як зовнішніх так і внутрішніх). 5. Звітність на різноманітні рівні управління. 6. Ана­ліз виконання навчального навантаження, трудоєм­кості окремих параметрів навчального процесу, ре­зультатів сесії, результатів державних атестаційних ко­місій, розробка графіка навчального процесу. 7. Скла­дання розкладу навчальних занять, екзаменів, заліків та графіка захистів. 8. Контроль зайнятості аудитор­ного фонду, складання семестрових планів, прове­дення занять, засідань державних атестаційних комі­сій, ходу сесій, діловодства факультетів, оформлення додатків до дипломів. 9. Облік руху студентів по всіх формах навчання (очна, очно-заочна та заочна, екс­тернат, друга вища освіта, навчання по скорочених програмах). |
| 7. Які документи чи яку інформацію можна вважати вхідними або необхідними для вашої діяльності? | 1. Накази та інструктивні листи Міністерства освіти України по навчально-методичних питаннях. 2. По­ложення про навчальне управління. 3. Посадові ін­струкції співробітників. 4. Накази ректора по кон­тингенту здобувачів (перші екземпляри). 5. Річні плани прийому абітурієнтів. 6. Плани роботи фа­культетів. 7. Звіти голів державних атестаційних ко­місій по всіх спеціальностях та напрямках. |
| 8. Які документи чи яку інформацію можна вважати вихідними чи результатом вашої діяльності? | 1. Семестрові плани занять. 2. Розклад навчальних занять. 3. Розклад екзаменів. 4. Звіти ВУЗа по на­вчально-методичній роботі за навчальний рік. 5. Звідні статистичні звіти ВНЗ про рух контингента здобу­вачів на початок та кінець навчального року. |
| 9. Як вимірюється ус­піх вашої діяльності? | В теперішній час відсутні кількісні показники оцінки діяльності управління |
| 10. Які проблеми впливають на успіш­ність вашої діяльності? | Проблеми, пов'язані з організацією навчального процесу. На сьогодні основною проблемою є переведення навчального процесу в систему залікових одиниць. |
| 11. Який інтерес чи які потреби у вас є від­носно майбутнього рі­шення (розроблюваного ПЗ)? | Система, що розробляється повинна бути макси­мально ергономічною, працювати стабільно (без збоїв); відгук системи не повинен викликати у користувачів роздратованості; функціональність, що реалізується повинна повністю задовольняти потреби користувача. |

***Список потреб зацікавлених осіб****.* В результаті анкетування і інтер­в'ювання усіх зацікавлених осіб були сформульовані потреби замовника ПЗ, що розроблявся. Далі необхідно провести аналогію між виявленими потре­бами і структурою і вимогами ТЗ відповідно до ДСТУ. Таким чином, потреби замовника в ТЗ можуть бути описані в розділі «Призначення і цілі створення системи».

У нашому прикладі були виявлені такі потреби:

1) уніфікувати процес оцінювання знань в системі кредитів на усіх кафедрах і факультетах ВНЗ;

2) мінімізувати суб'єктивність при оцінюванні студентів в проміжних атестаціях;

3) реалізувати можливість автоматичного формування рейтингів студентів по різних параметрах в системі кредитів;

4) реалізувати можливість формування єдиної звітності на кафедрах і факультетах.

**Питання самоконтролю**

1. Що є початковими даними для аналізу проблеми (предметної області)?

2. Що є результатом етапу системного аналізу предметної області?

3. Як визначити зацікавлених осіб?

4. Який метод збору інформації найбільш ефективний?

5. Для чого проводяться інтерв'ювання і анкетування?

***Рекомендована література*:** [***7***, c. 26–35; ***11***, c. 124–135; ***12***, c. 77–84; ***14***, c. 10–23]

**Лабораторна робота 2.**

**Розробка глосарію**

|  |
| --- |
|  |

***Мета***: набути навичок визначати та описувати основні терміни, які вико­ристовуються при проектуванні розроблюваної здобувачем системи.

**2.1. Завдання до лабораторної роботи**

Звіт з лабораторної роботи повинен складатись з таких пунктів:

1. Тема та мета.

2. Опис завдання згідно розроблюваної (на вибір) здобувачем системи.

3. Відповіді на питання з теорії та за результатами виконаної роботи.

**2.2 Теоретичні відомості і методичні рекомендації**

Глосарій є важливою частиною процесу розробки програмної сис­теми. Він служить документаційним інструментом, який містить визначення ключових термінів, що використовуються в системі. Глосарій допомагає за­безпечити єдність термінології серед учасників проекту та полегшує зрозумі­лість понять із доменної області.

Для створення глосарію при розробці програмної системи важливо виконати такі кроки:

1. Визначення основних термінів: проаналізуйте основні поняття, що використовуються в системі, та складіть список ключових термінів.

2. Пояснення значень: для кожного терміну напишіть коротке ви­значення, яке чітко пояснює його значення в контексті програмної системи.

3. Структурування глосарію: розподіліть терміни в глосарії за алфа­вітом або за семантичними групами, щоб забезпечити зручний доступ до них.

4. Документування глосарію: створіть документ, в якому буде зібрано всі терміни та їх визначення. Зазначте джерела інформації, якщо це необхідно.

5. Постійне оновлення глосарію: враховуйте зміни в програмній сис­темі та додавайте нові терміни до глосарію, щоб забезпечити його актуальність.

Створений глосарій буде важливим документом, що сприятиме уніфі­кації термінології в проекті та полегшить спілкування і розуміння між усіма учасниками розробки програмної системи.

***Приклад розробки глосарію для програмної системи «Інтернет-магазин продажу програмного забезпечення».***

Виконавець (менеджер) – фі­зична особа (фізична особа-підприємець) або юридична особа, яка діє в інте­ресах інтернет-магазину на підставі укладених договорів, угод, доручень, ви­даних довіреностей тощо, для здійснення таких операцій як: обробка замовлень, розрахунків, прийому–передачі чи відправки придбаних (замовлених) товарів.

Виробник – суб’єкт господарювання, який: виробляє товар або заяв­ляє про себе як про виробника товару чи про виготовлення такого товару на замовлення, розміщуючи на товарі та/або на упаковці чи супровідних доку­ментах, що разом з товаром передаються споживачеві своє найменування, торговельну марку або інший елемент, який ідентифікує такого суб’єкта гос­подарювання; або імпортує товар і несе відповідальність за виробництво, поставку та забезпечення відповідності кожного виготовленого нею товару.

Допустимий інтервал – терміни, в які може бути виконана (запла­нована) робота.

Завдання – атомарна одиниця планування диспетчером. В процесі по­переднього планування кожній роботі відповідає своє завдання. Завдання асо­ціюється з однією одиницею обладнання, працівником або групою.

Замовлення – заявлена замовником необхідність у виготовленні або придбанні продукції.

Інтернет-магазин – вітрина, розташована в інтернет мережі для роз­міщення, рекламування і продажу товарів, запропонованих виробниками.

Колізія – суперечлива інформація в плані, яка приводить або до не­можливості виконання плану, або до порушення обов’язків перед замовни­ком. Колізії можливі у випадках:

1) була вказана застаріла ціна;

2) неправильно вказані дати виконання замовлення.

Критичний термін виконання робіт – термін початку роботи в плані, перенесення роботи пізніше якого призводить до зриву строків відправлення замовлення.

Лендінг – сайт (сторінка) в інтернет мережі за адресою (посиланням), яка використовується власником для надання послуг виробникам товарів з розміщення, рекламування та їх продажу в інтернет-магазині.

Менеджер з продаж – атомарний людський ресурс при плануванні.

Одержувач – покупець, або уповноважена ним фізична чи юридична особа, яка на підставі відповідних документів (довіреність, договір) має право отримати придбаний покупцем товар в точці продажу, а також володіє правом провести його обмін чи повернути його з підстав передбачених цією офертою та діючим законодавством України.

Оферта – договір, який регулює відносини продажу товарів побуто­вого вжитку, що замовляються та придбаються через мережу інтернет за до­помогою вітрини та комплексу послуг, які надаються інтернет-магазином.

Покупець – фізична або юридична особа, яка має намір придбати то­вар через мережу інтернет використовуючи систему та послуги представлені інтернет-магазином на умовах дотримання цієї оферти.

Послуга – комплекс дій, які вчиняються виконавцем відповідно до оформленого покупцем за допомогою системи інтернет-магазину замовлення, для подальшої взаємодії та реалізації.

Постачальник – фізична або юридична особа яка на підставі укла­дених цивільно-правових угод, договорів, сертифікатів офіційного представ­ника (дилера) здійснює доставку товарів до пункту призначення.

Ресурс – одиниця обладнання або виконавець.

Робота – одиниця попереднього планування. Являє собою роботу на конкретному обладнанні (ресурсі) над однією одиницею продукції (замовленням).

Система – програмний комплекс (автоматизована інформаційна сис­тема) представлена інтернет-магазином покупцю, для дистанційного, спроще­ного перегляду товарів за допомогою мережі інтернет, де покупцю надається можливість ознайомитись з технічними характеристиками, інструкцією і властивостями зацікавленого товару, а також процедурою його замовлення, придбання та отримання.

Статус роботи – стан роботи з точки зору продаж. В першому на­ближенні розрізняють наступні статуси: «робота замовлення, прийнятого до обробки», «робота частково виконана», «робота виконана».

Товар – пристрій, виріб, засіб, аксесуар, а також інша продукція по­бутового вжитку, представлена виробником (постачальником) цього товару для продажу, який рекламується та реалізовується через вітрину інтернет-ма­газину за допомогою послуг системи.

**Питання самоконтролю**

1. Що таке глосарій і яку роль він відіграє в процесі розробки прог­рамної системи?

2. Які категорії термінів рекомендується включати до глосарію прог­рамної системи?

3. Які критерії ви використовуєте для визначення, які терміни слід включити до глосарію?

4. Чи є які-небудь узгоджені правила щодо форматування і структури глосарію у вашій програмній системі?

5. Як виявляється і вирішується конфлікт, якщо два терміни мають те ж саме скорочення або абревіатуру?

6. Як часто оновлюється глосарій програмної системи? Які чинники впливають на оновлення глосарію?

7. Які інструменти або програми використовуються для створення і управління глосарієм?

8. Як ви забезпечуєте однакове розуміння термінів усіма учасниками команди розробників програмної системи?

9. Чи існує процес аудиту глосарію? Як часто він проводиться і які його результати використовуються для вдосконалення глосарію?

10. Які чинники можуть впливати на використання глосарію в май­бутньому? Як планується підтримка і оновлення глосарію після завершення розробки програмної системи?

***Рекомендована література*:** [***10***, c. 11–27; ***12***, c. 14–33; ***14***; ***15***]

**Лабораторна робота 3.**

**Пошук акторів (Actors) і варіантів використання розроблюваної системи**

|  |
| --- |
|  |

***Мета***: навчитись виявляти та описувати вимоги користувачів до про­грамної системи, визначати основних акторів і формулювати варіанти вико­ристання системи.

**3.1. Завдання до лабораторної роботи**

Звіт з лабораторної роботи повинен складатись з таких пунктів:

1. Тема та мета.

2. Опис завдання згідно розроблюваної (на вибір) студентом системи.

3. Відповіді на питання по теорії та за результатами виконаної роботи.

**3.2. Теоретичні відомості і методичні рекомендації**

Суть діаграми варіантів використання полягає в тому, що система подається як сукупність сутностей або акторів, взаємодіючих із системою за допомогою варіантів використання.

Актором (actor) або діючою особою називається будь-яка сутність, що взаємодіє із системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система. Актор впливає на систему, що моду­люється, так, як визначить сам розроблювач.

Дії актора мають недетермінований характер. Таким чином, під­креслюється різниця між користувачем, як конкретною особою, й актором – роллю, яку будь-яка особа може грати в системі.

Допомогу у визначенні акторів можуть дати відповіді на деякі пи­тання, наприклад:

– у якому підрозділі організації повинна використовуватися система?

– чи може система використовувати зовнішні ресурси?

– хто буде поставляти системі ті або інші дані, звертатися до них і нести відповідальність за їхнє відновлення і видалення?

– чи здатний той самий актор грати кілька різних ролей?

– чи дозволено декільком суб'єктам здійснювати в системі однакові функції?

– чи буде система використовуватися разом з якими-небудь існую­чими успадкованими системами?

– хто має виконувати обов'язки адміністратора системи?

В загальному випадку актор завжди знаходиться поза сис­темою, його внутрішня структура ніяк не визначається. Для актора має зна­чення тільки його зовнішнє представлення, тобто те, як він сприймається з боку системи. Актори взаємодіють із системою за допомогою передачі і прийому повідомлень від варіантів використання.

Два і більш актори можуть мати загальні властивості, тобто взає­модіяти з тим самим безліччю варіантів використання однаковим образом. Така спільність властивостей і поводження представляється у виді розгляну­того відношення узагальнення з іншим, можливо, абстрактним актором, що моделює відповідну спільність ролей.

Варіант використання (use case) служить для опису сервісів, що система надає акторові. Іншими словами, кожен варіант використання визна­чає деякий набір дій, чинений системою при діалозі з актором. При цьому нічого не говориться про те, яким образом буде реалізована взаємодія акторів із системою.

Варіант використання можна представити як послідовність викону­ваних системою транзакцій, що приводить до одержання якогось відчутного результату, у якому зацікавлений визначений актор. Це означає, що після того, як система закінчить обробку запиту користувача, вона повинна повернутися у вихідний стан, у якому готова до виконання наступних запитів.

Допомогу у виборі варіантів використання системи можуть надати відповіді на наступні питання:

– які задачі вирішує кожен актор?

–·чи здатний той або інший актор створювати, зберігати, змінювати, видаляти або зчитувати дані в контексті системи?

– чи гарантують варіанти використання виконання зазначених вище функцій обробки даних?

– чи повинні актори повідомляти системі про які-небудь неперед­бачені обставини?

– чи має право актор одержувати інформацію про визначені події, що відбуваються в системі?

– які варіанти використання зв'язані з підтримкою й адмініструван­ням системи?

– чи задовольняються варіантами використання усі функціональні вимоги системи?

Між компонентами діаграми варіантів використання можуть існувати різні відносини, що описують взаємодію екземплярів одних акторів і варіантів використання з екземплярами інших акторів і варіантів. Один актор може взаємодіяти з декількома варіантами використання. У цьому випадку цей актор звертається до декількох сервісів даної системи. У свою чергу, один варіант використання може взаємодіяти з декількома акторами, надаючи для усіх них свій сервіс.

У мові UML мається кілька стандартних видів відносин між акто­рами і варіантами використання:

– відношення асоціації (assocіatіon relatіonshіp);

– ввідношення розширення (extend relatіonshіp);

– відношення узагальнення (generalіzatіon relatіonshіp);

– відношення включення (іnclude relatіonshіp).

1. Визначення понять «актор» і «програмна система»:

– актор – це сутність, яка здійснює взаємодію з програмною систе­мою і здатна виконувати певні дії;

– програмна система – це комплексність програмного забезпечення, що містить у собі необхідні компоненти для виконання певних функцій.

2. Роль акторів у розробці програмної системи:

– актори можуть бути різного типу, такі як користувачі, адмініст­ратори, системні модулі тощо;

– актори визначають вимоги до системи, взаємодіють з нею та ко­ристуються її функціями.

3. Важливість пошуку акторів при розробці програмної системи:

– пошук акторів допомагає розуміти основні потреби користувачів і визначає завдання, які система повинна вирішувати;

– результати пошуку акторів використовуються для визначення функ­ціональності та інтерфейсу програмної системи.

4. Варіанти використання розроблюваної системи:

– варіанти використання (use cases) описують можливі сценарії взаємодії користувачів (акторів) з системою та досягнення ними певної мети;

– варіанти використання допомагають встановити та документувати різні функціональні вимоги до системи.

5. Важливість аналізу акторів та варіантів використання при розробці програмної системи:

– аналіз акторів і варіантів використання дозволяє розробникам краще розуміти потреби і вимоги користувачів;

– встановлення акторів та варіантів використання є важливим етапом процесу розробки, що сприяє досягненню успішної функціональності та задоволенню користувачів.

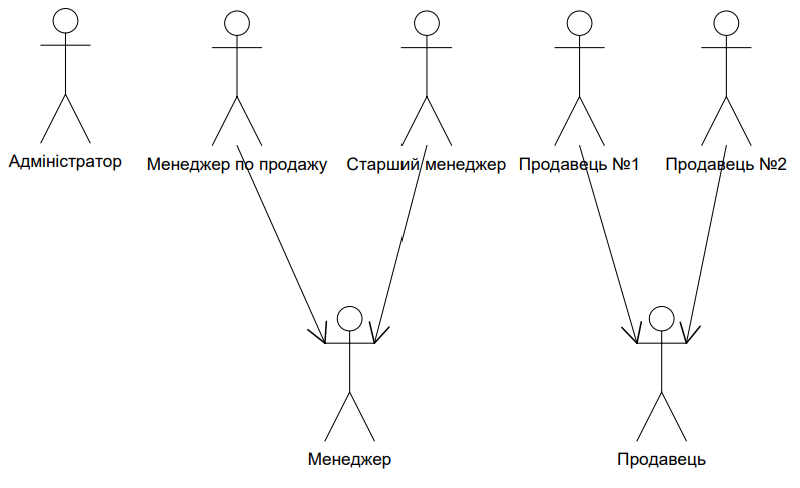
***Приклад виконання лабораторної роботи*** з покроковим описом завдань.

***Завдання 1.*** Виявлення акторів. Необхідно виявити максимально можливу кількість акторів (це можуть бути потенційні користувачі системи, а також зовнішні системи). Здійснити класифікацію акторів, розбити їх на групи; сформулювати кінцевий реєстр акторів. Кожному актору дати корот­кий (в один абзац) опис.

На рисунку 3.1 представлені основні кандидати в актори системи.

Інтерв’ю, яке було проведено з вказаними кандидатами показало, що менеджер по продажу та старший менеджер припускають використовувати програмну систему, що розробляється однотипно. Це дозволило узагальнити ці дві ролі в одну.

Аналогічна ситуація з продавцями, оскільки вони всі виконують од­накові операції. Короткий опис акторів представлено в таблиці 3.1.



**Рис. 3.1 – Аналіз акторів системи**

**Таблиця 3.1 – Короткий опис акторів**

|  |  |
| --- | --- |
| Актор | Короткий опис |
| Адміністратор | Слідкує за усіма процесами, надсилає вказівки іншим ко­ристувачам системи, прогнозує прибутки, витрати магазину |
| Менеджер | Слідкує за процесом обліку товарів, проводить ревізію, аналізує роботу продавця, замовляє нові товари |
| Продавець | Проводить облік товарів, веде статистику продажу, реєструє будь-які зміни в кількості, чи інших показниках товару |

***Завдання 2***

Виявити варіанти використання. Для кожного з акторів необхідно ви­явити максимально можливу кількість варіантів використання. Кожному з них дати коротке (в одно речення) формулювання. Виявлені варіанти викорис­тання зведені в таблицю 3.2.

**Таблиця 3.2 – Виявлення варіантів використання**

|  |  |  |
| --- | --- | --- |
| Основний актор | Найменування | Формулювання |
| Продавець | Реєстрація товару | Варіант дозволяє продавцю додати у систему новий товар |
| Продавець | Вилучення товару | Варіант дозволяє продавцю вилучити певну кількість товару з системи |
| Продавець | Формування звіту | Варіант дозволяє продавцю створити вихідний документ у системі, який показує яку кількість товару він продав за цей день |
| Менеджер | Перегляд звітності | Варіант дозволяє менеджеру пере­глянути звіти, які створили продавці |
| Менеджер | Реєстрація товару | Варіант дозволяє менеджеру додати у систему новий товар |
| Менеджер | Замовлення товару | Варіант дозволяє менеджеру замо-вити новий товар, та уточнити деталі про доставку |
| Адміністратор | Інформаційне повідомлення | Варіант дозволяє адміністратору надіслати будь-яке повідомлення будь-якому користувачу у будь-який момент часу |
| Адміністратор | Перегляд звітності | Варіант дозволяє адміністратору пе­реглядати звіти як продавців так і менеджерів магазину |
| Адміністратор | Запит  про статистику | Варіант дозволяє переглянути ста­тистику того, що було зроблено за будь-який день, кожним користу­вачем системи |

***Завдання 3***

Скласти діаграми варіантів використання. Складається одна загальна або декілька часткових діаграм варіантів використання. Загальні вимоги: кожний варіант використання і кожний актор повинні бути відображені хоча б на одній діаграмі. Всі варіанти використання показані на рисунку 3.2.

Зображення, що містить схема, ескіз, малюнок, ряд

Автоматично згенерований опис

**Рис. 3.2 – Діаграма прецедентів системи**

**Питання самоконтролю**

1. Що таке пошук акторів (actors) і які його основні принципи?

2. Які є варіанти використання розроблюваної системи в контексті пошуку акторів?

3. Які дані потрібно збирати та аналізувати для ефективного пошуку акторів?

4. Які можуть бути проблеми або виклики при пошуку акторів і як їх можна вирішити за допомогою розроблюваної системи?

5. Які можливості забезпечує розроблювана система для візуалізації та аналізу отриманих результатів пошуку акторів?

6. Які інструменти та технології використовуються у розробленні системи для пошуку акторів?

7. Як процес пошуку акторів може бути автоматизованим та оптимі­зованим за допомогою розроблюваної системи?

8. Які переваги може мати використання розроблюваної системи для пошуку акторів порівняно з традиційними методами?

9. Які можливі сценарії використання розроблюваної системи для по­шуку акторів в різних сферах діяльності?

10. Які критерії варто враховувати при виборі розроблюваної системи для пошуку акторів?

***Рекомендована література*:** [***10***, c. 15–23; ***17***, c. 27–42; ***20***, c. 12–30]

**Лабораторна робота 4.**

**Детальний опис ключових варіантів використання**

|  |
| --- |
|  |

***Мета роботи*:** навчитись розробляти опис ключових варіантів використання.

**4.1. Завдання до лабораторної роботи**

1. Здійснити аналіз ключових варіантів використання. Основні кри­терії для відбору: критична важливість для користувача (інвестора); варіант використання, як «архітектуротвірний» компонент; варіант використання, що дозволяє понизити ризики.

2. Описати основний потік подій ключових варіантів використання. Тут і далі використовувати шаблон опису на основі методології RUP.

3. Описати альтернативні сценарії ключових варіантів використання.

4. Оформити роботу.

5. Здійснити захист роботи.

**4.2. Теоретичні відомості і методичні рекомендації**

Ключовими варіантами використання є виділені види діяльності або сце­нарії, які характеризують основні способи використання певного об’єкта, технології, продукту або послуги.

Ці варіанти використання дозволяють краще розуміти та класифіку­вати різні способи використання і можуть використовуватися для аналізу, проєктування та розвитку систем, продуктів або послуг.

Описуючи ключові варіанти використання, необхідно надати де­тальні теоретичні відомості про кожний з них.

Наприклад, якщо ми розглядаємо ключові варіанти використання програмного продукту, ми можемо включити такі варіанти:

**1.** Основне використання. Описує найбільш поширені та очікувані сценарії використання програмного продукту. Наприклад, якщо це текстовий редактор, основні варіанти використання можуть включати створення, ре­дагування та збереження документів.

**2.** Розширене використання. Описує додаткові можливості або функ­ціональність програмного продукту, які можуть бути корисними для окремих користувачів або ситуацій. Наприклад, в текстовому редакторі розширені варіанти використання можуть включати форматування тексту, вставку гра­фіки або додаткові настройки.

**3.** Альтернативне використання. Описує менш типові або неочікувані способи використання програмного продукту. Наприклад, в текстовому ре­дакторі альтернативні варіанти використання можуть включати створення презентацій або розробку вебсторінок.

**4.** Інтеграція з іншими системами. Описує можливості взаємодії програмного продукту з іншими системами або продуктами. Наприклад, текс­товий редактор може мати варіанти використання для імпорту або експорту даних з інших програм або форматів файлів.

Конкретний детальний опис ключових варіантів використання буде залежати від об'єкта аналізу. Важливо провести дослідження й аналіз, щоб чітко ідентифікувати та описати ключові варіанти використання згідно з кон­текстом і потребами досліджуваного об'єкта.

Для специфікації функціональності системи будемо використовувати шаблон варіанту використання RUP.

Короткий огляд його розділів:

1) найменування і короткий опис. У цьому розділі вказується: найме­нування варіанту використання, актори варіанту використання, короткий (у один абзац) опис варіанту використання;

2) потік подій;

1. Основний потік подій так само, як в «Основний сценарій» в шаб­лоні по А. Коберну.

2. Альтернативні потоки подій. Кожен з альтернативних сценаріїв описується в окремому параграфі, в тому ж стилі, що і основний потік подій. Альтернативні сценарії описують поведінку системи при будь-яких від­хиленнях від основного сценарію, а також поведінку у виняткових ситуаціях.

3) спеціальні вимоги. Тут перераховуються нефункціональні вимоги, що мають безпосереднє відношення саме до цього варіанту використання;

4) передумови. Події, що описуються передумовами або постумовами, мають бути станами, які користувач може спостерігати. Передумова описує стан, в якому система повинна знаходитися до початку виконання прецеденту;

5) постумови. Постумова RUP по суті описує те ж, що і мінімальна гарантія у Коберна. Коректно сформульована постумова має бути істинною при будь-якому можливому сценарії прецеденту, а не описаному в основному потоці;

6) точки розширення. Цей пункт визначає положення точок, що роз­ширюють потік подій.

Методологія Rational Unified Process (RUP) є однією з найпопуляр­ніших інженерних методологій розробки програмного забезпечення. Однією з ключових складових RUP є детальний опис ключових варіантів викорис­тання. Цей елемент дозволяє докладно вивчити та описати можливі варіанти взаємодії користувачів або системи з програмним продуктом.

Основні принципи RUP:

1. ***Ітеративність та інкрементність***. Розробка в RUP відбува­ється в ітераціях, де кожна ітерація додає нові функціональності чи вдоско­налює існуючі.
2. ***Орієнтованість на архітектуру***: Архітектурна основа системи визначається на ранніх етапах та впливає на весь подальший процес розробки.
3. ***Використання каскадного та об'єктно-орієнтованого підходів***. Основна увага приділяється аналізу та проектуванню.

Детальний опис ключових варіантів використання.

1. ***Означення ключових варіантів використання* (*Use Cases*)**: Ключові варіанти використання є сценаріями, які описують взаємодію між системою та користувачами.
2. ***Структура ключового варіанту використання***:
   * ***назва***. Унікальна ідентифікація сценарію.
   * ***опис***. Повний опис сценарію використання.
   * ***актори****.* Основні учасники сценарію.
   * ***попередні умови****.* Умови, які повинні бути виконані перед почат­ком сценарію.
   * ***основні події.*** Кроки сценарію.
   * ***альтернативні події****.* Інші можливі розгляди подій.
   * ***післяумови***. Умови, які виникають після завершення сценарію.
3. ***Приклад детального опису варіанту використання***:

***Ключовий варіант використання:***

***Реєстрація нового користувача***

***Опис*:** цей ключовий варіант використання дозволяє новому користу­вачеві зареєструватися в системі та створити власний обліковий запис.

***Актори*:**

* новий користувач;
* система.

***Передумови*:**

* користувач має доступ до реєстраційної форми;
* система працює та доступна для взаємодії.

***Основні події*:**

1. Користувач натискає кнопку «Зареєструватися» на головній сто­рінці системи.
2. Система відображає реєстраційну форму.
3. Користувач вводить свої особисті дані: ім'я, прізвище, електронну пошту та пароль.
4. Користувач натискає кнопку «Зареєструватися».
5. Система перевіряє введені дані та переконується, що вони відпо­відають вимогам (наприклад, достатньо складний пароль).
6. Якщо дані валідні, система створює новий обліковий запис для користувача.
7. Система надсилає підтверджувальний лист на вказану електронну адресу.

***Альтернативні події*:**

– якщо введені дані не відповідають вимогам, система повідомляє користувача та просить виправити їх.

***Післяумови*:**

– користувач успішно зареєстрований в системі та може увійти в особистий кабінет за введеними обліковими даними.

**Питання самоконтролю**

1. Дайте означення RUP.

2. Дайте визначення основному потоку подій.

3. Дайте визначення альтернативному потоку подій.

4. Що таке передумови і постумови?

5. Дайте визначення точці розширення.

***Рекомендована література*:** [***1***, c. 6–25; ***2***, c. 12–35; ***3***, c. 7–24; ***4***, c. 100–103; ***6***; ***10***]

**Лабораторна робота 5.**

**Аналіз і специфікація вимог**

|  |
| --- |
|  |

***Мета роботи*:** набути практичних навичок для специфікування нефункціональних та функціональні вимог.

**5.1. Завдання до лабораторної роботи**

1. Написати вступну частину (цілі, посилання).

2. Описати функціональні вимоги. Вимоги описуються в простій розповідній формі (не у формі прецедентів).

3. Описати нефункціональні вимоги. Обов'язкові для розгляду кате­горії – застосовність, надійність, продуктивність, придатність до експлуатації, обмеження проектування.

4. Оформити роботу.

**5.2. Теоретичні відомості і методичні рекомендації**

Специфікація вимог є важливою частиною процесу розробки вимог. Це третій етап після збору та аналізу вимог. Мета полягає в тому, щоб створити документ або специфікацію вимог з відповідним рівнем деталізації. Цей документ міститиме всі вимоги, які мають бути накладені на проектування та перевірку продукту. Він також міститиме іншу відповідну інформацію, необхідну для розробки, перевірки та обслуговування продукту.

Специфікація вимог, також відома як документація, — це процес запису всіх вимог системи та користувача у вигляді документа. Ці вимоги мають бути чіткими, повними, вичерпними та послідовними. Це процес чіткого та точного документування всіх потреб і обмежень користувача та системи.

Переваги наявності специфікації вимог:

* Допомагає переконатися, що всі зацікавлені сторони мають спільне розуміння системи, яка має бути розроблена. Це дозволяє уникнути непорозумінь на наступних етапах розробки.
* Служить орієнтиром для всіх зацікавлених сторін у процесі розробки.
* Допомагає виявити будь-які прогалини у вимогах на ранній стадії.
* Зменшує загальну вартість і час розробки, оскільки дозволяє уникнути переробки через зміни вимог.

Вимоги користувача - це комбінація функціональних і нефункціональних вимог. Ці вимоги до користувача мають бути розроблені таким чином, щоб вони були легко зрозумілі користувачам, які не мають жодних технічних знань. Отже, вони повинні бути написані природною мовою з використанням простих таблиць, форм і діаграм. Крім того, у документі не повинно бути подробиць про дизайн системи, програмне забезпечення чи офіційні позначки.

## **Що таке нефункціональні вимоги?**

Нефункціональні вимоги (NFR) – це обмеження, накладені на сис­тему, які визначають її атрибути якості. Зазвичай вони позначаються такими прикметниками, як безпека, продуктивність і масштабованість. Не­функціо­нальні вимоги важливі, оскільки вони допомагають забезпечити від­повідність системи потребам користувача.

***Категорії нефункціональних вимог***

Нефункціональні вимоги можна розділити на дві категорії:

1. ***Атрибути якості*:** це характеристики системи, які визначають її загальну якість. Прикладами атрибутів якості можуть бути безпека, продуктив­ність і зручність використання.
2. ***Обмеження*:** це обмеження, накладені на систему. Приклади об­межень - час, ресурси та середовище.

## **Переваги нефункціональних вимог**

Є кілька переваг нефункціональних вимог:

1. Вони допомагають переконатися, що система відповідає потребам користувача.
2. Вони допомагають переконатися, що система відповідає меті.
3. Вони допомагають гарантувати, що система є масштабованою, безпечною та надійною.
4. Вони допомагають забезпечити зручність використання та обслу­говування системи.

## **Недоліки нефункціональних вимог**

Є кілька недоліків нефункціональних вимог:

1. Їх важко зрозуміти та реалізувати.
2. Вони можуть бути трудомісткими та дорогими для перевірки.
3. Вони можуть вплинути на функціональність системи, якщо їх не впровадити належним чином.

## **Які ж є переваги і недоліки нефункціональних і функціональних вимог**

[***Функціональні вимоги***](https://visuresolutions.com/uk/%D0%B1%D0%BB%D0%BE%D0%B3/%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%96-%D0%B2%D0%B8%D0%BC%D0%BE%D0%B3%D0%B8/), як випливає з назви, описують функції системи, яку необхідно проектувати. Це опис того, якою буде система і як вона буде функціонувати для задоволення потреб користувачів. Вони нада­ють чіткий опис того, як система повинна реагувати на певну команду, функ­ції та те, що очікують користувачі.

Нефункціональні вимоги пояснюють обмеження системи, що проек­тується. Ці вимоги не впливають на функціональність програми. Крім того, існує звичайна практика підкласифікації нефункціональних вимог на різні категорії:

* інтерфейс користувача;
* надійність;
* безпека;
* продуктивність;
* технічне обслуговування;
* стандарти.

Підкласифікація нефункціональних вимог є хорошою практикою. Це допомагає під час створення контрольного списку вимог, які мають відпові­дати створеній системі.

Нефункціональні вимоги такі ж важливі, як і функціональні. Якщо функціональні вимоги визначають, що повинна робити система, то нефунк­ціональні вимоги описують, як вона буде це робити. Наприклад, нова прог­рама надасть нам остаточний список усіх підключених користувачів. Це час­тина функціональних вимог. Якщо у вимозі зазначено, що система працюва­тиме лише в системах Windows і Linux, це буде частиною нефункціональних вимог.

Єдина відмінність між ними полягає в тому, що система не може функціонувати без задоволення всіх функціональних вимог. З іншого боку, система дасть вам бажаний результат, навіть якщо вона не задовольняє нефункціональним вимогам.

## **Приклади нефункціональних вимог**

Ось декілька прикладів нефункціональних вимог:

1. ***Безпека*:** система повинна бути захищена від несанкціонованого доступу.
2. ***Продуктивність*:** система повинна бути здатна обслуговувати не­обхідну кількість користувачів без будь-якого зниження продуктивності.
3. ***Масштабованість*:** система повинна мати можливість масштабу­вання за потреби.
4. ***Наявність*:** система повинна бути доступною, коли це необхідно.
5. ***Технічне обслуговування*:** система повинна бути простою в обслуговуванні та оновленні.
6. ***Портативність*:** система повинна мати можливість працювати на різних платформах з мінімальними змінами.
7. ***Надійність*:** система повинна бути надійною і відповідати вимо­гам користувача.
8. ***Зручність використання:***система повинна бути простою у використанні та зрозу­мілою.
9. ***Сумісність*:** система повинна бути сумісна з іншими системами.
10. ***Відповідність*:** система має відповідати всім чинним законам і нормам.

Нефункціональні вимоги є важливими для будь-якої системи. Вони допомагають переконатися, що система відповідає потребам користувача та здатна функціонувати за призначенням. Перед проектуванням і розробкою системи важливо ретельно розглянути всі нефункціональні вимоги.

## **Що таке збір нефункціональних вимог?**

Збір нефункціональних вимог – це процес ідентифікації та докумен­тування нефункціональних вимог до системи. Це можна зробити за допо­могою інтерв’ю, опитування, фокус-груп або іншими методами. Після того, як нефункціональні вимоги зібрано, їх можна проаналізувати та визначити пріоритети.

Процес збору нефункціональних вимог є важливою частиною роз­робки системи. Це допомагає переконатися, що всі необхідні вимоги визна­чені та що їм приділено відповідний рівень уваги. Без ретельного процесу збору нефункціональних вимог було б важко розробити систему, яка б відпо­відала потребам користувача.

## **Що таке методи виявлення нефункціональних вимог?**

Методи виявлення нефункціональних вимог використовуються для ідентифікації та документування нефункціональних вимог до системи. Існує багато різних методів, які можна використовувати, наприклад, інтерв’ю, опи­тування, фокус-групи чи інші методи. Після того, як нефункціональні вимоги зібрано, їх можна проаналізувати та визначити пріоритети.

Процес виявлення нефункціональних вимог є важливою частиною розробки системи. Це допомагає переконатися, що всі необхідні вимоги визначені та що їм приділено відповідний рівень уваги. Без ретельного про­цесу визначення нефункціональних вимог було б важко розробити систему, яка б відповідала потребам користувача.

## **Що таке нефункціональний аналіз вимог?**

Аналіз нефункціональних вимог – це процес аналізу нефункціональ­них вимог до системи. Це можна зробити, переглянувши вимоги, оцінивши їх і розставивши пріоритети. Мета аналізу нефункціональних вимог полягає в тому, щоб переконатися, що всі необхідні вимоги визначені та що їм при­ділено відповідний рівень уваги.

Аналіз нефункціональних вимог є важливою частиною розробки системи. Без ретельного аналізу нефункціо­нальних вимог було б важко розробити систему, яка б відповідала потребам користувача.

## **Найкращі практики для написання нефункціональних вимог**

Під час написання нефункціональних вимог слід дотримуватися кіль­кох найкращих практик. До них належать:

* переконайтеся, що вимоги чіткі та лаконічні;
* будьте конкретні щодо того, що потрібно;
* уникайте використання жаргону;
* використовуйте просту мову;
* переконайтеся, що вимоги досяжні;
* будьте реалістами щодо того, чого можна досягти;
* розставте вимоги за пріоритетністю;
* зберігайте вимоги гнучкими;
* за потреби перегляньте та відредагуйте вимоги;
* отримайте відгук від зацікавлених сторін щодо вимог.

Нефункціональні вимоги є важливою частиною будь-якого проекту розробки системи. Дотримуючись цих кращих практик, ви можете перекона­тися, що ваші нефункціональні вимоги є чіткими, лаконічними та досяжними.

## ***Висновок.*** Нефункціональні вимоги є важливою частиною розробки системи. Вони допомагають переконатися, що система відповідає потребам користувача та здатна функціонувати за призначенням. Збір нефункціональ­них вимог, аналіз і найкращі практики є важливими для будь-якого проекту. Дотримуючись цих вказівок, можна зробити висновок, що нефункціо­нальні вимоги є чіткими, лаконічними та досяжними.

Нефункціональні вимоги – це обмеження або вимоги, що наклада­ються на систему. Вони визначають атрибут якості програмного забезпе­чення. Нефункціональні вимоги стосуються таких проблем, як масштабова­ність, зручність обслуговування, продуктивність, портативність, безпека, на­дійність і багато інших. Нефункціональні вимоги стосуються життєво важ­ливих питань якості програмних систем.

Функціональні вимоги регламентують функціонування або поведінку системи (behavioral requirements). Функціональні вимоги відповідають на пи­тання «що повинна робити система» в тих або інших ситуаціях. Функціо­нальні вимоги визначають основний «фронт робіт» Розробника, і встанов­люють цілі, завдання і сервіси, що надаються системою Замовникові.

Нефункціональні вимоги, відповідно, регламентують внутрішні і зовнішні умови або атрибути функціонування системи. К. Вігерс виділяє на­ступні основні групи нефункціональних вимог:

* зовнішні інтерфейси (External Interfaces);
* атрибути якості (Quality Attributes);
* обмеження (Constraints).

Серед зовнішніх інтерфейсів в більшості сучасних АІС найбільш важливим є інтерфейс користувача (User Interface, UI). Крім того, виділяються інтерфейси із зовнішніми пристроями (апаратні інтерфейси), програмні інтер­фейси та інтерфейси передачі інформації (комунікаційні інтерфейси).

Основні атрибути якості:

* застосовність;
* надійність;
* продуктивність;
* експлуатаційна придатність.

Функціональні вимоги складаються з двох частин: функції та пове­дінки. Функція – це те, що виконує система (наприклад, «розрахувати пода­ток з продажу»). Поведінка залежить від того, як система це робить (наприклад, «Система розраховує податок із продажу, помноживши ціну покупки на ставку податку»).

Ось ***найпоширеніші типи*** функціональних вимог:

* Регламент підприємницької діяльності
* Вимоги до сертифікації
* Вимоги до звітності
* Адміністративні функції
* Рівні авторизації
* Відстеження аудиту
* Зовнішні інтерфейси
* Управління даними
* Законодавчі та нормативні вимоги

***Створення функціональних* *вимог*.** Створюючи функціональні ви­моги, важливо мати на увазі, що вони повинні бути конкретними, вимірними, досяжними, релевантними та обмеженими у часі. Іншими словами, функціональні вимоги повинні:

* конкретизувати, що повинна робити система;
* бути вимірними, щоб можна було визначити, чи система це робить;
* бути досяжними протягом встановленого терміну
* бути відповідними бізнес-цілям
* прив’язати себе до часу, щоб відстежувати прогрес

Дотримуючись цих вказівок, можна бути впевненим, що функціо­нальні вимоги чіткі та допоможуть команді розробників створити правильний продукт.

***Приклади*:** щоб краще зрозуміти функціональні вимоги, розглянемо кілька прикладів.

***Приклад 1***

: Користувач повинен мати можливість увійти в систему, використо­вуючи своє ім’я користувача та пароль.

У цьому прикладі функція – «вхід», а поведінка — «Система дозво­лить користувачеві ввійти за допомогою свого імені користувача та пароля».

***Приклад 2***

: система розраховує податок із продажу для покупки користувача.

У цьому прикладі функція – «обчислити податок із продажу», а пове­дінка – «Система обчислить податок із продажу, помноживши ціну покупки на ставку податку».

***Приклад 3***

: система надішле користувачеві електронний лист із підтвердженням після того, як він успішно розмістить замовлення.

У цьому прикладі функція – «надіслати електронний лист із підтвер­дженням», а поведінка – «Система надішле електронний лист із підтверджен­ням користувачеві після того, як він успішно розмістить замовлення».

Як бачите, функціональні вимоги – це конкретні твердження про те, що повинна робити система. Вони відрізняються від нефункціональних ви­мог, які визначають, як система працює всередині (наприклад, продуктив­ність, безпека тощо). Створюючи функціональні вимоги, важливо мати на увазі, що вони повинні бути конкретними, вимірними, досяжними, релевант­ними та обмеженими у часі (SMART). Дотримуючись цих вказівок, ви можете бути впевнені, що функціональні вимоги чіткі та допоможуть команді розробників створити правильний продукт.

Обмеження – формулювання умов, що модифікують вимоги або на­бори вимог, звужуючи вибір можливих рішень по їх реалізації. Вибір плат­форми реалізації і розгортання (протоколи, сервери додатків, баз даних), які, у свою чергу, можуть відноситися, наприклад, до зовнішніх інтерфейсів.

**Питання самоконтролю**

1. Дайте визначення функціональним вимогам.

2. Дайте визначення нефункціональним вимогам.

3. Назвіть основні групи нефункціональних вимог.

4. Охарактеризуйте кожну групу нефункціональних вимог

***Рекомендована література*:** [1***7***, c. 15–77; ***19***, c. 120–130; 23, c. 12–28]

**Лабораторна робота 6.**

**Документування вимог**

|  |
| --- |
|  |

***Мета роботи*:** набути навичок в описі зовнішньої поведінки системи, а також вимог і обмежень в документі «Специфікація вимог програмного забезпечення».

**6.1. Завдання до лабораторної роботи**

1. Розробити документ SRS ґрунтуючись на методології RUP.

2. Оформити роботу.

3. Здійснити захист роботи.

**6.2. Теоретичні відомості і методичні рекомендації**

Документ специфікації вимог до програмного забезпечення (SRS) є важливим документом для розробки програмного забезпечення, який містить детальний опис потреб і вимог конкретного проекту. У ньому викладено цілі, обсяг, довідкову інформацію, деталі проекту, план впровадження та інші пов’язані дії. Документ SRS служить контрактом між замовником і роз­роб­ником, щоб гарантувати, що обидві сторони розуміють специфікації та очіку­вання продукту, що розробляється. Крім того, це допомагає зменшити ризики, гарантуючи, що всі зацікавлені сторони повністю розуміють, чого від них очікують на кожному етапі проекту.

Добре складений документ SRS має бути повним, чітким і лаконіч­ним, щоб його легко зрозуміли як розробники, так і клієнти. Крім того, наявність SRS гарантує, що будь-які зміни або оновлення продукту під час розробки можна легко задокументувати та відстежити. Це допомагає мінімі­зувати плутанину та гарантує, що кінцевий продукт відповідає всім вимогам, зазначеним у документі. Загалом SRS є критично важливим інструментом для успішних проектів розробки програмного забезпечення, оскільки він забезпе­чує чітку основу для успіху. За правильного використання це може допомогти командам досягти якісних результатів з мінімальними зусиллями.

***Відміності між специфікаціми вимог до програмного забезпечення та специ­фікаціями бізнес-вимог*.** Люди іноді змішують поняття програмного забезпе­чення та специфікації бізнес-вимог. Насправді вони обидва досить різні.

Основна відмінність між специфікацією вимог до програмного забезпечення та специфікацією бізнес-вимог розглянуто в таблиці 6.1.

Таблиця 6.1 – Аналіз специфікації вимог

|  |  |  |
| --- | --- | --- |
| №  з/п | Специфікація вимог до програм-ного забезпечення (SRS) | Специфікація бізнес-вимог (BRS) |
| 1 | Він визначає функціональні та не-функціональні вимоги до прог-рамного забезпечення. | Це офіційний документ, який опи-сує різні вимоги, надані клієнтом/-зацікавленими сторонами. |
| 2 | Він походить від специфікації бізнес-вимог (BRS). | Він випливає з вимог і взаємодії клієнта. |
| 3 | Його створює системний аналі-тик, або системний архітектор, або бізнес-аналітик. | Його створює бізнес-аналітик. |
| 4 | Він також на високому рівні опи-сує як технічні, так і функціональні характеристики програмного забез-печення. | Він описує функціональні характе-ристики програмного забезпечення на дуже високому рівні. |
| 5 | Він має справу з ресурсами, які надає компанія. | Він стосується вимог бізнесу. |
| 6 | Він описує, як функціонує бізнес під час використання програм-ного забезпечення чи програми. | Він визначає потреби клієнта. Документ використовується від по-чатку до кінця проекту. |
| 7 | Таблиці та випадки використання включені. | Таблиці та випадки використання не включені. |

Вона полягає в тому, що перша фіксує всю інформацію, пов’язану з програмним забезпеченням, тоді як остання фіксує всю інформацію, пов’я­зану з бізнесом.

***Основні компоненти SRS***

Основні розділи специфікації вимог до програмного забезпечення:

* ***бізнес*-*драйвери*** – у цьому розділі описано причини, чому клієнт хоче побудувати систему. Цей розділ також містить проблеми, з якими клієнт стикається з поточною системою, і можливості, які надасть нова система;
* ***бізнес-модель*** – у цьому розділі обговорюється бізнес-модель, яку система має підтримувати. Він також містить різні деталі, як-от організацій­ний і бізнес-контекст, основні бізнес-функції та діаграми процесів системи;
* ***функціональні та системні вимоги*** – у цьому розділі зазвичай детально описуються вимоги, організовані в ієрархічній структурі. Функціо­нальні вимоги знаходяться на верхньому рівні, а детальні системні вимоги перераховані як підпункти;
* ***випадки* *використання* *системи*** – цей розділ містить діаграму варіантів використання Уніфікованої мови моделювання (UML), яка пояснює всі ключові зовнішні об’єкти, які взаємодіятимуть із системою, і різні варі­анти використання, які вони повинні виконувати;
* ***технічні вимоги*** – у цьому розділі обговорюються всі нефункціо­нальні вимоги, які складають технічне середовище, і технічні обмеження, в яких працюватиме програмне забезпечення;
* ***системні якості*** – у цьому розділі визначаються численні якості системи, такі як надійність, придатність до обслуговування, безпека, масшта­бованість, доступність і зручність обслуговування.
* ***обмеження та припущення*** – у цьому розділі описані всі обме­ження, накладені на дизайн системи з точки зору клієнта. Тут також обгово­рюються різні припущення команди інженерів щодо того, чого очікувати під час розробки;
* ***критерії прийняття*** – детально про всі умови, які повинні бути виконані перед передачею системи кінцевим споживачам, обговорюються в цьому розділі.

***Структура SRS***

1 Введення

У вступі пояснюється загальне значення SRS, її сфера застосування для вашої команди та її структура.

1.1 Мета

Тут поясніть мету та структуру програмної документації SRS: типи вимог, які будуть розглянуті, а також персонал, який використовуватиме його.

Цей розділ має буде коротким: достатньо 1–2 абзаци.

1.2 Цільова аудиторія

В цьому розділі детально пояснюється, як зацікавлені сторони та команди працюватимуть із SRS, а також яку участь будуть приймати у її розробці. Зазвичай це власники продуктів, інвестори, бізнес-аналітики, розробники, іноді тестувальники та робочий персонал. Уся структура визначається підходом до розробки програмного забезпечення та організаційною структу­рою команди.

1.3 Цільове використання

Тут описується, у яких ситуаціях команда використовуватиме SRS. За­звичай він використовується в таких випадках:

* проектування та мозковий штурм нових функцій;
* планування тривалості проекту, спринтів, оцінка витрат;
* оцінка ризиків;
* моніторинг та вимірювання успішності команди;
* конфліктні ситуації, коли залучені сторони мають різні бачення добре виконаного продукту.

1.4 Область застосування

Ця частина охоплює сферу застосування продукту, тому потрібно дати короткий огляд системи – її основне призначення, функції та положення. Це можна порівняти з тим, як пояснити продукт на зустрічі зацікавлених сторін, за винятком того, що дозволяється глибше заглиблюватися в технічні особливості.

Цей розділ повинен описувати:

– всі типи користувачів які будуть взаємодіяти з системою;

– усі важливі частини архітектури.

1.5 Визначення та акроніми

У документі команда часто використовує певні слова. Усунути будь-які потенційні непорозуміння, дозволити новим розробникам приєднатися та вирішити конфліктні ситуації – усе це стане легше, якщо з’ясувати значення слів.

Зазначені компоненти складають визначення. Визначення надають інформацію про функцію, базові технології, цільові персони, суб’єктів господарювання (користувачів, клієнтів, посередників) і зацікавлених сторін. Можна використовувати абревіатуру, щоб швидше написати свій SRS, якщо виникає потреба. Документ можна буде прочитати, доки він міс­титься в таблиці визначень.

2 Загальний опис.

У другій частині описуються основні функції продукту, цільових користувачів і масштаби системи. Цей опис зосереджується лише на ключових функціях і архітектурі програмного забезпечення, не вдаючись до особливостей додаткових компонентів і з’єднань.

2.1 Потреби користувача

Ця частина є питанням вибору, тому деякі організації вирішують не включати її до своєї технічної документації SRS. Існкє думка, що краще зараз перерахувати проблеми, які мають бути вирішені за допомогою своїх функцій. Це стане в нагоді пізніше під час мозкового штурму та функцій мо­ніторингу. Можна повернутися до цього розділу в будь-який час під час процесу розробки продукту та перевірити, чи не збилася команда з розробки користувацького досвіду з наміченого шляху.

Потреби стосуються питань, які користувачі зможуть вирішити за допомогою системи. Можна розділити ці потреби на підкатегорії, якщо це дуже сегментована аудиторія. Не потрібно вдаватися в подробиці потреб кожного користувача. Потрібно залишити певний простір для тлумачення на випадок, якщо проблема виявиться більш суттєвою, ніж спочатку думали.

2.2 Припущення та залежності

Припущення — це припущення команди щодо продукту та його можливостей, які будуть правильними в 99% ситуацій. Природно припустити, наприклад, що платформа, яка допомагає водіям орієнтуватися вночі, буде використовуватися переважно в нічному режимі.

Яке значення мають припущення? Вони дозволяють спершу зо­середитися на найважливіших функціях програми. Це припущення допомагає зрозуміти, що дизайнери повинні розробити інтерфейс, придатний для ба­чення в темряві для асистента водіння вночі. Деякі користувачі, звичайно, можуть відкрити програму протягом дня, але це довго, тому не потрібно відразу включати пов’язані елементи в прототип.

3 Системні функції та вимоги

У цій частині детально описано характеристики продукту та критерії виконання. Оскільки попередні два розділи стосуються продукту в цілому, тут розміщений більш повний опис.

3.1 Функціональні вимоги

Зазначені в списку функцій, які виконуватимуться в системі. Ці кри­терії стосуються того, «що буде створено?» а не «як» і «коли».

Функціональні вимоги починаються з опису функціональних можли­востей, необхідних на основі того, наскільки вони важливі для про­грами. Якщо запланованоспочатку попрацювати над цим, то можна почати з дизайну, але потім слід перейти до розробки. Функціональні вимоги не вдаються в деталі технологічних стеків, оскільки вони можуть змінюватися в ході проекту. Замість того, щоб зосереджуватися на внутрішній логіці, функціо­нальні вимоги зосереджуються на функціях кінцевого користувача.

**3.2 Вимоги до зовнішнього інтерфейсу**

Функціональні вимоги є значною частиною специфікації системних вимог. Щоб охопити всі необхідні функції системи, знадобиться 4–5 сто­рінок інформації. Деякі команди розбивають їх за темами, щоб полегшити читання документа.

Зазвичай компоненти дизайну SRS називають окремими від сервер­ної частини та бізнес-логіки. Це має сенс, оскільки дизайнери, а не розроб­ники займаються більшістю цієї сфери, а також тому, що саме тут почина­ється процес розробки продукту.

Залежно від проекту вимоги до зовнішнього інтерфейсу можуть скла­датися з чотирьох типів:

* користувацький інтерфейс;
* програмний інтерфейс;
* апаратний інтерфейс;
* інтерфейс зв'язку.

Вимоги до зовнішнього інтерфейсу описують елементи сторінки, які будуть видимі кінцевому клієнту. Вони можуть включати список сторінок, елементи дизайну, ключові стилістичні теми, навіть художні елементи тощо, якщо вони важливі для продукту.

**3.3 Системні вимоги**

Системні вимоги продукту визначають умови, за яких його можна використовувати. Зазвичай вони стосуються специфікацій і функцій облад­нання. Вимоги до обладнання SRS часто визначаються мінімальними та максимальними діапазонами, а також оптимальним порогом продуктивності продукту.

Створення системних вимог перед початком створення продукту може здатися важким, але це важливо. Розробники повинні дотримуватися вимог до апаратного забезпечення, щоб пізніше їм не довелося перезапускати проект. Мобільні програми (з багатьма змінними, які слід враховувати) і програми, які потребують високої реактивності, особливо вразливі.

**3.4 Нефункціональні вимоги**

Для багатьох організацій ця частина SRS є найскладнішою. Якщо функціональні вимоги стосуються питання, що створити, то нефункціональні стандарти визначають, як. Вони встановлюють критерії відповідно до того, наскільки ефективно повинна працювати система. Порогові значення продук­тивності, безпеки та зручності використання включені в цю область.

Справжня цінність – це те, що ускладнює визначення нефункціо­нальних вимог. Визначити такі фрази як «паралелізм» або «переносність» важко, оскільки вони можуть мати різні тлумачення для всіх залучених сторін. Тому потрібно давати оцінку кожної нефункціональної вимоги. Можна будь-коли переглянути вимоги свого проекту, щоб перевірити, чи відповідає поточна система початковим очікуванням.

***Яких помилок слід уникати під час розробки специфікації систем­них вимог*?**

У міру того, як навички розробки SRS зростатимуть, процес при­скорюватиметься. Тим не менш, на початку, доцільно мати під рукою список типових помилок для довідки. З цією метою розглянемо помилки, які зустрічаються найчастіше:

1. ***нехтування включенням вичерпного глосарію*:** перегляньте чи містить ваша SRS набір термінів, з яким знайомі лише люди в галузі? Якщо так, створіть простий для розуміння розділ словника та додайте визначення будь-яких слів або виразів, які не є широко відомими. Це допоможе гарантувати, що всі користувачі зможуть зрозуміти як мету документа, так і термінологію.
2. ***створення безладу шляхом поєднання ідей*:** структуруйте свій документ упорядковано та переконайтеся, що інформація доводиться до читачів у логічній послідовності. Щоб уникнути непорозумінь або плутанини, не плутайте поняття разом у тексті.
3. ***незнання потреб і бажань цільової аудиторії*:** щоб зрозуміти очікуваний вихід програмного забезпечення, важливо розрізнити, хто його використовуватиме, а також які результати очікуються. Наприклад, створено додаток для створення звітів. Деякі з його вимог мають передбачати те, як користувачі можуть натискати певні кнопки, щоб отримати різні документи. Щоб справді зрозуміти, що вимагається від цієї програми звітності, а також визначити, хто її використовуватиме, потрібно мати уявлення про користувача та його очікування щодо функціональності.
4. ***бути неоднозначним*:** переконайтеся, що ваші потреби одноз­начні. SRS сформульовано, щоб уникнути непорозумінь, тому важливо пере­конатися, що документ не створює плутанини. Для кожного опису функції чи стану потрібно переконатися, що не включено ті функції, які ще не визначено.

***Найкращі практики оформлення документів з*** ***SRS.*** Написання до­кумента специфікації системних вимог (SRS) є ключовим аспектом роз­робки програмного забезпечення, і дотримання найкращих практик може значно підвищити якість і ефективність документа. Ось кілька найкращих практик щодо написання документів SRS:

1. ***Використовуйте чітку та стислу мову:*** уникайте технічного жаргону та абревіатур, які можуть бути незрозумілими для всіх. Використо­вуйте чітку та зрозумілу мову, щоб усі зацікавлені сторони могли легко зрозу­міти зміст.
2. ***Додайте наочні посібники:*** покращте розуміння, додавши діаг­рами, блок-схеми та макети разом із текстовим описом вимог. Візуальні по­сібники можуть забезпечити більш інтуїтивне представлення складних кон­цепцій і поведінки системи.
3. ***Пріоритетні вимоги:*** чітко визначте пріоритетність кожної ви­моги. Використовуйте такі мітки, як «необхідно», «необхідно мати» та «при­ємно мати», щоб вказати відносну важливість різних функцій. Пріоритезація допомагає команді розробників спершу зосередитися на критичних функціях.
4. ***Тримайте його оновленим:*** підтримуйте контроль версій для до­кумента SRS. Регулярно оновлюйте документ, щоб відобразити будь-які зміни у вимогах проекту, обсягу або відгуках зацікавлених сторін. Ведіть чіткий журнал змін, щоб відстежувати зміни з часом.
5. ***Залучення зацікавлених сторін:*** тісно співпрацювати з усіма від­повідними зацікавленими сторонами протягом усього процесу розробки SRS. Беріть участь в обговореннях, збирайте відгуки та переконайтеся, що їхні потреби та очікування точно відображені в документі. Така участь сприяє спільному розумінню цілей проекту.
6. ***Будьте комплексними:*** не залишайте місця для інтерпретацій чи припущень. Надайте детальний і вичерпний опис кожної вимоги, включаючи функціональні та нефункціональні аспекти. Неоднозначність вимог може призвести до непорозумінь і затримок проекту.
7. ***Використовуйте структурований формат***: організуйте доку­мент SRS у чітко визначені розділи, такі як Вступ, Вимоги до зацікавлених сторін, Архітектура системи, Функціональні вимоги, Нефункціональні ви­моги, Обмеження, Припущення, Залежності та Матриця відстеження. Струк­турований формат полегшує читачам пошук певної інформації.
8. ***Переконайтеся в тестованості***: напишіть вимоги таким чином, щоб полегшити тестування та перевірку. Кожна вимога має бути перевіреною, дозволяючи тестувальникам створювати тестові випадки, які підтверджують, чи відповідає система заданим критеріям. Необхідні чіткі критерії прийняття для кожної вимоги.
9. ***Уникайте двозначності***: будьте пильні щодо усунення неодноз­начності у вимогах. Використовуйте чітку мову, уникайте розпливчастих тер­мінів і переконайтеся, що немає місця для багаторазового тлумачення вимоги. Неясності можуть призвести до непорозумінь і переробки проекту.
10. ***Розглянемо майбутню масштабованість:*** подумайте про довго­строкову масштабованість програмної системи. Передбачте потенційні май­бутні потреби та переконайтеся, що документ SRS їх враховує. Такий про­активний підхід може заощадити час і ресурси в майбутньому.
11. ***Перегляньте та перевірте:*** проведіть ретельний аналіз доку­мента SRS із зацікавленими сторонами, включаючи клієнта, команду розроб­ників та експертів у відповідній галузі. Усуньте будь-які розбіжності, невідпо­відності або двозначності, які виникають під час процесу перегляду. Пере­вірка гарантує, що документ точно представляє цілі проекту.
12. ***Отримати офіційне схвалення***: після остаточного оформлення документа SRS отримайте офіційне схвалення та підпис від клієнта або спон­сора проекту. Це формалізує угоду щодо обсягу та вимог проекту, створюючи чітку основу для розробки.

Включення цих передових практик у процес написання документів SRS може сприяти успіху проектів розробки програмного забезпечення. Добре розроблений документ SRS служить надійним посібником як для ко­манди розробників, так і для зацікавлених сторін, допомагаючи переконатися, що кінцева система програмного забезпечення відповідає очікуванням і вимогам.

Написання ефективного документу специфікації систем­них вимог (SRS) є критично важливим кроком у процесі розробки програм­ного забезпечення. Він служить основою для успішного планування, роз­робки, тестування та перевірки проекту.

Щоб вимоги, виявлені і описані, прийняли силу угоди між замовни­ком і розробником, їх необхідно оформити у вигляді документу.

Шаблон «Специфікація вимог програмного забезпечення» (SRS), за­пропонований в RUP, по суті є контейнером, в який необхідно «упакувати» артефакти, отримані в процесі специфікації вимог. Крім того, SRS частково перекликається з документом «Бачення». Шаблон зручний своєю компакт­ністю і лаконізмом.

Шаблон SRS по RUP:

1. Вступ.

1.1. Мета. Документ повинен вичерпним чином описувати зовнішню поведінку системи, а також нефункціональні вимоги і обмеження.

1.2. Коротке зведення можливостей.

1.3. Визначення, акроніми і скорочення.

1.4. Посилання.

1.5. Короткий зміст.

2. Огляд системи

2.1. Огляд прецедентів. Містить список імен і коротких описів варі­антів використання і акторів з ілюстраціями у вигляді діаграм прецедентів.

2.2. Припущення і залежності. Ця секція описує ключові технічні можливості, компоненти, підсистеми, пов'язані проекти, які можуть впливати на життєздатність системи, що розробляється. Припущенням (assumption) називається положення, яке вважається істинним за відсутності доказу або визначальної інформації. При визначенні залежностей (dependencies) проекту від зовнішніх чинників, необхідно проаналізувати, які нові операційні сис­теми, регламенти бізнес-процесів, стандарти якості, інформаційні системи можуть з'явитися на підприємстві впровадження і як це може вплинути на функціонування автоматизованої інформаційної системи, що виготовляється.

3. Опис вимог

3.1. Опис варіантів використання. Параграф містить опис варіантів використання і пов'язаних з ними нефункціональних вимог, або посилання на відповідні артефакти.

3.2. Спеціальні вимоги. Параграф містить опис функціональних ви­мог (не описаних в якості варіантів використання), а також опис нефунк­ціональних вимог загального характеру (не зіставлених жодному прецеденту в попередньому розділі), або посилання на відповідні артефакти.

4. Допоміжна інформація. Сюди включається інформація, що полег­шує розуміння документу. Це може бути зміст і додатки, наприклад, прото­типи призначеного для користувача інтерфейсу, що описують.

**Питання самоконтролю**

1. Що таке SRS?

2. Навіщо використовується SRS?

3. Які розділи містить SRS?

***Рекомендована література*:** [***5***, c. 5–25; ***6***, c. 14–35; ***8***, c. 9–30; ***10***, c. 100–103; ***12***; ***13***]

**Лабораторна робота 7.**

**Тестування програмного забезпечення**

|  |
| --- |
|  |

***Мета роботи***: набути навичок роботи з основними принципами тес­тування програмного забезпечення та використовувати різні техніки тестування для покриття програмного коду.

**7.1. Завдання до лабораторної роботи**

1. Напишіть тест-кейси для покриття функціоналу програмного коду.

2. Виконайте автоматичні тести для перевірки працездатності про­грамного застосунку.

3. Проведіть аналіз результатів тестування та запишіть усі виявлені помилки у звіт.

**7.2. Теоретичні відомості і методичні рекомендації**

Тестування програмного продукту є невід’ємною частиною процесу створення проекту, і спрямоване на те, щоб надати замовнику винятковий досвід користувача, що відповідає їх очікуванням – без багів, помилок та інших недоробок.

***Перевірка вимог*** – це процес підтвердження того, що системні ви­моги містять усі необхідні елементи добре написаних вимог. Перевірка вимог є критично важливим кроком у розробці програмного забезпечення, оскільки вона допомагає переконатися, що система відповідає своїм цілям і функціям за призначенням.

Перед проектуванням вимоги мають бути підтверджені та затвер­джені, щоб запобігти переробці. Якщо критерії не перевіряються, як пере­вірка вимог, так і перевірка продукту неминуче будуть виконані під час роз­робки та створення продукту. Оскільки перевірка здійснюється відповідно до вимог, існує велика ймовірність того, що несправні або відсутні вимоги не будуть знайдені. Відсутні або неправильні вимоги можуть при­звести до того, що продукти не відповідають очікуванням клієнтів. Пере­вірку вимог важливо проводити на ранній стадії та часто, щоб запобігти цим проб­лемам.

Основними цілями перевірки вимог є забезпечення повноти, пра­вильності та узгодженості системних вимог.

Цей етап може виявити відсутні вимоги або недійсні, зменшуючи переробку та перевитрати коштів. Набагато ефективніше вирішити невелику проблему заздалегідь, ніж у майбутньому, коли потрібно буде відстежити та виправити сотні рядків коду.

Перевірка вимог необхідна, тому що вона допомагає переконатися, що система відповідає своїм цілям і функціям за призначенням. Неповні, неправильні або суперечливі вимоги можуть призвести до проблем під час розробки програмного забезпечення, тестування та розгортання.

Перевірка – це процес, який використовується для перевірки відпо­відності системі. Валідація відповідає на запитання: «Чи ми будуємо правильну систему?» Йдеться про тестування та перевірку системи та ви­значення правильності створеної нами системи та чи відповідає вона очіку­ванням клієнта чи ні. Різні методи, які використовуються для перевірки системи, включають тестування чорної скриньки, тестування білої скриньки, інтеграційне тестування та модульне тестування. Валідація завжди приходить після перевірки. Зазвичай ми використовуємо перевірку вимог, щоб переві­рити помилки на початковому етапі розробки, оскільки помилка може при­звести до надмірної переробки, якщо вона буде виявлена пізніше в процесі розробки. Перевірка вимог важлива, оскільки вона допомагає нам переко­натися, що вимоги відповідають ідеальним правилам і стандартам.

Перевірка вимог допомагає перевірити проблеми, пов’язані з вимогами, визначеними під час попередніх варіантів розробки вимог. Зазвичай перевірка використовується для виявлення будь-яких помилок на початкових етапах циклу розробки. Якщо ці помилки не виявити вчасно, вони можуть надмірно збільшити роботу. Перевірка забезпечує точність і ясність даних шляхом усунення будь-яких дефектів у зібраних вимогах. Без перевірки існує високий ризик отримання неточних даних, що призведе до неточних ре­зультатів. Крім того, основною перевагою перевірки вимог є те, що вона зменшує витрати на обслуговування. Міцна база забезпечує надійну струк­туру проекту та знижує ймовірність невдач і відмов.

Люди часто плутають верифікацію та валідацію. Насправді вони не однакові.

Відповідно теорії про управління проектами:

* перевірка: впевненість у тому, що продукт, послуга або система відповідають потребам замовника та інших визначених зацікавлених сторін;
* перевірка: оцінка того, чи відповідає продукт, послуга або система нормам, вимогам, специфікаціям або встановленим умовам.

Простіше кажучи, перевірка вимог – це процес підтвердження того, що системні вимоги містять усі необхідні елементи добре написаних вимог. Перевірка вимог – це процес підтвердження того, що система відповідає своїм цілям і функціям за призначенням. Під верифікацією йдеться мова про перевірку того, чи вимоги повні, правильні та узгоджені. Валідація – це пе­ревірка того, чи відповідає система своїм цілям і функціям за призначенням.

Існують різні методики, які можна використовувати для перевірки вимог. Вони включають:

* ***чеки* –** перевіряючи вимоги, ми перевіряємо документи з вимогами, щоб переконатися, що жодні примітки щодо виявлення не будуть упущені. Під час цих перевірок ми також перевіряємо рівень відстеження всіх вимог. Для цього необхідно створити матрицю простежуваності. Ця матриця га­рантує, що всі вимоги розглядаються серйозно, і все, що зазначено, є ви­правданим. Під час цих перевірок ми також перевіряємо формат вимог. Ми дивимося, чи вимоги чіткі та добре прописані чи ні;
* ***прототипування* –** це спосіб побудови моделі або імітації сис­теми, яку мають створити розробники. Це дуже популярний метод підтвер­дження вимог серед зацікавлених сторін і користувачів, оскільки він допо­магає їм легко визначити проблеми. Ми можемо просто зв’язатися з корис­тувачами та зацікавленими сторонами та отримати їхні відгуки;
* ***дизайн тесту* –** під час розробки тесту ми дотримуємося невеликої процедури, коли спочатку завершуємо команду тестування, а потім ство­рюємо кілька сценаріїв тестування. Функціональні тести можуть бути отри­мані з самої специфікації вимог, де кожна вимога має відповідний тест. На­впаки, нефункціональні вимоги важко перевірити, оскільки кожен тест має бути відстежений до його вимоги. Мета цього полягає в тому, щоб з’ясувати помилки в специфікації або пропущені деталі;
* ***огляд вимог* –** під час перегляду вимог група обізнаних людей структуровано та детально аналізує вимоги та визначає потенційні проблеми. Після цього вони збираються, щоб обговорити проблеми та знайти спосіб вирішення проблем. Складається контрольний список, що складається з різних стандартів, і рецензенти відзначають прапорці, щоб надати офіційну перевірку. Після цього виконується остаточне схвалення.

***Огляд основних принципів тестування програмного забезпечення.***

***Види тестування*** (функціональне, навантажувальне, регресійне тощо).

Функціональне тестування – це вид тестування, в ході якого пере­віряються функціональні можливості програмного продукту шляхом введення різних вхідних значень та перевірки коректності отриманих результатів. Ме­тою функціонального тестування є переконатися, що продукт працює відповідно до очікувань та виконує всі необхідні функції.

Навантажувальне тестування – це вид тестування, в ході якого пере­віряється реакція програмного продукту на велику кількість користувачів або велику кількість запитів. Метою навантажувального тестування є виявлення проблем з продуктивністю та шкалюванням, таких як перевантаження сер­вера чи падіння продуктивності при великому навантаженні.

Регресійне тестування – це вид тестування, в ході якого перевіря­ється, чи не впливають внесені зміни на вже наявну функціональність прог­рамного продукту. Це виконується шляхом запуску попередньо пройдених тестів та перевірки, чи продукт працює коректно після змін. Метою регре­сійного тестування є виявлення потенційних проблем, що можуть з'явитися після внесення змін або впливати на функціональність програмного продукту.

.

**Сьогодні існує дві популярні методології тестування ПЗ:**

1. **Waterfall – стратегія тестування, в якій забезпечення якості являє собою окремий етап розробки, і починається тільки після завершення створення програмного продукту.**
2. **Agile – більш гнучкий метод тестування ПЗ, в якому розробка та тестування продукту виконуються паралельно.**

**Переваги такого підходу в тому, що помилки не накопичуються та швидко виправляються. До його особливостей також варто віднести високий рівень залучення з боку замовників проекту.**

**У міру того, як програмне забезпечення стає складніше, життєвий цикл тестування програмного забезпечення продовжує еволюціонувати. Все частіше розробникам стає невигідно чекати фінальної розробки для початку тестування, оскільки виправлення помилок, у такому разі, може обходитися дорожче за розробку. Тому метод Agile набирає дедалі більшої популярності.**

**Зображення, що містить текст, знімок екрана, Шрифт, схема

Автоматично згенерований опис**

**Рис.7.1. – Методології тестування**

**Перед релізом програмний продукт повинен пройти чотири рівні тестування. Розглянемо кожен рівень докладніше.**

**МОДУЛЬНЕ ТЕСТУВАННЯ**

**Модульне або функціональне тестування програмного забезпечення є першим рівнем QA(забезпечення якості), під час якого перевіряється працездатність окремих програмних модулів, компонентів та функцій. Його мета полягає в тому, щоб упевнитись у коректності роботи кожної одиниці програмного коду.**

**Найчастіше модульне тестування виконується розробниками на етапі кодингу.**

**ІНТЕГРАЦІЙНЕ ТЕСТУВАННЯ**

**В ході інтеграційного тестування окремо перевірені модулі та елементи програмного забезпечення об’єднуються в групи, і перевіряються як цілісні механізми. Головне завдання такого тестування у тому, щоб виявити баги при взаємодії різних модулів.**

**Інтеграційне тестування необхідно з кількох причин:**

* + **найчастіше програмні модулі створюються різними фахівцями, через що їхня логіка може відрізнятися. Тому таке тестування необхідне перевірки взаємодії компонентів.**
  + **у ході розробки програмного продукту вимоги замовника можуть змінюватися, внаслідок чого окремі функції можуть просто не встигнути пройти модульне тестування. Інтеграційне тестування дозволяє виявити у них баги, якщо такі є.**
  + **Обробка винятків може виконуватись із проблемами.**
  + **Зовнішні апаратні інтерфейси можуть бути неправильними.**

**Існує чотири основні типи інтеграційного тестування:**

* 1. **Інтеграція «згори донизу». Головний принцип такого типу тестування полягає в тому, щоб перевіряти працездатність спочатку блоків вищих, а потім нижчих рівнів. Тобто, простими словами, інженер по забезпеченню якості виконує перевірку «зверху вниз».**
  2. **Інтеграція «знизу нагору». Зворотна модель тестування передбачає початок перевірки програмного продукту з низьких рівнів і поступовий перехід до вищих компонентів.**
  3. **Гібридна інтеграція. Комбінована стратегія, у якій використовуються перші два підходи. У цьому випадку модулі верхнього рівня інтегруються та тестуються з блоками нижнього рівня, і низькорівневі модулі навпаки – з верхньорівневими. У такому підході використовуються драйвери та заглушки.**
  4. **Стратегія “Великого вибуху”. І тут всі компоненти продукту збираються разом і тестуються як єдине ціле.**

**СИСТЕМНЕ ТЕСТУВАННЯ**

**Під час системного тестування програмний продукт компілюється як єдине ціле, потім тестується. За допомогою цієї стратегії фахівці по забезпеченню якості перевіряють, у тому числі, функціональність, безпеку та переносимість проєкту.**

**ПРИЙМАЛЬНЕ ТЕСТУВАННЯ**

**Приймальний та системний рівні тестування ПЗ практично не відрізняються за своєю функціональністю, але з істотною поправкою – приймальне тестування виконується групою кінцевих користувачів, які перевіряють програмний продукт з боку споживача та визначають, наскільки він готовий до релізу для широкої публіки.**

**Цей рівень має дві класифікації тестування:**

**Альфа-тестування – виконується групою вибраних користувачів, але разом із командою розробників.**

**Бета-тестування – проводиться користувачами індивідуально, на їх власних пристроях та операційних системах. Очікується, що ця стадія пройде бездоганно, без дефектів і задовольнить потреби аудиторії.**

**Для забезпечення коректної роботи програмного продукту важливо дотримуватися всіх рівнів та методів тестування програм.**

**Тестування – це не просто пошук помилок. Це складний та структурований процес, спрямований на підвищення якості програмного продукту. В основі тестування ПЗ (програмного забезпечення) лежить 7 основних етапів:**

**АНАЛІЗ ВИМОГ**

**Перші кроки забезпечення якості починаються ще з початку розробки програмного продукту — на етапі аналізу вимог замовника. Тестувальникам потрібно:**

* **визначити специфіку та завдання майбутнього продукту;**
* **переконатися у відсутності протиріч у вимогах;**
* **переконатися у повноті та здійсненності всіх завдань;**
* **виявити потенційні дефекти та «вузькі місця» у функціоналі.**

**ПЛАНУВАННЯ ТА РОЗРОБКА СТРАТЕГІЇ**

**Коли вимоги до проєкту сформовані та затверджені, фахівці по забезпеченню якості можуть розпочинати розробку стратегії тестування та планування процедур, спрямованих на покращення якості ПЗ. На цьому етапі визначається бюджет, вирішується, які методи тестування програми будуть використовуватися на кожній стадії її створення.**

**РОЗРОБКА ТЕСТ-КЕЙСІВ**

**На цьому етапі інженери по забезпеченню якості вже мають готовий план тестування і приступають до його безпосередньої реалізації: створюють сценарії для автоматичного тестування, готують тестувальне середовище для проведення безпечних тестів, формують чек-листи та виконують інші підготовчі роботи.**

**ТЕСТУВАННЯ ПЗ**

**Команда інженерів по забезпеченню якості проводить тестування програми, виявляє баги, недоробки, та заповнюють тестувальну документацію, яка згодом буде передана у відділ продакшну на доопрацювання.**

**ПОВТОРНЕ ТЕСТУВАННЯ**

**Коли розробники усувають усі виявлені проблеми, відділ по забезпеченню якості знову береться за роботу та проводить повторне, так зване регресійне тестування. Воно допомагає переконатися, що технічні коригування було внесено правильно, і після всіх доопрацювань продукт почав нормально функціонувати. Це важливий етап, оскільки внесення будь-яких правок може вплинути на роботу програми непередбачуваним чином.**

**ЗАВЕРШЕННЯ ЦИКЛУ ТЕСТУВАННЯ**

**Після завершення всіх етапів тестування фахівцями по забезпеченню якості формують детальний звіт, в якому описують всі тести, проведені протягом життєвого циклу програмного продукту.**

**Питання самоконтролю**

* 1. **Які є види тестування.**
  2. **Яка різниця між верифікацією та валідацією?**
  3. **Назвіть основні етапи тесиування**
  4. **Які є моделі тестування?**
  5. **Які є методики перевірки вимог?**
  6. **Які є популярні методології тестування.**
  7. **Чим відрізняється методологія** Waterfall від методології Agile?
  8. **Які є чотири рівні тестування?**

***Рекомендована література*:** [***9***, c. 5–25; ***11***, c. 10–22; ***13***, c. 7–24]

**Лабораторна 8.**

**Технічне завдання**

|  |
| --- |
|  |

***Мета*:** набути практичних навичок для використання стандартів, що визначають структуру технічного завдання при розробці програмного забезпечення та його створення згідно шаблонів стандартів.

**8.1 Завдання до лабораторної роботи**

1. Розробити технічне завдання згідно вибраного шаблону

2. Оформити роботу.

3. Здійснити захист роботи.

**8.2. Теоретичні відомості і методичні рекомендації**

Технічне завдання - один з основних видів документації при розробці будь-якого проекту. Перш ніж приступити до розробки того чи іншого проекту, спочатку повинен бути складений план. Будівництво, підприємництво, житлові роботи - абсолютно будь-яка трудова сфера вимагає розробки відповідного плану. При цьому абсолютно неважливо, наскільки складною чи серйозною є та чи інша робота. Розробка технічного завдання, а, по суті, звичайного плану дій, є тут ключовим етапом. Технічне завдання потрібно відразу обом сторонам робочого процесу: і виконавцю, і замовнику. Часто між цими двома особами виникають сварки, конфлікти і непорозуміння. Грамотно складений план дій допоможе суворо регламентувати всі зобов'язання кожної сторони.

Найважливіше, що можна відзначити, це той факт, що технічне завдання розробляється тільки замовником. Це свого роду план дій, договір про надання послуг. За допомогою цього документа виконавці можуть чітко визначити свої робочі функції, а також те, що саме від них вимагають. Розглянутий документ завжди повинен розроблятися дуже якісно і уважно. Так, замовник повинен враховувати всі основні тези і пункти, а також не допускати суперечливих моментів. Якщо документ складено грамотно, то замовник завжди зможе вказати незадоволеному виконавцю на певний пункт договору.

Виконавець отримує зразки технічних завдань перед початком виконання тієї чи іншої роботи. Робітник зобов 'язаний дуже уважно ознайомитися з усіма пунктами, що є в документі. Цей крок допоможе уникнути маніпуляцій з боку замовника. Іноді керівники можуть вимагати від працівників чогось такого, про що не йшлося в технічному завданні.

Будь-яке технічне завдання (на поставку, будівництво, транспортування тощо) необхідно дуже грамотно і якісно оформляти. Це потрібно, по-перше, для того, щоб надалі не виникало судових розглядів, спорів і конфліктів через непорозуміння сторін. А по-друге, для простої зручності.

Існує декілька простих правил:

* договір повинен бути детальним і докладним (проте перебільшувати не варто; багатотомні коментарі до вимог навряд чи захоче читати хоч один виконавець);
* договір повинен бути чітким, без води і зайвих відомостей;
* завдання не повинно бути таким собі догматом; варто пам 'ятати, що це лише вказівка, хоч і суворо регламентована.

Стандарт ДСТУ ГОСТ 34.603–92, також відомий як «Технічне за­вдання на розробку програмного забезпечення», є ключовим документом в галузі розробки програмних продуктів в Україні. Він був розроблений та впроваджений з метою нормування вимог до структури та вмісту технічних завдань.

Перед прийняттям стандарту ДСТУ ГОСТ 34.603–92, процес розробки програмного забезпечення в Україні не мав чітких стандартів та загально­визнаних методів підготовки технічних завдань. Це часто викликало невідпо­відність між очікуваннями замовника та результатами розробки.

Процес розробки стандарту ДСТУ ГОСТ 34.603–92 включав у себе роботу експертів з галузі програмної інженерії, представників замовників та виконавців програмних проектів. Вони спільно визначали найважливіші аспекти, які повинні бути враховані в технічних завданнях.

У 1992 році стандарт ДСТУ ГОСТ 34.603–92 був прийнятий та офіційно впроваджений. Це стало важливим кроком у стандартизації процесу розробки програмного забезпечення в Україні.

***Важливі Аспекти Стандарту***

1. Єдність та Уніфікація

Однією з ключових переваг стандарту ДСТУ ГОСТ 34.603–92 є його здатність створювати єдиний мовник для комунікації між замовниками та виконавцями програмних проектів. Це запобігає недорозумінням та сприяє точності передачі вимог.

1. Врахування Вимог Різних Проектів

Стандарт передбачає можливість адаптації до різноманітних проєк­тів, включаючи програмне забезпечення для різних галузей та масштабів.

1. Прозорість та Доступність

Врахування стандарту ДСТУ ГОСТ 34.603–92 спрощує процес підго­товки технічного завдання та робить його більш доступним для всіх учасників проекту.

Стандарт ДСТУ ГОСТ 34.603–92 є ключовим документом для підго­товки технічних завдань в Україні. Він нормує вимоги до змісту, структури та оформлення технічних завдань для різних видів проектів. Даний стандарт за­безпечує єдність у розумінні вимог і умов при розробці програмного продукту.

***Головні положення стандарту***

1. Загальні відомості

Стандарт визначає загальні відомості, такі як назву проекту, інформа­цію про замовника та виконавця, терміни виконання та інші основні атрибути.

***Приклад:***

***Назва проекту*:** розробка вебдодатку для управління складом товарів.

***Замовник*:** ТОВ «Мега-Трейд».

***Виконавець*:** ФОП Іванов І.І.

***Терміни виконання*:** з 01.04.2023 по 30.06.2023.

1. Мета та призначення

У цьому розділі вказуються мета та основні завдання, які перед­бачаються вирішити за допомогою розроблюваного продукту.

***Приклад:***

***Мета*:** розробка системи, що дозволяє ефективно вести облік та керування складськими запасами товарів.

1. Функціональні вимоги

Цей розділ містить перелік функцій, які повинен виконувати про­грамний продукт. Кожна функція описується коротко та конкретно.

***Приклад:***

***Функція 1:*** Додавання нового товару до бази даних.

***Функція 2:*** Редагування інформації про товар.

1. Нефункціональні вимоги

У цьому розділі вказуються вимоги, які не стосуються конкретних функцій продукту, але впливають на його характеристики.

***Приклад:***

***Надійність:*** Система повинна працювати без відмов протягом не менше ніж 99 % часу.

1. Вимоги до інтерфейсу

У цьому розділі описується спосіб взаємодії користувача з програ­мою, вигляд та поведінка користувацького інтерфейсу.

***Приклад:***

***Головне меню*** має містити кнопки для доступу до основних функцій.

***Документування вимог на основі*** ***IEEE Standard 830-1998***

Розглянемо шаблон документа опису вимог, складений К. Вигерсом на основі стандарту. Даний стандарт містить розгорнутий опис вимог, що може бути оптимизировано для потреб конкретної організації.

1. Введення

1.1 Призначення документа.

1.2. Підтримувані угоди.

1.3. Передбачувана аудиторія й рекомендації з послідовності роботи з документом для кожного класу читачів.

1.4. Границі проекту. Тут утримується посилання на документ «Концепція», якщо такий є, або коротке резюме продукту.

1.5. Посилання.

2. Загальний опис.

2.1. Загальний погляд на продукт. Тут необхідно визначити – чи є описуваний продукт новим членом зростаючого сімейства продуктів, новою версією існуючої системи, заміною існуючого додатка або зовсім новим продуктом. Якщо специфікація вимог визначає компонент більшої системи, укажіть, як це ПЗ співвідноситься з усією системою й визначте основні інтерфейси між ними.

2.2. Про особливості продукту. Перераховуються ключові особливості продукту або його головні властивості. Тут доречно помістити контекстну діаграму (у вигляді діаграми варіантів використання, потоків даних або ін. специфікацій).

2.3. Класи й характеристики користувачів. Документується процес пошуку акторов, у якому виявляються всі користувачі системи й здійснюється узагальнення (виділення класів) користувачів. Знайдені класи описуються (наприклад – рівень кваліфікації, доступний функціонал тощо).

2.4. Операційне середовище. Розглядається середовище функціону­вання АІС(автоматизованої інформаційної системи), включаючи апаратні засоби, операційні системи, для розпо­ділених систем – географічне розташування користувачів і серверів, топо­логія мережі.

2.5. Обмеження проектування й реалізації. Розглянемо класифікацію обмежень: певні технології, засоби, мови програмування й бази даних, які варто використати або уникати; обмеження, що накладають операційним середовищем продукту; обов'язкові угоди або стандарти розробки; зворотна сумісність із продуктами, випущеними раніше; обмеження, що накладають бізнесами-правилами; обмеження, пов'язані з устаткуванням, наприклад вимоги до швидкодії, обмеження пам'яті або процесора; угоди, пов'язані з користувальницьким інтерфейсом існуючого продукту, які необхідно дотримувати при його поліпшенні, формати й протоколи обміну даними.

2.6. Документація для користувачів.

2.7. Припущення й залежності

3. Функції системи

3.1 Опис і пріоритети. Приводиться короткий опис функції й вказу­ється її пріоритет (ступінь важливості/черговості реалізації).

3.2. Послідовності «вплив–відгук». Необхідно перелічити послідов­ність впливів, надаваних на систему (дії користувачів, сигнали зовнішніх пристроїв тощо), і відгуки системи, що визначають реакцію конкретної функції.

3.3. Функціональні вимоги. Необхідно дати деталізацію *i*-й функції, перелічити деталізовані функціональні вимоги, включаючи реакцію на очі­кувані помилки й невірні дії. Кожній детальній функціональній вимозі при­власнюється унікальний ідентифікатор.

4. Вимоги до зовнішнього інтерфейсу.

Далі розглянуті конкретні рекомендації з написання розділів цього параграфа:

4.1. Інтерфейси користувача:

* посилання на стандарти графічного інтерфейсу користувачів або стильових рекомендацій для сімейства продукту, які необхідно дотримувати;
* стандарти шрифтів, значків, назв кнопок, зображень, колірних схем, послідовностей полів вкладок, часто використовуваних елементів керування тощо;
* конфігурація екрана або обмеження дозволу;
* стандартні кнопки, функції або посилання переміщення, однакові для всіх екранів, наприклад кнопка довідки;
* швидкі клавіші;
* стандарти відображення повідомлень;
* стандарти конфігурації для спрощення локалізації ПЗ;
* спеціальні можливості для користувачів із проблемами із зором.

4.2. Інтерфейси устаткування.

Опишіть характеристики кожного інтерфейсу між компонентами ПЗ й устаткуванням системи. В опис можуть входити типи підтримуваних пристроїв, взаємодії даних і елементів керувань між ПЗ й устаткуванням, а також протоколи взаємодії, які будуть використовуватися,

4.3. Інтерфейси ПЗ.

Опишіть з'єднання продукту й інших компонентів ПЗ (іденти­фіковані по імені й версії), у т. ч. бази даних, операційні системи, засоби, бібліотеки й інтегровані комерційні компоненти. Укажіть призначення еле­ментів повідомлень, даних і елементів керування, обмін якими відбувається між компонентами ПЗ. Опишіть служби, необхідні зовнішнім компонентам ПЗ, і природу взаємодії між компонентами. Визначите дані, до яких будуть мати доступ компонента ПЗ.

4.4 Інтерфейси передачі інформації.

Укажіть вимоги для будь-яких функцій взаємодії, які будуть викорис­товуватися продуктом, включаючи електронну пошту, Web-браузер, протоколи мережного з'єднання й електронні форми. Визначите відповідні формати по­відомлень. Опишіть особливості безпеки взаємодії або шифрування, частоти передачі даних і механізмів синхронізації.

5. Інші нефункціональні вимоги. В цьому розділі описуються інші нефункціональні вимоги, що не використовуються до вимог до інтерфейсу, які пред­ставлені в розділі 4, і до обмежень, описуваних у розділі 2.5.

5.1 Вимоги до продуктивності.

Укажіть спеціальні вимоги до продуктивності для різних системних операцій. Обґрунтуйте їхню необхідність для того, щоб допомогти розроблю­вачам прийняти правильні рішення, що стосуються дизайну.

Наприклад, через сталі вимоги до часу відгуку бази даних розроб­лювачі можуть продублювати базу даних у декількох географічних метапо­ложениях або денормалізувати зв'язані таблиці баз даних для отримання більш швидкої відповіді на запит.

Додаток А. Словник термінів (глоссарий).

Додаток Б. Моделі аналізу. У цей розділ містяться всі моделі, побу­довані в процесі аналізу вимог.

Додаток В. Список питань

Це динамічний список ще не реалізованих проблем, пов'язаних з ви­могами. Це можуть бути елементи, позначені як «ТВ» (to be determined – необхідно визначити), відкладені рішення, необхідна інформація, недозволені конфлікти тощо.

***Документування вимог в MSF***

На початку фази проектування проектна група працює із проектними вимогами. Вони підрозділяються на:

* бізнес-вимоги;
* вимоги до експлуатації;
* системні вимоги;
* вимоги користувача.

Одним з основних результатів фази проектування є функціональна специфікація, що служить:

* інструкцією команді розроблювачів про те, що вони повинні будуть створити;
* основою для оцінювання об'єму роботи;
* чітка угода із Замовником про те, що повино бути зроблено;
* основою для синхронізації роботи всієї проектної команди.

**Питання самоконтролю**

* 1. Що називається технічним завданням?
  2. З яких елементів складається стандарт ДСТУ ГОСТ 34.603–92?
  3. Для чого використовується технічне завдання?
  4. Назвіть важливі аспекти стандарту MSF.
  5. Назвіть основні розділи IEEE Standard 830-1998.

***Рекомендована література*:** [***10***, c. 6–25; ***12***, c. 10–21; ***13***, c. 7–44]

**ЛІТЕРАТУРА**

1. ДСТУ ГОСТ 34.603-92 "Завдання на розробку програмного забезпечення"

2. ДСТУ ГОСТ 34.602-92 "Завдання на розробку програмних систем"

3. ДСТУ ГОСТ 34.601-92 "Завдання на розробку програмних продуктів"

4. Вигерс К.И. Розробка вимог до ПЗ. – М.: Російська редакція Microsoft, 2004. – 575 c.

5. Леонов И. В. Введення в методологію розробки програмного забезпечення за допомогою Rational Rose // Ескейп, 2004. 301 с.

6. Zave P., Jackson M. Four Dark Corners of Requirements Engineering // ACM Transactions on Software Engineering, January 1997. № 1.

7. Pinheiro Francisco A. C., Goguen Joseph A.. An Object-Oriented tool for Tracing Requirements // Software. Mach 1996. № 3.

8. Guckkenheimer S., Peter J. Software Engineering With Microsoft Visual Studio. Team System. Adison Wesley, 2006. 273 p.

**9.**Sommerville, I. Software Engineering (10th Edition). Addison-Wesley, **2015. 241 p.**

**10.**Pressman, R.S. Software Engineering: A Practitioner's Approach (8th Edition). McGraw-Hill Education, **2014. 136 p.**

**11.**Pfleeger, S.L., Atlee, J.M. Software Engineering: Theory and Practice (4th Edition). Prentice Hall, **2009. 321 р.**

**12.**Gause, D.C., Weinberg, G.M. Exploring Requirements: Quality Before Design. Dorset House Publishing, **1989. 277 р.**

**13.**Wiegers, K. Software Requirements (3rd Edition). Microsoft Press, **2013. 420 р.**

**14.** Beizer, B. Software Testing Techniques (2nd Edition). Van Nostrand Reinhold, **1990. 108 р.**

**15.** Kan, S.H. Metrics and Models in Software Quality Engineering (2nd Edition). Addison-Wesley Professional, **2011. 260 р.**

**16.** Ghezzi, C., Jazayeri, M., Mandrioli, D. Fundamentals of Software Engineering (2nd Edition). Pearson Education, **2010. 333 р.**

**17.** Leffingwell, D., Widrig, D. Managing Software Requirements: A Use Case Approach (2nd Edition). Addison-Wesley Professional, **2003. 111 р.**

**18.**Wiegers, K., Beatty, J. Software Requirements (2nd Edition). Microsoft Press, **2013. 378 р.**

ДОДАТОК А

**Анкета для опитування зацікавлених осіб**

1. Ім'я.

2. Найменування організації.

3. Найменування структурного підрозділу.

4. Посада.

5. Кому ви безпосередньо підкоряєтеся?

6. Які ваші основні обов'язки?

7. Що ви в основному робите?

8. Для кого?

9. Які документи або яку інформацію можна рахувати вхідними або необхідними, для вашої діяльності?

10. Які документи або яку інформацію можна рахувати вихідними або результатом вашої діяльності?

11. Як вимірюється успіх вашої діяльності?

12. Які проблеми впливають на успішність вашої діяльності?

13. Які тенденції, якщо такі існують, роблять вашу роботу простішою або складнішою?

14. Який інтерес або які потреби у вас є відносно майбутнього рі­шення (ПЗ, що розробляється)?

ДОДАТОК Б

**Глосарій**

|  |  |  |
| --- | --- | --- |
| Скорочення | Терміни | Визначення |
| RM | Управління вимогами | Процес ідентифікації, документування, аналізу, відстеження, встановлення пріоритетів, затвер-дження та підтримки вимог до проєкту або продукту. |
| BRD | Документ бізнес-вимог | Офіційний документ, який описує бізнес-вимоги високого рівня для проєкту або продукту. За-зви-чай він містить інформацію про бізнес-по-треби, обсяг, зацікавлених сторін, функціональні вимоги, нефункціональні вимоги, припущення, обмеження, ризики та графік проєкту. |
| FRD | Документ функціональних вимог | Детальний документ, який описує конкретні функ-ціональні вимоги до проєкту або продукту. Заз-вичай він містить інформацію про функції сис-теми, вимоги користувача, варіанти викорис-тання, сценарії, вимоги до даних і критерії прийняття. |
| NFRD | Документ нефункціональних вимог | Детальний документ, який описує конкретні не-функціональні вимоги до проєкту або продукту. Зазвичай він містить інформацію про продук-тивність системи, масштабованість, доступність, надійність, безпеку, придатність до обслугову-вання, зручність використання та доступність. |
| SRS | Специфікація вимог до програмного забезпечення | Комплексний документ, який описує функціо-нальні та нефункціональні вимоги до програм-ної системи. Зазвичай він містить інформацію про ар-хітектуру системи, дизайн, впровадження, тесту-вання та розгортання. |
| Використовуйте Case | Використо-вуйте Case | Техніка для фіксації та опису функціональних ви-мог системи шляхом визначення взаємодії між системою та її користувачами чи іншими систе-мами. Зазвичай він містить опис кроків, зроблених користувачем або системою для досягнення конк-ретної мети чи завдання. |
| Матриця простежуваності | Матриця простежуваності | Документ, який забезпечує відстежуваний зв’язок між вимогами, дизайном, впровадженням, тесту-ванням і розгортанням системи. Зазвичай він міс-тить інформацію про зв’язок між вимогами та ін-шими системними артефактами, такими як тес-тові випадки, дефекти та запити на зміни. |

Продовдження додатка Б

|  |  |  |
| --- | --- | --- |
| Скорочення | Терміни | Визначення |
| Змінити контрольну панель | Змінити контрольну панель | Група зацікавлених сторін, відповідальних за оцінку, схвалення та керування змінами вимог, проєктування, впровадження, тестування та роз-гортання системи. Зазвичай до нього входять пред-ставники різних відділів, наприклад відділу біз-несу, розробки, тестування та операцій. |
| Вимоги Виявлення | Вимоги Виявлення | Процес збору та документування вимог до проєкту чи продукту від зацікавлених сторін, користувачів та інших джерел. Зазвичай це включає такі методи, як інтерв’ю, опитування, спостереження, фокус-групи та мозкові штурми. |
| Зацікавлена сторона | Зацікавлена сторона | Особа або група людей, які зацікавлені в успіху проєкту чи продукту. Зазвичай це клієнти, корис-тувачі, спонсори, власники бізнесу, розробники, тестувальники та допоміжний персонал. |
| Пріоритезація вимог | Пріоритезація вимог | Процес ранжування вимог до проєкту чи продукту в порядку важливості чи терміновості. Зазвичай це включає в себе визначення критичних вимог, які повинні бути розглянуті в першу чергу, і призна-чення рівня пріоритету кожній вимозі на основі її бізнес-цінності, технічної здійсненності та ризику. |
| Інструменти управління вимогами | Інструменти управління вимогами | Програмна програма, яка використовується для підтримки процесу керування вимогами. Зазвичай він включає такі функції, як збір вимог, відсте-ження, контроль версій, співпраця, звітування та аналітика. Приклади інструментів керування вимо-гами включають Visure Solutions, IBM Rational DOORS, Jama Connect і HP ALM. |
| Базова лінія | Базова лінія | Набір затверджених вимог, який є основою для подальшого розвитку та тестування системи. За-звичай він включає функціональні та нефункціо-нальні вимоги, які були погоджені зацікавленими сторонами та підписані радою контролю змін. |
| Перевірка | Перевірка | Процес оцінки того, чи є вимоги до системи пов-ними, точними та відповідними потребам і очіку-ванням зацікавлених сторін. Зазвичай це включає перегляд документів вимог, проведення перевірок зацікавленими сторонами та перевірку того, що система відповідає заданим вимогам за допомогою тестування та інших методів. |
| Перевірка | Перевірка | Процес оцінки відповідності системи заданим ви-могам. Зазвичай це включає перевірку системи на відповідність критеріям прийнятності, визначеним у документах вимог, і забезпечення правильності виконання всіх вимог. |

Додаток В

Приклад створення технічного завдання для сайту

Приклад структури сайту:

1.Головна сторінка

2.Товари (каталог)

2.1. Розділи

2.1.1. Товар

2.2. Корзина

2.3. Замовлення та його оформлення

3.Новини/статті

3.1. Новина/стаття

4.Акції/пропозиції

4.1. Акція/пропозиція

5.Контакти

6.Про нас

7.Доставка та оплата

Додатково у опитуванні замовник має надати інформацію про свою компанію: галузь, продажі та їх специфіка, канали з яких залучаються клієнти. Таким чином виконавець матиме можливість зрозуміти які потреби має бізнес та як створити сайт таким аби він відрізнявся від конкурентів та задовольнив побажання клієнта.

Пункти та приклади технічного завдання на сайт

Технічне завдання зазвичай має декілька розділів, які можуть мати різні назви, але мати однаковий сенс.

Питання щодо організації

1.Дедлайн здачі сайту. Не можна назвати оптимального терміну до визначення особливостей проекту, проте приблизні терміни - можна. Розробка сайту на шаблоні може тривати 2-4 тижні. Якщо ж це лендінг на шаблоні, то термін ще скорочується. Проте, якщо сайт робиться не з шаблону, а “з чистого листка” без наявного дизайну, із складними функціями, то така робота може зайняти щонайменше місяць.

2.Об’єм роботи в деталях. Складність задач визначає об’єм роботи. Лендінг із однієї сторінки та інтернет-магазин (із залученням підтримки сторонніх сервісів) матимуть зовсім різні задачі та відповідно різний об’єм робіт для виконавця.

3.Етапи роботи та детальних опис кожного із них. Сюди входять і програмні рушії і дизайн і основні розділи сайту, вміст, контент, доменне ім’я та особливості усього перерахованого.

Які є вимоги до сайту

1.Використання сайта та сценарій

2.Вимоги до структури

a.до навігації

b. до дизайну

c. до юзабіліті сайту

d. до функціональних можливостей

e. до технологій

Це досить узагальнений список, проте достатній для розуміння що потрібно мати у ТЗ для розробки сайта.

Приклад технічного завдання на розробку сайту на відображення головної сторінки:

1.Вміст Головної сторінки

2.Шапка

2.1. Логотип (при натисканні веде на головну сторінку)

2.2. Номери телефонів

2.3. Пошук

2.4. Корзина (вказано кількість обраних позицій та загальна сума)

2.5. Зворотній зв’язок (кнопка)

3.Меню навігації

3.1. Каталог (включно із функцією випадаючого меню із блоками категорій)

3.2. Акції

3.3. Оплата та доставка

3.4. Блог

3.5. Контакти

Детальніше про головні пункти у ТЗ на розробку сайту

Програмний рушій

В технічному завданні має бути вказано який програмний рушій сайту є в пріоритеті. Варіанти:

* Wix або Tilda - SaaS-платформи (спосіб оренди програмного забезпечення)
* framework (платформа для створення, шаблон) або самописний програмний рушій
* Wordpress, Drupal або OpenCart - CMS (система керування контентом, набір скриптів для створення, редагування і управління ним на сайті)

Для вибору відповідного програмного рушія на допомогу прийдуть такі питання:

1.Адміністрування сайту. Хто буде займатись? Професійний веб-майстер чи замовник бере це на себе?

2.Публікація контенту. Хто буде займатись?

3.Обслуговування сайту. Хто обслуговуватиме? Розробник чи замовник бере це на себе?

Дизайн

Оскільки описати бажаний дизайн у письмовому вигляді досить важко, то в цьому випадку варто мати вже розроблений бренд-бук. За його відсутності потрібно дізнатись вимоги до дизайну:

* Шрифти: головні та другорядні
* Тематика зображень
* Кольори: головні та другорядні
* Поєднання кольорів (які можна, а які суворо недопустимі)
* Візуальний простір та його компонування

Опис форм, фігур, тіней, ліній тощо варто винести у окреме технічне завдання для веб-дизайнерів.

Розділи на сайті

Технічне завдання на розробку обов’язково має містити інформацію про головні розділи та особливості посадкових сторінок для основних розділів.

Приклади типів структури: лінійна та блокова

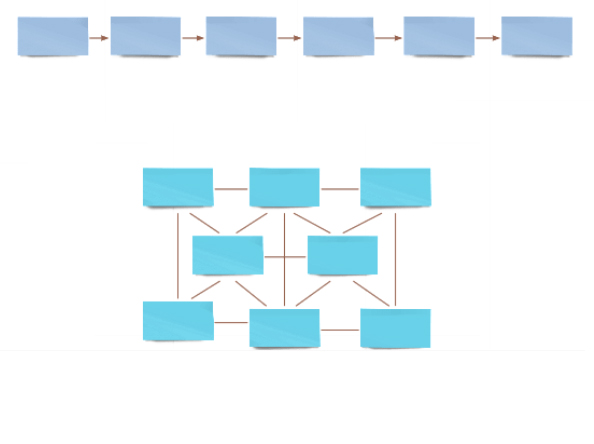


Рис.1.1 – Приклади типів структури.

Навігація зазвичай стандартна, проте якщо є якісь особливі побажання, обов’язково потрібно вказати їх у ТЗ.

Для опису структури сайту у технічному завдання варто використовувати блоки, проте, якщо структура проста, то достатньо просто списків. Плюсом буде базове семантичне ядро для основних сторінок.

Вміст сторінок

Виконавець має зробити макети для головної, посадкових та сторінок для контенту. Структура може мати вигляд списку, проте й графічні прототипи за потреби будуть доцільними.

Опис елементів вмісту сторінок

Елементи сторінок - кнопки, шапка сайта, футер (нижній колонтитул) та сайдбар (бічна панель навігації або додаткової інформації). У ТЗ для елементів сторінок мають бути прописані всі деталі (ідентичність шапки, футера на кожній сторінці, шрифти тощо).

Генерація сторінок

В цьому розділі потрібно вказати які особливості можуть бути формуючи службові сторінки такі як 404.

Фільтри

Ця частина найбільш корисна для комерційних сторінок. Сторінки із фільтрами зазвичай дуже загружені, оскільки із безконтрольно перезагружають, скидають вже вибрані позиції тощо. Консультація та поради розробників допоможе у вирішенні питань сторінок-фільтрів та їх генерації.

Інтернет-магазин

Якщо в ТЗ є інтернет-магазин, то варто звернути увагу на прописання оптимізації фільтрів. Варіант вимог для фільтрів:

* Перший рівень сторінок-фільтрів або ж там, де перетинаються два та більше параметри, потрібно задіяти статистичні посилання (URL повинен бути семантичним). Також такі сторінки повинні мати можливість налаштування тегів таких як Title, заголовки H1-H6, Description та Keywords власноруч.
* Формуючи URL фільтри потрібно продумати порядок, оскільки можна в результаті зіткнутися із помилками та непрацюючими параметрами.
* Також будуть корисними налаштування індексації сторінок-фільтрів та коректна внутрішня перелінковка.

Контент

Якщо не прописати наявність контенту під час формування технічного завдання, можна отримати пустий сайт. Варто прописати цей момент у ТЗ на розробку сайта. Контент може надати як замовник так і виконавець, прописавши вимоги до текстів, фото тощо у технічному завданні.

Технології, які мають підтримуватись

Вимоги щодо технічних особливостей до сайту варто прописати в окремому списку. Для прикладу:

1.Мобільна версія та адаптивний шаблон

2.Верстка під всі браузери

3.Відповідність параметрам щодо:

a. швидкості загрузки основного контенту (до 2,5 сек.)

b. переміщення елементів протягом завантаження сторінки (до 0,1 бала)

c. очікування першої взаємодії із контентом (до 100 мілісек.)

4.Пошукова оптимізація головних сторінок

5.Коректний код відповіді сервера на усіх сторінках

6.Доступ через FTP

7.Семантичний URL

8.Карта сайта

Ім’я та зона доменуЗ

авданням замовника є надання варіантів для ім’я сайта. Весь час додаються сотні нових зон доменів, а отже збільшується можливість обрати бажане ім’я.

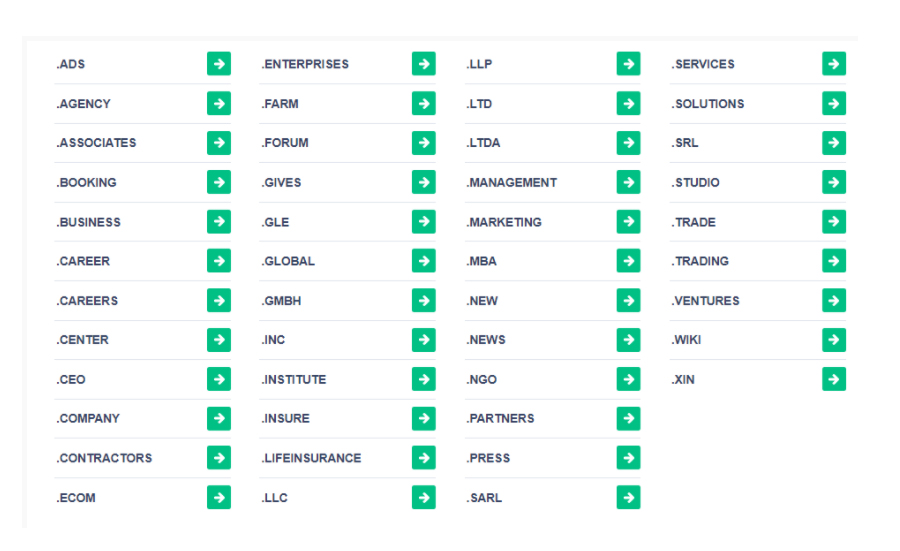


Рис.1.2 – Приклади імен домену

Для обрання вірного імені варто надати чек-ліст із вимогами для доменного імені:

1.Продаючі слова в імені такі як kupi, sale тощо

2.Назва компанії в імені

3.Ключові слова (бажано мати список слів які можна використати в доменному імені)

4.Тематика або категорія доменного імені: реклама, продукти, спорт тощо.

Хостинг-провайдер та вимоги до нього

На скільки буде доступним ваш сайт та на скільки безперебійно працюватиме, залежить від хостингу. Заповнюючи технічне завдання варто вказати наступне:

* тип - віртуальний, VPS або ж сервер
* швидке встановлення CMS
* доступ по SSH, захист від DDoS-атак, backup
* статична нагрузка на акаунт (за планом)
* трафік (за планом: к-сть користувачів/добу)
* файловий менеджер
* безкоштовне перенесення сайту