ЗМІСТ

[Перелік познак та скорочень 4](#_Toc198109216)

[Вступ 5](#_Toc198109217)

[1 Обґрунтування необхідності автоматизації процесу пошуку кулінарних рецептів 6](#_Toc198109218)

[1.1 Аналіз сучасного стану пошуку та підбору кулінарних рецептів 6](#_Toc198109219)

[1.2 Моделювання та вдосконалення бізнес-процесів підбору рецептів на основі наявних продуктів 11](#_Toc198109220)

[1.3. Розробка специфікації вимог до програмного забезпечення 20](#_Toc198109221)

[1.3.1 Загальний опис програмного забезпечення, що створюється 20](#_Toc198109222)

[1.3.2 Розробка системи вимог до прикладного програмного забезпечення 22](#_Toc198109223)

[1.3.3 Обмеження проектування, розробки та впровадження 26](#_Toc198109224)

[1.3.4 Визначення послідовності розробки ПЗ 27](#_Toc198109225)

[2 Проєктування «Recipe Helper» 29](#_Toc198109226)

[2.1 Розробка поведінкових діаграм 29](#_Toc198109227)

[2.1.1 Розробка діаграм варіантів використання 29](#_Toc198109228)

[2.1.2 Уточнення реалізації варіантів використання 32](#_Toc198109229)

[2.2 Розробка бази даних для зберігання інформації 41](#_Toc198109230)

[2.2.1 Визначення бізнес-правил предметної області 41](#_Toc198109231)

[2.2.2 Розробка структури бази даних 42](#_Toc198109232)

[3 Вибір стеку технологій 48](#_Toc198109233)

[3.1 Вибір типу застосунку 48](#_Toc198109234)

[3.2 Вибір формату системи керування базами даних 49](#_Toc198109235)

[3.3 Опис мови програмування та фреймворків 50](#_Toc198109236)

[3.4 Опис засобів розробки програмного забезпечення 52](#_Toc198109237)

[3.5 Визначення компонентів та розробка діаграми розгортання 53](#_Toc198109238)

[4 Розділ 4 54](#_Toc198109239)

[Висновки 55](#_Toc198109240)

[Список джерел інформації 56](#_Toc198109241)

[Додаток А Приклад горизонтального розташування рисунків 62](#_Toc198109242)

[Додаток Б Приклад додатку із структурними розділами, таблицями та рисунками 63](#_Toc198109243)

[Додаток В Приклад змісту пояснювальної записки до курсового проєкту 67](#_Toc198109244)

# Перелік познак та скорочень

БП – бізнес-процес;

ПЗ – програмне забезпечення;

СКБД – система керування базами даних;

AI – artificial intelligence;

API – application programming interface;

CRUD – Create, Read, Update, Delete;

GDPR – General Data Protection Regulation;

HTTPS – Hyper Text Transfer Protocol Secure;

JDBC – Java Database Connectivity;

UML – Unified Modeling Language:

FAQ – frequently asked questions.

# Вступ

Оцифрування повсякденних завдань значно спрощує життя людини, і процес приготування їжі не є винятком. Вибір страви, облік наявних продуктів та формування списку покупок можуть займати багато часу, що особливо актуальною проблемою для людей із динамічним ритмом життя. Сучасні мобільні застосунки та онлайн-сервіси частково допомагають у пошуку рецептів, але часто не враховують реальні запаси продуктів користувача, не пропонують зручні альтернативи відсутнім інгредієнтам і не автоматизують процес складання списку покупок.

Саме тому постає потреба у розробці розумної системи, яка інтегрує всі ці етапи в єдиний зручний механізм. Використовуючи можливості штучного інтелекту та аналізу даних, система навчиться адаптуватися до індивідуальних вподобань користувача. Наприклад, якщо людина надає перевагу здоровому харчуванню або має дієтичні обмеження, алгоритм підбиратиме відповідні рецепти та виключатиме небажані продукти.

Автоматизація цих процесів не лише підвищить комфорт у повсякденному житті, а й допоможе користувачам вести більш раціональне харчування, контролювати свої витрати та зменшити споживання ресурсів. Такий підхід відповідає сучасним трендам цифрової трансформації, коли технології не просто спрощують завдання, а й формують новий рівень взаємодії з повсякденням.

Метою роботи є зменшення витрат часу на планування приготування їжі.

Об’єктом дослідження є підбір кулінарних рецептів.

Предметом дослідження є підходи до пошуку рецептів за списком інгредієнтів, а також принципи зберігання даних про наявні продукти та рецепти.

Задача роботи полягає у створенні програмного забезпечення, яке надасть користувачам можливість швидко знаходити рецепти, раціоналізувати використання продуктів та покращити процес приготування страв.

# 1 Обґрунтування необхідності автоматизації процесу пошуку кулінарних рецептів

## 1.1 Аналіз сучасного стану пошуку та підбору кулінарних рецептів

У сучасному світі цифрові технології значно спрощують багато аспектів повсякденного життя, включаючи планування харчування та приготування їжі. Однак, процес пошуку кулінарних рецептів все ще займає багато часу та залишається безрезультатним для багатьох користувачів.

Для читання роботи можна використати декілька скорочень.

1. БП – бізнес-процес;
2. ПЗ – програмне забезпечення;
3. СКБД – система керування базами даних;
4. AI – artificial intelligence;
5. API – application programming interface;
6. CRUD – Create, Read, Update, Delete;
7. GDPR – General Data Protection Regulation;
8. HTTPS – Hyper Text Transfer Protocol Secure;
9. JDBC – Java Database Connectivity;
10. UML – Unified Modeling Language:
11. FAQ – frequently asked questions.

Значна кількість людей стикається з проблемою підбору страв на основі наявних продуктів. Це стосується як звичайних користувачів, які хочуть швидко приготувати їжу з того, що є під рукою, так і організацій, що займаються харчуванням (ресторани, кафе, кейтеринг). Проблема ускладнюється тим, що ручний пошук рецептів займає багато часу, а також може не враховувати індивідуальні особливості користувача.

Це особливо актуально для людей, які ведуть активний спосіб життя, дотримуються спеціальних дієт або мають обмеження у продуктах через алергію чи інші фактори.

Традиційні методи підбору рецептів, такі як кулінарні книги, перегляд сайтів із рецептами чи консультації з іншими людьми, не завжди є швидким та зручним. Користувачам доводиться витрачати значні зусилля на пошук та адаптацію рецептів під наявні продукти, що часто призводить до марнування ресурсів та необхідності додаткових закупівель.

Більшість кулінарних вебсайтів та мобільних додатків надають користувачам величезну базу рецептів, але часто не дозволяють здійснювати точний пошук за наявними інгредієнтами. Це призводить до ситуації, коли користувач змушений вручну підбирати страви або докуповувати відсутні продукти, що не завжди є зручним чи економічно вигідним рішенням.

Основні проблеми існуючих рішень у предметній області наведені нижче.

1. Відсутність автоматизованого підбору страв за доступними продуктами.

На поточний момент користувачі, які бажають приготувати страву, змушені самостійно підбирати рецепти, аналізуючи список інгредієнтів у кожному з них. Це може бути тривалим і незручним процесом, оскільки:

відсутність автоматизованого механізму підбору рецептівУнаслідок цих факторів користувачам доводиться або довго підбирати рецепт, або імпровізувати, що може вплинути на якість і збалансованість харчування.

1. Неврахування індивідуальних особливостей (рис.2.4).

Кожна людина має власні харчові уподобання, обмеження та дієтичні потреби, але більшість кулінарних сервісів не враховують цього, що створює певні незручності:

* алергії та непереносимість продуктів – користувачі, які мають алергію (наприклад, на горіхи, глютен або молочні продукти), змушені самостійно аналізувати кожен рецепт, щоб переконатися, що він їм підходить;
* вегетаріанство та веганство – рецепти не завжди адаптовані для людей, які не вживають певні категорії продуктів; наприклад, навіть у розділі «вегетаріанські страви» можуть міститися інгредієнти тваринного походження;
* дієтичні обмеження – люди, які дотримуються певних дієт (кетогенна, безвуглеводна, низькокалорійна), змушені вручну відбирати рецепти, що відповідають їхньому режиму харчування;
* особисті вподобання – деякі користувачі просто не люблять певні продукти, наприклад, кінзу або баклажани, і хочуть їх виключити з усіх рецептів.

Через відсутність персоналізації користувачам доводиться переглядати десятки рецептів або адаптувати їх вручну, що ускладнює процес планування харчування.

1. Відсутність гнучкості у фільтрації рецептів за доступними продуктами (п.3).
2. На сьогодні більшість кулінарних сервісів пропонують лише базовий пошук рецептів за назвою або категорією (наприклад, «Сніданок», «Обід», «Вечеря»). Однак вони не враховують конкретні продукти, які є у користувача. Це призводить до низки проблем:

* жорсткі обмеження підбору – навіть якщо користувач вводить кілька продуктів у пошуку, система не враховує можливість приготування страв із частковим збігом інгредієнтів;
* відсутність вибору рівня відповідності – користувач не може вказати, наскільки важливий збіг інгредієнтів; наприклад, він міг би приготувати страву, якщо збігається 80% продуктів, але система не надає такої можливості;
* обмеження для користувачів із нестандартним набором продуктів – люди, які часто експериментують на кухні, не можуть знайти рецепти, які хоча б частково відповідають їхнім запасам;
* відсутність можливості комбінованого пошуку – користувачі не можуть поєднувати пошук за категоріями та продуктами одночасно, нВаприклад, знайти всі сніданки, які можна приготувати з яєць, сиру та помідорів.

Через ці обмеження користувачам доводиться переглядати рецепти вручну, витрачаючи більше часу на підбір потрібної страви.

1. Неможливість швидкої заміни відсутніх інгредієнтів (табл.5).

Коли користувач знаходить рецепт, у якому бракує кількох інгредієнтів, система не пропонує альтернативи, що значно ускладнює процес приготування:

* неавтоматизований процес пошуку альтернатив – користувач змушений шукати інформацію в інтернеті, що займає додатковий час;
* не враховується наявність схожих продуктів – якщо в рецепті є «філе лосося», а у користувача – «філе форелі», система не пропонує заміну, хоча ці продукти є взаємозамінними;
* відсутність рівня пріоритетності замін – іноді заміна інгредієнтів може значно змінити смак або текстуру страви, якщо система не враховує цього, користувач може отримати невдалий результат.

Через ці обмеження приготування їжі стає складнішим, особливо для людей, які не мають достатнього кулінарного досвіду.

1. Неоптимальне використання продуктів (див. п.3.4).

Без зручного обліку наявних продуктів деякі інгредієнти можуть залишатися невикористаними і користувачі часто не встигають використати їх до закінчення терміну придатності, що призводить до фінансових втрат та збільшення харчових відходів.

1. Відсутність інтеграції з обліком наявних продуктів у користувача.

Більшість кулінарних сервісів не мають функції обліку продуктів, які є вдома, тому користувачі змушені вести такі списки вручну або просто покладатися на пам’ять. Це створює кілька проблем:

* відсутність автоматичного оновлення запасів – користувач не може бачити, які продукти він уже використав і що залишилося; наприклад, якщо він приготував страву, система не зменшує кількість використаних інгредієнтів;
* відсутність продуктів через забуття – без обліку користувач може не помітити, що певний продукт закінчується або псується, що призводить до харчових відходів і зайвих витрат;
* складність у плануванні покупок – якщо система не враховує, що є в холодильнику, користувач змушений окремо перевіряти свої запаси перед покупкою нових продуктів;
* немає можливості швидкого доповнення списку – користувач не може додавати продукти до списку покупок безпосередньо з розділу обліку запасів.

Як наслідок, процес планування харчування стає менш зручним, а користувачі можуть витрачати більше грошей на непотрібні покупки.

1. Відсутність зручного способу формування списку відсутніх інгредієнтів для подальшої покупки.

Користувачі, які знайшли рецепт і бажають його приготувати, часто стикаються з проблемою, що деякі інгредієнти відсутні. У такому разі їм доводиться вручну складати список покупок, що спричиняє низку незручностей:

1. висока ймовірність помилок – через людський фактор користувач може забути додати якийсь інгредієнт у список, що призведе до повторного походу в магазин або потреби змінювати рецепт;
2. дублювання вже наявних продуктів – якщо користувач не перевіряє свої запаси перед покупкою, він може придбати те, що вже є вдома, що призводить до зайвих витрат;
3. немає інтеграції з супермаркетами – більшість сучасних рецепт-сервісів не дозволяють одразу ж замовити відсутні інгредієнти в онлайн-магазині, що могло б значно спростити процес.

Через ці недоліки процес покупки інгредієнтів для приготування їжі стає менш зручним, що може призводити до того, що користувачі уникають готування складних страв і віддають перевагу напівфабрикатам або замовленню їжі.

Автоматизована система підбору рецептів дозволить вирішити цю проблему шляхом швидкого аналізу наявних продуктів, пошуку відповідних рецептів та рекомендації можливих замінників для відсутніх інгредієнтів.

## 1.2 Моделювання та вдосконалення бізнес-процесів підбору рецептів на основі наявних продуктів

На даний момент процес підбору рецептів з врахуванням продуктів користувача здійснюється вручну або за допомогою сторонніх онлайн-ресурсів, які не забезпечують достатньої гнучкості та автоматизації. Основна проблема полягає у відсутності єдиного інтегрованого рішення, що дозволяло б користувачам швидко знаходити рецепти, перевіряти наявність необхідних інгредієнтів та формувати списки покупок.

У поточному вигляді ("AS-IS") цей процес складається з кількох етапів, які користувач проходить самостійно. Відсутність автоматизації змушує людину витрачати більше часу та зусиль на підбір відповідного рецепта, перевірку наявних інгредієнтів та складання списку необхідних покупок. Ці етапи будуть розглянуті детальніше нижче.

1. Пошук рецептів.

Пошук страв, які можна приготувати, здійснюється кількома способами. Деякі користувачі звертаються до кулінарних книг або записаних раніше рецептів. Інші шукають рецепти в інтернеті за допомогою пошукових систем або спеціалізованих сайтів. У процесі пошуку необхідно враховувати багато факторів, зокрема доступність інгредієнтів, складність приготування та індивідуальні уподобання щодо їжі.

Якщо користувач шукає рецепти онлайн, він може ввести назву бажаної страви або перелік продуктів, які є в наявності. Проте більшість платформ не дозволяють гнучко відфільтровувати рецепти за доступними інгредієнтами. Це означає, що навіть після введення переліку продуктів користувачеві доводиться вручну переглядати рецепти та визначати, наскільки вони підходять.

Якщо ж рецепти шукають в кулінарних книгах або записниках, користувачеві необхідно самостійно згадати, які страви можна приготувати з доступних продуктів. Це вимагає більше часу та зусиль, особливо якщо людина не має чітко сформованої системи ведення записів.

1. Перевірка наявності інгредієнтів.

Після того як користувач знаходить рецепт, він має перевірити, чи всі необхідні інгредієнти є вдома. Цей процес відбувається вручну: людина переглядає полиці в холодильнику, кухонні шафи, щоб зрозуміти, які продукти є в наявності.

У цьому етапі можуть виникати кілька труднощів, які наведені нижче.

По-перше, відсутність системи обліку. Оскільки немає автоматизованого обліку продуктів, користувачеві доводиться покладатися на власну пам’ять або вести записи вручну. Це може спричинити помилки: деякі продукти можуть бути в наявності, але забуті, а інші, навпаки, вважатися наявними, хоча насправді їх уже немає.

По-друге, потреба в додаткових перевірках. Навіть після візуального огляду запасів користувач може не бути впевненим, чи вистачить певного інгредієнта для приготування страви.

Якщо необхідного продукту немає, користувачеві доводиться приймати рішення:

* знайти інший рецепт;
* спробувати замінити відсутній інгредієнт чимось схожим;
* записати продукт у список покупок і придбати його пізніше.

1. Прийняття рішення про приготування страви.

Якщо всі необхідні продукти є в наявності, користувач може розпочати приготування страви. Якщо ж деяких інгредієнтів бракує, доводиться шукати альтернативні варіанти або відкладати приготування до моменту покупки.

Важливим моментом є те, що при виборі рецепта користувач не завжди враховує терміни придатності наявних продуктів. Це може призводити до ситуацій, коли певні продукти залишаються невикористаними та псуються.

1. Формування списку покупок.

Якщо інгредієнтів не вистачає, користувач має записати їх у список покупок. Зазвичай це робиться вручну: людина записує необхідні продукти в блокнот, на аркуші паперу або в нотатки на смартфоні.

Цей процес також має кілька недоліків:

1. відсутність централізованого обліку: оскільки списки покупок ведуться вручну, у них можуть потрапляти повторювані або зайві продукти;
2. немає інтеграції з рецептом: користувачу доводиться самостійно визначати, які саме продукти варто додати до списку;
3. необхідність оновлення вручну: після покупки людина має викреслити придбані продукти зі списку, що може спричинити плутанину.
4. Закупівля продуктів.

Під час відвідування магазину користувач використовує свій список покупок, щоб придбати необхідні інгредієнти. Якщо деякі продукти не в наявності або дорогі, доводиться приймати рішення про їх заміну або виключення.

Труднощі цього етапу:

* якщо список покупок був складений неправильно або деякі позиції були забуті, користувачу доведеться повертатися в магазин пізніше;
* відсутність синхронізації списку з реальними запасами може призвести до того, що людина купить вже наявні продукти, не знаючи, що вони є вдома;
* витрати часу на пошук необхідних продуктів у магазині.

1. Приготування страви.

Після покупки всіх необхідних інгредієнтів користувач повертається додому та розпочинає приготування страви. На цьому етапі він знову звертається до рецепта та виконує інструкції.

Процес приготування може ускладнюватися кількома факторами:

* якщо рецепт був знайдений онлайн, потрібно мати доступ до інтернету або попередньо зберегти його;
* відсутність покрокових підказок і таймерів може призводити до помилок під час приготування;
* необхідність постійного перемикання між рецептом і кухонними процесами.

Таким чином, бізнес-процес (БП) у моделі "AS-IS" є ряд проблем:

1. відсутність автоматизованого механізму підбору рецептів за доступними продуктами змушує користувача вручну здійснювати пошук, що займає багато часу;
2. немає єдиної системи обліку продуктів, через що виникають помилки у визначенні наявності інгредієнтів;
3. процес формування списку покупок здійснюється вручну та не синхронізується із залишками продуктів;
4. користувач не має змоги легко замінити відсутні інгредієнти на альтернативні;
5. відсутність персоналізованих рекомендацій не дозволяє враховувати індивідуальні вподобання, алергени та дієтичні обмеження.

Ці проблеми ускладнюють процес приготування їжі, що потребує впровадження автоматизованого рішення. Побудована модель БП у варіанті "AS-IS", тобто модель бізнес-процесу приготування страви у поточному стані, зображена на рис. 1.1.

Зображення, що містить текст, схема, План, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Рисунок 1.1 – Модель "AS-IS" бізнес-процесу приготування страви

Вдосконалена модель бізнес-процесу приготування страви "TO-BE" передбачає впровадження автоматизованої системи, яка значно оптимізує всі етапи підбору рецептів, обліку продуктів та формування списків покупок. Основна ідея – створити єдину інтегровану платформу, яка дозволить користувачам швидко знаходити рецепти, автоматично визначати доступність необхідних інгредієнтів, пропонувати альтернативи у разі їхньої відсутності та формувати список покупок без ручного введення даних.

Завдяки автоматизації процесів користувачеві більше не потрібно переглядати кулінарні книги, шукати рецепти на різних сайтах, перевіряти вміст холодильника вручну та складати списки покупок окремо. Система зробить усе це автоматично, виходячи з поточних запасів продуктів та індивідуальних вподобань користувача.

1. Автоматизований підбір рецептів за наявними продуктами.

Однією з основних проблем поточної моделі є те, що користувач самостійно шукає рецепти, витрачаючи на це багато часу. У вдосконаленій моделі система виконує цей процес автоматично.

Користувач вносить у програму список продуктів, які є в наявності. Це можна зробити вручну або за допомогою сканування штрих-кодів чи імпорту даних з інтегрованих сервісів (наприклад, хмарного списку покупок). Після цього система аналізує доступні інгредієнти та пропонує список можливих страв, які можна приготувати.

Алгоритм підбору враховує не лише точну відповідність продуктів, а й можливі варіанти заміни. Наприклад, якщо у рецепті вказане вершкове масло, а в користувача є тільки рослинна олія, система може запропонувати цей варіант як альтернативу.

Додатково, користувач може налаштувати фільтри, наприклад:

* час приготування (від швидких рецептів до складніших страв);
* кухня світу (італійська, азійська, українська тощо);
* дієтичні обмеження (вегетаріанські, безглютенові, низьковуглеводні).

1. Інтеграція з обліком наявних продуктів.

У поточній моделі користувач змушений вручну перевіряти наявність інгредієнтів. В оновленому бізнес-процесі ця функція автоматизована.

Система веде електронний облік продуктів, що є у користувача, включаючи їхню кількість і термін придатності. Дані можуть оновлюватися наступними способами:

* автоматично при додаванні нового продукту через сканування штрих-коду або введення вручну;
* при списанні продуктів після приготування страви (система автоматично зменшує кількість використаних інгредієнтів у базі);
* шляхом інтеграції з онлайн-магазинами або супермаркетами (за умови доступу до чеків покупок).

Якщо якийсь продукт наближається до закінчення терміну придатності, система повідомляє про це користувача та пропонує відповідні рецепти для його використання, що дозволяє мінімізувати харчові відходи.

1. Автоматизоване формування списку покупок.

Раніше користувач самостійно складав список необхідних покупок, переглядаючи рецепти та обираючи інгредієнти вручну. У вдосконаленій моделі цей процес повністю автоматизований.

Якщо обраний рецепт містить продукти, яких немає у наявності, система автоматично додає їх до списку покупок. При цьому враховується:

* чи є у користувача вже частина необхідної кількості продукту (наприклад, якщо потрібно 500 грам борошна, а є 300 грам, то до списку додається лише 200 грамів);
* чи можна замінити відсутній продукт іншим, що є в наявності;
* чи потрібен цей продукт для інших страв у найближчі дні (щоб уникнути повторних покупок).

Список покупок формується в електронному вигляді та може бути синхронізований зі смартфоном користувача. Також можливе інтегрування з онлайн-магазинами, що дозволить здійснювати замовлення продуктів безпосередньо через додаток.

1. Розширені можливості персоналізації.

Система враховує індивідуальні особливості користувача, такі як:

* дієтичні обмеження (наприклад, безлактозні або безглютенові продукти);
* алергії на певні інгредієнти (система автоматично виключає такі продукти з підбору рецептів);
* особисті вподобання щодо страв (користувач може оцінювати рецепти, і алгоритм підлаштовуватиме рекомендації відповідно до смаків).

Також система аналізує харчові звички та пропонує нові страви на основі попереднього вибору, що дозволяє урізноманітнити раціон.

1. Покращена взаємодія при приготуванні страви.

У поточному процесі користувач змушений постійно переключатися між рецептом та приготуванням їжі. В оновленій моделі бізнес-процесу пропонується покроковий режим приготування з голосовими командами та вбудованим таймером.

Можливі додаткові покращення:

* відеоінструкції та зображення процесу приготування;
* автоматичне регулювання розрахунку кількості інгредієнтів залежно від кількості порцій;
* голосове керування для зручності використання під час готування.

Основні переваги вдосконаленого бізнес-процесу:

1. економія часу – користувач не витрачає годин на пошук рецептів та складання списків покупок;
2. зменшення харчових відходів – система пропонує страви на основі наявних продуктів, не допускаючи псування їжі;
3. зручність – автоматизований облік продуктів, інтеграція зі списком покупок та покрокові інструкції значно спрощують процес приготування їжі;
4. гнучкість – можливість персоналізувати систему під власні вподобання та дієтичні обмеження;
5. оптимізація покупок – уникнення непотрібних витрат завдяки автоматичному підрахунку необхідної кількості продуктів.

У результаті впровадження цієї системи користувач отримує максимально комфортний та швидкий процес приготування їжі, який значно покращує його якість життя. Побудована модель БП приготування страви у варіанті "TO-BE" зображена на рис. 1.2.

Зображення, що містить текст, схема, План, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Рисунок 1.2 – Модель "TO-BE" бізнес-процесу приготування страви

Виконане моделювання дозволило виявити ключові учасники процесу, їхні дії, а також логіку взаємодії з системою під час підбору рецептів відповідно до наявних продуктів. У результаті було не лише формалізовано поточні бізнес-процеси, але й запропоновано оптимізації для підвищення зручності користувача та покращення роботи системи. Це моделювання стало основою для подальшої формалізації функціональності майбутнього програмного продукту, що реалізується через визначення вимог до нього. У наступному розділі буде представлено загальну специфікацію програмного забезпечення, що створюється.

## 1.3. Розробка специфікації вимог до програмного забезпечення

### 1.3.1 Загальний опис програмного забезпечення, що створюється

#### 1.3.1.1 Опис продукту, що розроблюється

У рамках даної курсової роботи розробляється вебзастосунок під назвою «Recipe Helper». Це програмне забезпечення (ПЗ), орієнтоване на спрощення щоденного приготування їжі шляхом автоматизації вибору страв, виходячи з наявних у користувача продуктів. Основна мета проекту — надати користувачеві інструмент, який дозволить зручно керувати власними запасами їжі, зменшити харчові відходи, заощадити час на планування меню та зробити процес приготування більш приємним і організованим.

Програмне забезпечення є інтерактивним вебзастосунком створеним з використанням сучасного стеку технологій, таких як Spring Boot для серверної частини та JavaScript — для клієнтського інтерфейсу. Система дозволяє користувачам додавати до свого сховища продуктів список наявних інгредієнтів, після чого автоматично формує підбірку відповідних рецептів. У разі, якщо деяких продуктів не вистачає, система пропонує можливі заміни або формує список покупок.

Застосунок працюватиме в середовищі сучасних браузерів (Google Chrome, Mozilla Firefox) на десктопних і мобільних пристроях. Серверна частина буде розгортатися на Windows-сервері з підтримкою Java 21+, а також використовуватиме реляційну базу даних PostgreSQL.

Розроблюване програмне забезпечення безпосередньо пов’язане з поточними потребами цифровізації побуту. Воно слугуватиме інструментом оптимізації використання ресурсів, допоможе у зниженні харчових відходів, поліпшить щоденне планування та підвищить обізнаність користувачів у сфері здорового харчування. Проект відповідає сучасним тенденціям сталого розвитку та раціонального споживання, що робить його актуальним не лише на побутовому, а й на соціальному рівні.

#### 1.3.1.2 Перспектива програмного продукту

«Recipe Helper» є автономним програмним продуктом, що розробляється як незалежна система, яка в подальшому може стати частиною більшої екосистеми побутових сервісів. Наприклад, у перспективі можливе розширення функціоналу: інтеграція з обліковими системами кухонних ваг, смарт-холодильників, сервісів замовлення продуктів або додатків для контролю калорійності раціону.

На момент розробки вебзастосунок не є частиною існуючого сімейства продуктів, але структура програмного забезпечення побудована таким чином, щоб у майбутньому була можлива масштабована інтеграція з іншими компонентами. В архітектурному плані реалізація має модульну структуру з application programming interface (API)-інтерфейсами для взаємодії з зовнішніми сервісами (наприклад, служби доставки, хмарне зберігання даних, рекомендовані системи харчування).

#### 1.3.1.3 Класи та характеристики користувачів

Передбачено кілька основних типів користувачів, що мають різні сценарії взаємодії з програмним забезпеченням.

Найбільш чисельною групою є домашні користувачі, які прагнуть швидко планувати приготування їжі. Це можуть бути студенти, молоді сім’ї, зайняті офісні працівники або літні люди. Усі вони мають різний технічний рівень, тому інтерфейс має бути інтуїтивно зрозумілим, з простим керуванням та адаптивним дизайном.

Технічно підковані користувачі можуть виявити інтерес до більш гнучких функцій — кастомізації рецептів, налаштування дієт, додавання власних рецептів або підключення сторонніх сервісів через API.

У перспективі можуть з’явитися й бізнес-користувачі, наприклад, малі кав’ярні або заклади харчування, які використовують застосунок для формування меню згідно з наявними залишками інгредієнтів.

Усі ці категорії користувачів потребують різного рівня підтримки та документації. До складу кінцевого програмного продукту буде включено наступні компоненти документації:

* інтерактивна довідка (help center);
* frequently asked questions (FAQ)-розділ на вебсайті.

Ці елементи мають бути доступними українською мовою. Формат подачі передбачає як класичний текстовий, так і візуальний супровід (інфографіка, скріншоти), що зробить використання системи комфортним для користувачів будь-якого рівня підготовки.

Таким чином, «Recipe Helper» — це не просто застосунок для пошуку страв, а інтелектуальний побутовий помічник, орієнтований на автоматизацію кухонної рутини з урахуванням індивідуальних потреб і сучасних цифрових стандартів.

### 1.3.2 Розробка системи вимог до прикладного програмного забезпечення

#### 1.3.2.1 Функціональні вимоги до розроблюваного програмного забезпечення

Розроблюване програмне забезпечення «Recipe Helper» виконуватиме низку основних функцій, спрямованих на полегшення процесу планування приготування їжі, виходячи з наявних у користувача продуктів. Програма має забезпечити повний цикл взаємодії користувача з системою — від створення облікового запису до отримання рекомендацій щодо рецептів, а також управління власним сховищем продуктів. Усі функції мають бути організовані логічно та відповідати стандартним очікуванням сучасних користувачів, при цьому враховувати гнучкість, масштабованість і простоту у використанні.

Першочерговою функціональною вимогою є реєстрація та автентифікація користувачів, що дозволяє створювати особисті облікові записи, зберігати індивідуальні налаштування. Система повинна підтримувати функції реєстрації нових користувачів через електронну пошту.

Після входу в систему користувач має отримати доступ до функціоналу сховища продуктів, де він може додавати продукти. Продукти можуть бути обрані зі списку доступних, або створені вручну користувачем. До кожного продукту додається кількість. Користувач може редагувати або видаляти продукти у будь-який момент.

На основі вмісту сховища продуктів система повинна здійснювати автоматичний підбір рецептів, які відповідають наявним інгредієнтам. У випадку, коли якогось продукту не вистачає, програма або пропонує альтернативні рецепти, або надає можливість згенерувати список покупок. Алгоритм підбору має враховувати кількість інгредієнтів, приблизний час приготування, складність рецепту та, за бажанням користувача, інші критерії — наприклад, тип кухні світу, тип страви, дієтичні вимоги. Усі рецепти зберігаються в централізованій базі даних, яка модерується адміністрацією на відповідність базовим правилам спільноти.

Інтегрованою є функція фільтрації та пошуку рецептів, яка дає змогу знаходити страви за різними параметрами: за назвою, категорією, часом приготування, наявністю всіх чи більшості інгредієнтів. Пошукова система повинна враховувати часткові збіги слів.

Для системного адміністратора передбачено окремий інтерфейс для керування базою продуктів і рецептів, модерації контенту, аналізу помилок і забезпечення стабільної роботи сервера.

Також система має підтримувати локалізацію, що дозволить відображати інтерфейс різними мовами. На початковому етапі реалізується одномовна версія українською.

#### 1.3.2.2 Нефункціональні вимоги до розроблюваного програмного забезпечення

1. Вимоги до продуктивності.

Основне навантаження передбачається на модуль підбору рецептів, взаємодії з сховищем продуктів, а також під час одночасного доступу багатьох користувачів до системи.

У стандартному режимі програмне забезпечення має забезпечувати що 95% запитів на підбір рецептів обробляються до 3 секунд. Додавання чи видалення продукту з сховища продуктів має здійснюватися за менш ніж за 1 секунду.

Серверна частина повинна бути здатна обробляти не менше 1000 запитів на годину, включаючи авторизацію, зміну списку продуктів та генерацію списку покупок. У межах кожного запиту система повинна мати змогу обробляти до 10 одиниць інгредієнтів (об’єкти типу «продукт» з кількісними характеристиками) без помітного погіршення продуктивності.

Також передбачається підтримка 1 одночасно відкритої сесії адміністратора на сервері управління, із середнім часом відповіді інтерфейсу — не більше ніж 3 секунди під час модерації контенту.

1. Атрибути якості програмного забезпечення.

Якісні характеристики програмного забезпечення відіграють ключову роль для забезпечення високого рівня задоволення кінцевих користувачів. З погляду зовнішньої якості, першочергове значення мають доступність, зручність використання, надійність та цілісність даних.

Програмне забезпечення має бути легко інсталювати та оновлювати — процес встановлення вебверсії повинен обмежуватися стандартною авторизацією в обліковому записі.

Система має зберігати цілісність даних при збоях або непередбачуваних ситуаціях, таких як раптове відключення живлення або втрата інтернет-з'єднання. Усі ключові операції (зміни сховища, додавання рецептів, оновлення профілю) зберігаються транзакційно, з підтримкою відкату в разі помилки.

Програмне забезпечення повинне забезпечувати стабільну взаємодію з базами даних PostgreSQL.

Зручність використання оцінюється за показником часу навчання користувача: середній час освоєння базового функціоналу (створення профілю, додавання продуктів, пошук рецептів) не повинен перевищувати 10 хвилин. Інтерфейс повинен бути інтуїтивно зрозумілим, підтримувати адаптивну верстку для різних розмірів екранів та мати чітку навігацію.

З внутрішніх атрибутів якості головними є модифікованість, масштабованість і можливість повторного використання компонентів. Архітектура програмного забезпечення проектується таким чином, щоб зміни в окремих модулях (наприклад, алгоритмі підбору рецептів чи форматі відображення картки страви) не вимагали перегляду всієї системи.

Компоненти мають бути реалізовані з урахуванням принципів повторного використання, зокрема логіка пошуку та фільтрації, модулі взаємодії з базою даних і компонент формування списку покупок.

Усі основні модулі повинні бути легко тестованими за допомогою автоматизованих unit-тестів, з покриттям не менше 70% основного коду. Це дозволяє підтримувати можливість перевірки на кожному етапі змін.

1. Вимоги до безпеки та розмеження доступу.

Безпека даних та контроль доступу користувачів до системи є критично важливими елементами, що гарантують довіру користувачів і відповідність нормативним вимогам щодо обробки персональної інформації. Система повинна бути захищена від несанкціонованого доступу, втрати або пошкодження даних, а також забезпечувати цілісність та конфіденційність оброблюваної інформації.

Усі користувачі повинні проходити автентифікацію за допомогою логіна та пароля, або через зовнішні сервіси. Паролі зберігаються в базі даних у зашифрованому вигляді з використанням сучасних хеш-функцій як bcrypt.

Передбачено розмежування ролей у системі: звичайні користувачі можуть додавати та редагувати лише власні дані, у той час як адміністратори мають доступ до всіх об’єктів, модераційного інтерфейсу. Усі критично важливі дії (видача прав, видалення даних, зміна ролей) логуються та зберігаються щонайменше 40 днів для можливого аудиту.

Комунікація між клієнтською та серверною частинами захищена протоколом Hyper Text Transfer Protocol Secure (HTTPS). Дані користувачів (email, пароль, історія дій) не передаються третім сторонам та не використовуються в рекламних цілях. Система повинна відповідати вимогам регламенту General Data Protection Regulation (GDPR) у разі зберігання або обробки даних користувачів з території Європейського Союзу.

У разі підозрілої активності (часті невдалі входи, зміна IP-адреси, спроби доступу до захищених ресурсів) система повинна тимчасово блокувати доступ та надсилати відповідне повідомлення адміністратору. Крім того, при спробі видалення облікового запису передбачено обов’язкове підтвердження особи, щоб запобігти випадковій або шкідливій втраті даних.

### 1.3.3 Обмеження проектування, розробки та впровадження

У межах проекту існує низка обмежень, які прямо або опосередковано впливають на процес прийняття технічних рішень. Деякі з них є об’єктивними з погляду апаратного забезпечення або політики організації, інші — результатом необхідності інтеграції з зовнішніми сервісами чи дотримання галузевих стандартів.

Зокрема, у процесі розробки враховується корпоративна політика щодо використання відкритого коду та ліцензування компонентів — дозволяється використання лише бібліотек з ліцензіями типу MIT або Apache 2.0. Система повинна бути незалежною від платформ або сервісів, які у майбутньому можуть обмежити гнучкість підтримки або масштабування. Також існує вимога до підтримки GDPR, що накладає обмеження на способи зберігання персональних даних.

З технічної точки зору, система орієнтована на роботу на хмарному сервері з обмеженням ресурсів: не більше 2 CPU, 4 ГБ оперативної пам’яті та 20 ГБ простору на диску. Це зумовлює вибір легковагових фреймворків, кешування даних та оптимізації обчислень.

Серед інтерфейсних обмежень важливо зазначити необхідність взаємодії з існуючим користувацьким обліковим записом, який може бути спільним для інших сервісів компанії, що передбачає єдину систему авторизації.

Щодо технологічних обмежень визначено стек інструментів: Java з використанням Spring Boot на серверній частині, PostgreSQL як основна база даних, JavaScript для побудови інтерфейсу, GitLab як система контролю версій. Забороняється використання бета-версій бібліотек та будь-яких інструментів, які не мають документації англійською мовою.

Крім того, є обмеження щодо мовної підтримки — система повинна бути багатомовною (мінімум українська), що зумовлює необхідність реалізації механізмів інтернаціоналізації на ранніх етапах проєктування.

Додатково варто згадати припущення, які були закладені у план проєкту: передбачається стабільна робота серверної інфраструктури, своєчасне надання вхідних даних замовником (наприклад, список базових продуктів або правила складання рецептів), а також наявність технічного супроводу після впровадження системи. Якщо хоча б одне з цих припущень зміниться, це може вплинути на терміни або бюджет реалізації.

### 1.3.4 Визначення послідовності розробки ПЗ

Розробка програмного забезпечення «Recipe Helper» відбувається у кілька логічних етапів, кожен з яких передбачає чіткий розподіл обов’язків між членами команди, послідовність виконання робіт, а також створення відповідних артефактів, які будуть передані замовнику після завершення кожного етапу. Важливою частиною процесу є також врахування низки обмежень і припущень, які впливають на вибір архітектури, технологій, термінів та можливостей реалізації.

До складу команди розробки входять фахівці з різним профілем, які забезпечують повний цикл створення програмного продукту — від аналізу вимог до впровадження готової системи. Основу команди складає аналітик (який також виконує роль менеджера проєкту), один програміст (відповідальний за фронтенд та бекенд), фахівець із тестування, а також консультант із UX/UI-дизайну. При потребі залучаються зовнішні спеціалісти з безпеки або технічної підтримки.

Приблизна тривалість розробки становить 3 місяці. Цей період включає всі ключові фази: від формалізації вимог й проектування архітектури, до тестування, документування та впровадження. У перший місяць проводиться активна фаза планування, аналізу предметної області, створення дизайн-макетів інтерфейсу та розробки технічного завдання. У другий і третій місяць реалізується основна частина бізнес-логіки, забезпечується зв’язок між клієнтською та серверною частинами, ведеться первинне тестування модулів. Останній місяць передбачає фінальне тестування, усунення дефектів, підготовку супровідної документації, розгортання системи у продуктивному середовищі та демонстрацію замовнику.

На кожному етапі формуються певні артефакти, які є підтвердженням виконаної роботи. До них належать: технічне завдання, прототипи інтерфейсу, документація з архітектури системи, звіт з результатами тестування (оцінки покриття тестами), а також доступ до репозиторію з готовим кодом.

# 2 Проєктування «Recipe Helper»

## 2.1 Розробка поведінкових діаграм

### 2.1.1 Розробка діаграм варіантів використання

У системі передбачено декілька категорій користувачів, зокрема гість, зареєстрований користувач (Cooker) та адміністратор. Категорія Cooker є ключовою, оскільки саме вона охоплює більшість функціональних можливостей платформи. Користувачі цієї ролі мають бути здатними взаємодіяти з інтерфейсом програми, мати базові цифрові навички, зокрема — вміти користуватися вебзастосунками, вводити текст, працювати з фільтрами пошуку, зберігати результати на пристрої. Обов’язковою умовою є наявність зареєстрованого облікового запису та успішна авторизація в системі.

Для користувачів не встановлюється особливих професійних чи вікових вимог, оскільки система орієнтована на широке коло осіб — від початківців у кулінарії до досвідчених користувачів, які прагнуть оптимізувати процес приготування їжі.

Use Case діаграми є одним із базових засобів моделювання в Unified Modeling Language (UML) і призначені для графічного представлення функціональних вимог до системи з погляду взаємодії зовнішніх учасників (акторів) з системою. Ці діаграми дозволяють отримати цілісне уявлення про поведінку системи, не заглиблюючись у внутрішню реалізацію. Ключовою перевагою є можливість візуалізувати, які саме дії можуть виконувати різні типи користувачів.

На діаграмах обов’язково використовуються наступні ключові елементи.

1. Актори (Actors) – зовнішні користувачі системи або інші системи, які взаємодіють з програмним забезпеченням (ПЗ). Для даного ПЗ, актор "Cooker" є зареєстрованим користувачем.
2. Варіанти використання (Use Cases) – функції або сервіси, які система надає користувачу.
3. Відношення (Relationships):
4. association – зв’язок між актором і варіантом використання, що означає можливість взаємодії;
5. include – показує обов’язкове включення одного Use Case до іншого (наприклад, перевірка наявності інгредієнтів є обов’язковим етапом пошуку за наявними продуктами);
6. extend – відображає необов’язкове, умовне розширення (наприклад, збереження рецепта на пристрій або використання заміни інгредієнтів).

Побудова Use Case діаграми ґрунтувалася на специфікації вимог та реальному сценарії користування програмою. Застосовано ієрархічну організацію варіантів використання: від загального сценарію (наприклад, перегляд списку рецептів) до детальних підпроцесів (пошук за інгредієнтами, збереження результатів, підтвердження використання). Така структура дозволяє уникнути дублювання та зробити модель гнучкішою для підтримки.

Побудована Use Case діаграма для зареєстрованого користувача "Cooker" зображена на рис. 2.1.

Під час розробки діаграми варіантів використання для користувача Cooker було враховано наступні моменти.

1. Виділення окремих сценаріїв взаємодії із системою – усі дії кулінара були згруповані відповідно до логічних функціональних блоків: робота з рецептами, списками покупок, інвентарем продуктів, повідомлення про проблему.
2. Повторне використання поведінки через include – наприклад, перевірка наявності інгредієнтів виконується у кількох місцях, тому була винесена в окремий Use Case і включена в ті сценарії, які її потребують.

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

Вміст, створений ШІ, може бути неправильним.

Рисунок 2.1 – Use Case діаграма для зареєстрованого користувача "Cooker"

1. Уточнення через розширення extends – допомогло відобразити залежні або необов’язкові варіанти використання. Наприклад, перегляд деталей рецепту може включати перевірку наявності інгредієнтів, збереження рецепта на пристрій.
2. Мінімізація надмірної деталізації – діаграма мала бути достатньо загальною, тому деталі інтерфейсу, запитів до бази даних або внутрішньої логіки не моделювались у цій діаграмі.
3. Врахування реальних дій користувача – усі сценарії відповідають діям, які кулінар дійсно може виконати в застосунку: пошук рецептів, створення списку покупок, ведення інвентарю продуктів.

Формування діаграм варіантів використання дозволяє отримати загальне уявлення про функціональні можливості програмного забезпечення та взаємодію користувачів з системою. Проте для забезпечення більш глибокого розуміння логіки поведінки системи в рамках кожного варіанта використання, а також для підготовки до етапу реалізації, необхідно здійснити уточнення цих сценаріїв. Таке уточнення досягається за допомогою текстових специфікацій, які детально описують послідовність дій, альтернативні потоки, умови виконання та інші аспекти кожного з варіантів використання. Розглянемо ці уточнення у наступному розділі.

### 2.1.2 Уточнення реалізації варіантів використання

Уточнення варіантів використання (Use Cases) у межах цього програмного забезпечення виконано шляхом складання текстових специфікацій Use Case. Для кожного з ключових сценаріїв взаємодії кулінара із системою (наприклад, "Пошук рецептів", "Підтвердження використання рецепта", "Перегляд інвентарю" тощо) були складені докладні текстові описи, що структуровано відображають поведінку системи у відповідь на дії кулінара.

Такий підхід дав змогу достатньо повно описати логіку виконання кожного варіанта використання без побудови додаткових поведінкових діаграм. Специфікації Use Case містять такі важливі елементи:

* актор, який ініціює сценарій;
* короткий опис сценарію;
* передумови, за яких сценарій може бути виконаний;
* основний потік подій (Main Flow, Basic Flow), що показує послідовність дій користувача та реакцій системи;
* альтернативні потоки (Alternative Flows), які описують виняткові ситуації або гілкування логіки.

Цей спосіб уточнення виявився найбільш доцільним, з огляду на наступне.

1. Невисоку складність функціоналу на поточному етапі розробки. Більшість сценаріїв використання є лінійними або включають одне-два простих розгалуження. Таку логіку легко і чітко передати за допомогою структурованого тексту.
2. Необхідність забезпечити зрозумілу документацію для інших членів команди, зокрема розробників і тестувальників. Специфікації Use Case є достатньо зрозумілими, не вимагають спеціальної підготовки для сприйняття, як у випадку з UML-діаграмами, і можуть використовуватись безпосередньо під час реалізації функціоналу або написання тестів.
3. Однорідність і системність опису. Усі специфікації оформлені за єдиною структурою, що дозволяє легко співставити їх із елементами діаграм Use Case та перевірити відповідність вимогам.

Таким чином, вибір на користь специфікацій Use Case як єдиного способу уточнення обґрунтований простотою сценаріїв, потребою у стислому і зрозумілому документуванні, а також доцільністю витрат часу на поточному етапі розробки. У подальшому, за потреби деталізації більш складних сценаріїв, може бути розглянута можливість створення поведінкових діаграм (наприклад, Activity або Sequence Diagram), однак на поточному етапі це не є необхідним.

Нижче наведено специфікації для всіх Use Cases. Для подібних операцій (наприклад, Create, Read, Update, Delete (CRUD)) буде використано узагальнений шаблон, щоб уникнути дублювання.

1. Специфікація варіанту використання: переглянути загальний список рецептів.

Ім'я: переглянути загальний список рецептів.

Актор: кулінар, система.

Короткий опис: кулінар переглядає список доступних рецептів, наданих системою.

Передумови:

* кулінар увійшов у систему;
* база даних рецептів заповнена.

Потоки подій:

* основний потік:

1. кулінар обирає опцію перегляду рецептів;
2. система отримує рецепти з таблиці бази даних Recipes;
3. система відображає список рецептів із базовою інформацією (назва, кухня, час приготування);
4. кулінар може взаємодіяти з розширеннями (пошук рецептів, перегляд деталей рецепта);

* альтернативний потік 3а: якщо рецепти відсутні, система відображає повідомлення: "Рецепти не знайдено".

1. Специфікація варіанту використання: пошук рецептів.

Ім’я: пошук рецептів.

Актор: кулінар, система.

Короткий опис: кулінар шукає рецепти за певними критеріями (назва, інгредієнти, інвентар).

Передумови: кулінар переглядає загальний список рецептів.

Потоки подій:

* основний потік:

1. кулінар обирає опцію пошуку;
2. система пропонує варіанти пошуку (за назвою, інгредієнтами, інвентарем);
3. кулінар обирає тип пошуку та вводить дані;
4. система виконує запит до таблиці Recipes і повертає відфільтровані результати;
5. система відображає відфільтрований список рецептів;

* альтернативний потік 5а: якщо результати відсутні, система відображає: "Рецепти за вашими критеріями не знайдено".

1. Специфікація варіанту використання: перевірка доступності інгредієнтів.

Ім’я: перевірка доступності інгредієнтів.

Актор: система.

Короткий опис: система перевіряє, чи є необхідні для рецепта інгредієнти в інвентарі кулінара.

Передумови: кулінар взаємодіє з дією, пов’язаною з рецептом.

Потоки подій:

* основний потік:

1. система отримує інгредієнти рецепта з таблиці recipe\_ingredients;
2. система порівнює інгредієнти з таблицею users\_products;
3. система повертає результат (усі доступні, деякі відсутні, або жоден недоступний);

* альтернативний потік 3а: якщо жоден рецепт не відповідає повністю, система пропонує рецепти із замінниками.

1. Специфікація варіанту використання: перегляд деталей рецепта.

Ім’я: перегляд деталей рецепта.

Актор: кулінар, система.

Короткий опис: кулінар переглядає детальну інформацію про обраний рецепт.

Передумови: кулінар переглядає загальний список рецептів.

Основний потік подій:

1. кулінар обирає рецепт зі списку;
2. система отримує детальні дані з таблиці Recipes (інгредієнти, інструкції, час тощо);
3. система виконує перевірку доступності інгредієнтів;
4. система відображає деталі рецепта та статус доступності.
5. Специфікація варіанту використання: збереження рецепта на пристрій.

Ім’я: збереження рецепта на пристрій.

Актор: кулінар, система.

Короткий опис: кулінар зберігає рецепт на своєму пристрої для офлайн-доступу.

Передумови: кулінар переглядає деталі рецепта.

Основний потік подій:

1. кулінар обирає опцію збереження рецепта;
2. система завантажує дані рецепта з таблиці Recipes на пристрій;
3. система підтверджує успішне збереження.
4. Специфікація варіанту використання: підтвердження використання рецепта.

Ім’я: підтвердження використання рецепта.

Актор: кулінар, система.

Короткий опис: кулінар підтверджує, що готуватиме обраний рецепт.

Передумови: кулінар переглядає деталі рецепта.

Потоки подій:

* основний потік:

1. кулінар підтверджує рецепт для приготування;
2. система виконує перевірку доступності інгредієнтів;
3. система оновлює інвентар;
4. система активує режим приготування з покроковими інструкціями;

* альтернативний потік 2а: якщо відсутні інгредієнти, система пропонує замінники або додавання до списку покупок.

1. Специфікація варіанту використання: додавання до списку покупок.

Ім’я: додавання до списку покупок.

Актор: кулінар, система.

Короткий опис: кулінар додає відсутні інгредієнти до списку покупок під час підтвердження рецепта.

Передумови: кулінар підтвердив рецепт, виявлено відсутні інгредієнти.

Основний потік подій:

1. система визначає відсутні інгредієнти;
2. кулінар підтверджує додавання їх до списку покупок;
3. система додає інгредієнти до таблиці ShoppingList.
4. Специфікація варіанту використання: підтвердження замінника інгредієнта.

Ім’я: підтвердження замінника інгредієнта.

Актор: кулінар, система.

Короткий опис: кулінар підтверджує використання запропонованого замінника інгредієнта.

Передумови: кулінар підтвердив рецепт, деякі інгредієнти відсутні, але є замінники.

Потоки подій:

* основний потік:

1. система пропонує замінники з таблиці product\_substitutes;
2. кулінар обирає замінник;
3. система оновлює інструкції рецепта із замінником;
4. система виконує перевірку доступності інгредієнтів;

* альтернативні потоки:

1. 1а, якщо замінники відсутні, система пропонує додати до списку покупок;
2. 2а, якщо кулінар відмовляється від замінника, система пропонує додати до списку покупок.
3. Специфікація варіанту використання: оновлення інвентарю.

Ім’я: оновлення інвентарю.

Актор: система.

Короткий опис: система оновлює інвентар продуктів після приготування або ручних змін.

Передумови: кулінар виконує дію, пов’язану з інвентарем.

Основний потік подій:

1. система визначає використані або оновлені вручну інгредієнти;
2. система оновлює кількість у таблиці Products;
3. система підтверджує оновлення.
4. Специфікація варіанту використання: перегляд списку покупок.

Ім’я: перегляд списку покупок.

Актор: кулінар, система.

Короткий опис: кулінар переглядає поточний список покупок.

Передумови: кулінар увійшов у систему.

Потоки подій:

* основний потік:

1. кулінар обирає опцію перегляду списку покупок;
2. система отримує дані з таблиці ShoppingList;
3. система відображає список елементів із кількістю;

* альтернативний потік 3а: якщо список порожній, система відображає: "Список покупок порожній".

1. Специфікація варіанту використання: збереження списку на пристрій.

Ім’я: збереження списку на пристрій.

Актор: кулінар, система.

Короткий опис: кулінар зберігає список покупок на своєму пристрої для офлайн-доступу.

Передумови: кулінар переглядає список покупок.

Основний потік подій:

1. кулінар обирає опцію збереження списку покупок;
2. система завантажує дані списку покупок з таблиці ShoppingList на пристрій;
3. система підтверджує успішне збереження.
4. Специфікація варіанту використання: CRUD операції зі списком покупок.

Ім’я: збереження списку на пристрій, додавання до списку, оновлення списку, видалення зі списку.

Актор: кулінар, система.

Короткий опис: кулінар виконує операції створення, читання, оновлення та видалення елементів списку покупок.

Передумови: кулінар переглядає список покупок.

Основний потік подій:

1. кулінар обирає операцію (додати, оновити, видалити);
2. для додавання: кулінар додає новий елемент, система оновлює ShoppingList;
3. для оновлення: кулінар змінює кількість або деталі елемента, система оновлює ShoppingList;
4. для видалення: кулінар видаляє елемент, система оновлює ShoppingList;
5. система підтверджує операцію.
6. Специфікація варіанту використання: перегляд інвентарю продуктів.

Ім’я: перегляд інвентарю продуктів.

Актор: кулінар, система.

Короткий опис: кулінар переглядає поточний інвентар продуктів.

Передумови: кулінар увійшов у систему.

Потоки подій:

* основний потік:

1. кулінар обирає опцію перегляду інвентарю;
2. система отримує дані з таблиці Products;
3. система відображає список продуктів із кількістю та термінами придатності;

* альтернативний потік 3а: якщо інвентар порожній, система відображає: "Інвентар порожній".

1. Специфікація варіанту використання: CRUD операції з інвентарем.

Ім’я: додавання до інвентарю, оновлення інвентарю, видалення з інвентарю.

Актор: кулінар, система.

Короткий опис: кулінар виконує операції створення, оновлення та видалення елементів інвентарю продуктів.

Передумови: кулінар переглядає інвентар продуктів.

Основний потік подій:

1. кулінар обирає операцію (додати, оновити, видалити);
2. для додавання: кулінар додає новий продукт, система оновлює Users\_products;
3. для оновлення: кулінар змінює кількість або деталі продукту, система оновлює Users\_products;
4. для видалення: кулінар видаляє продукт, система оновлює Users\_products;
5. система підтверджує операцію.
6. Специфікація варіанту використання: повідомлення про проблему.

Ім’я: повідомлення про проблему.

Актор: кулінар, система.

Короткий опис: кулінар повідомляє про проблему з рецептом.

Передумови: кулінар увійшов у систему.

Потоки подій:

* основний потік:

1. кулінар обирає опцію повідомлення про проблему;
2. кулінар вводить опис проблеми;
3. система зберігає відгук у таблиці Feedback;
4. система підтверджує відправлення;

* альтернативний потік 3а: якщо введення порожнє, система просить ввести опис.

Уточнення варіантів використання шляхом складання специфікацій дозволило деталізувати очікувану поведінку системи у відповідь на дії користувача та описати альтернативні сценарії. Це створює чітке бачення логіки роботи програмного забезпечення, що є важливою передумовою для організації внутрішньої структури зберігання даних. Відтак наступним етапом є розробка структури бази даних, яка дозволить реалізувати функціональні можливості, описані у варіантах використання.

## 2.2 Розробка бази даних для зберігання інформації

### 2.2.1 Визначення бізнес-правил предметної області

У процесі створення програмного забезпечення, яке дозволяє користувачам керувати рецептами, власним продуктовим інвентарем та списками покупок, важливим завданням є проектування надійної бази даних. Метою цього етапу є не лише створення схеми таблиць, але і визначення правил зберігання, цілісності та обробки інформації, необхідної для функціонування системи. База даних буде реалізована у середовищі PostgreSQL, з урахуванням технологічного стеку Spring Boot + Hibernate.

Аналізуючи процес вибору рецептів на основі продуктів в наявності були виявлені особливості, які було виокремлено в наступні основні бізнес-правила, що безпосередньо вплинули на логічну модель даних.

1. Користувачі повинні мати можливість реєструватися, входити до системи, а також мати певну роль (звичайний користувач або адміністратор). Роль визначає права доступу в системі.
2. Кожен рецепт складається з заголовку, опису, інструкцій приготування, часу приготування та списку інгредієнтів.
3. Інгредієнт рецепту — це зв’язок між продуктом і рецептом, який містить кількість потрібного продукту.
4. Кожен продукт належить до певної категорії (наприклад, овочі, молочні продукти тощо) та має одиницю виміру.
5. Користувач має власний інвентар продуктів, який відображає наявну кількість продуктів у нього.
6. Користувач може формувати список покупок, додаючи туди продукти, які він хоче придбати.
7. Категорії продуктів та замінники визначаються централізовано, й користувачі не можуть їх змінювати.
8. Цілісність даних забезпечується шляхом встановлення зовнішніх ключів, обмежень на значення (наприклад, кількість продукту не може бути від’ємною) та унікальності там, де це логічно обґрунтовано (наприклад, ім’я користувача та email).

Ці правила були закладені в основу при побудові логічної моделі даних, яка буде трансформована в фізичну структуру бази даних з використанням SQL-скриптів.

### 2.2.2 Розробка структури бази даних

Структура бази даних побудована після чіткого формулювання функціональних вимог до програмної системи, з урахуванням передбачуваних сценаріїв взаємодії користувача з інтерфейсом. Це дозволило виділити основні сутності предметної області, між якими були встановлені логічні зв’язки, що згодом трансформувалися у таблиці фізичної моделі даних. Таке проектування є стандартною практикою розробки, коли структура бази даних безпосередньо відображає бізнес-правила та поведінку системи.

Загалом, проектована структура забезпечує наступні можливості:

* реєстрацію та автентифікацію користувачів;
* зберігання категорій і продуктів із зазначенням одиниць виміру;
* збереження рецептів із деталізацією інгредієнтів і кроків приготування;
* ведення користувацького інвентарю продуктів;
* формування списку покупок з можливістю редагування.

Нижче наведено опис отриманих результатів проектування та створення бази даних, що включає не лише структуру таблиць, а й семантику кожної колонки, опис домену значень, обмежень і принципів введення даних. Це забезпечує повне розуміння з боку команди розробників і дозволяє довгострокове використання бази даних. На рис.2.2 наведено структуру таблиць бази даних RecipeHelper в системі керування базами даних (СКБД) PostgreSQL.

Таблиця «users» призначена для зберігання облікових даних користувачів. Кожний запис таблиці складається з колонок, опис яких наведений в табл. 2.1.

Таблиця «categories» зберігає список категорій, до яких належать продукти. Категорії є незалежними об’єктами й можуть бути змінені лише адміністративно. Таблиця складається з колонок, опис яких наведений в табл. 2.2.

Таблиця 2.1 – Опис структури таблиці «users»

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Ім’я колонки | Тип даних колонки | Семантика та опис особливостей вводу |
| PK | id | int(11) | Унікальний ідентифікатор користувача. Генерується автоматично. |
|  | username | varchar(50) | Унікальне ім’я користувача. Обов’язкове для введення, використовується для входу в систему. |
|  | password | varchar(100) | Захешований пароль користувача. Обов’язкове поле. |
|  | role | varchar(20) | Роль користувача. Можливі значення: 'USER', 'ADMIN'. |
|  | email | varchar(100) | Унікальна email-адреса. Обов’язкове поле, використовується для реєстрації. |

Зображення, що містить текст, знімок екрана, число, Шрифт

Вміст, створений ШІ, може бути неправильним.

Рисунок 2.2 – Структура таблиць бази даних RecipeHelper

Таблиця 2.2 – Опис структури таблиці «categories»

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Ім’я колонки | Тип даних колонки | Семантика та опис особливостей вводу |
| PK | id | int(11) | Унікальний ідентифікатор категорії. |
|  | name | varchar(50) | Назва категорії. Має бути унікальною. |

Таблиця «products» зберігає список продуктів з назвою, одиницею виміру та посиланням на категорію. Таблиця складається з колонок, опис яких наведений в табл. 2.3.

Таблиця 2.3 – Опис структури таблиці «products»

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Ім’я колонки | Тип даних колонки | Семантика та опис особливостей вводу |
| PK | id | int(11) | Унікальний ідентифікатор продукту. |
|  | name | varchar(50) | Назва продукту. Обов’язкове поле. |
| FK | category\_id | int(11) | Зовнішній ключ до таблиці categories. При видаленні категорії значення стає NULL. |
|  | measure\_type | varchar(10) | Тип одиниці виміру: 'g' (грам), 'ml' (мілілітр), 'pcs' (штуки). Обов’язкове поле. |

Таблиця «recipes» зберігає загальні відомості про рецепти. Кожний запис таблиці складається з колонок, опис яких наведений в табл. 2.4.

Таблиця 2.4 – Опис структури таблиці «recipes»

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Ім’я колонки | Тип даних колонки | Семантика та опис особливостей вводу |
| PK | id | int(11) | Унікальний ідентифікатор рецепту. |
|  | title | varchar(100) | Назва рецепту. Обов’язкове поле. |
|  | description | text | Опис рецепту (короткий зміст). Необов’язкове. |
|  | instructions | text | Покрокова інструкція приготування. Необов’язкове. |
|  | time\_to\_prepare | int(11) | Орієнтовний час приготування в хвилинах. Повинен бути невід’ємним числом. |

Таблиця «recipe\_ingredients» реалізує зв’язок "багато до багатьох" між рецептами та продуктами, вказуючи необхідну кількість продукту для кожного рецепту. Таблиця складається з колонок, опис яких наведений в табл. 2.4.

Таблиця 2.4 – Опис структури таблиці «recipe\_ingredients»

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Ім’я колонки | Тип даних колонки | Семантика та опис особливостей вводу |
| PK | recipe\_id | int(11) | Зовнішній ключ до таблиці recipes. При видаленні рецепту інгредієнти видаляються. |
| PK | product\_id | int(11) | Зовнішній ключ до таблиці products. |
|  | quantity | double | Кількість продукту у відповідному рецепті. Значення повинно бути більшим за нуль. |

Таблиця «users\_products» зберігає інформацію про наявні у користувача продукти та їх кількість (інвентар). Таблиця складається з колонок, опис яких наведений в табл. 2.5.

Таблиця «shopping\_list» зберігає список продуктів, які користувач планує придбати, разом із потрібною кількістю. Таблиця складається з колонок, опис яких наведений в табл. 2.6.

Таблиця 2.5 – Опис структури таблиці «users\_products»

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Ім’я колонки | Тип даних колонки | Семантика та опис особливостей вводу |
| PK | user\_id | int(11) | Зовнішній ключ до users. При видаленні користувача записи видаляються. |
| PK | product\_id | int(11) | Зовнішній ключ до таблиці products. |
|  | amount | double | Кількість продукту, якою володіє користувач. Значення повинно бути більшим за нуль. |

Таблиця 2.6 – Опис структури таблиці «shopping\_list»

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Ім’я колонки | Тип даних колонки | Семантика та опис особливостей вводу |
| PK | user\_id | int(11) | Зовнішній ключ до users. При видаленні користувача записи видаляються. |
| PK | product\_id | int(11) | Зовнішній ключ до таблиці products. |
|  | amount | double | Кількість продукту, яку треба купити. Значення повинно бути більшим за нуль. |

Таблиця «product\_substitutes» призначена для зберігання інформації про допустимі замінники продуктів у рецептах. Це дозволяє реалізувати функціональність вибору альтернативних інгредієнтів у разі відсутності основного продукту в інвентарі користувача. Таблиця містить записи про те, який продукт (product\_id) може бути замінений на який (substitute\_id), а також задає пріоритет такої заміни. Таблиця складається з колонок, опис яких наведений в табл. 2.7.

Таблиця 2.7 – Опис структури таблиці «product\_substitutes»

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Ім’я колонки | Тип даних колонки | Семантика та опис особливостей вводу |
| PK | product\_id | int(11) | Зовнішній ключ до таблиці products. Основний продукт, який може бути замінений. При видаленні продукту запис видаляється. |
| PK | substitute\_id | int(11) | Зовнішній ключ до таблиці products. Продукт-замінник. При видаленні запис видаляється. |
|  | priority | int(11) | Пріоритет заміни. Чим менше значення, тим вища перевага використання даного замінника. За замовчуванням 1. |

Ця структура таблиць дозволяє повністю реалізувати функціональність системи: зберігання профілів користувачів, категорій і продуктів, створення та перегляд рецептів, ведення інвентаря та списку покупок. Було обрано PostgreSQL як СКБД, враховуючи її сумісність із Java (через Java Database Connectivity (JDBC) та Hibernate), підтримку транзакцій, гнучкість типів даних та обмежень.

# 3 Вибір стеку технологій

## 3.1 Вибір типу застосунку

У межах розробки програмного забезпечення (ПЗ) для підбору рецептів за наявними продуктами було обрано вебзастосунок як тип реалізації системи. Обґрунтування цього вибору спирається на функціональні та нефункціональні вимоги, визначені на етапі аналізу, а також на переваги вебтехнологій у контексті сучасного користувацького досвіду та технічного обслуговування.

Аналіз можливих варіантів наведено в табл. 3.1.

Таблиця 3.1 – Аналіз можливих варіантів типів застосунку

|  |  |  |  |
| --- | --- | --- | --- |
| Тип застосунку | Опис | Переваги | Недоліки |
| Вебзастосунок | Додаток працює в браузері, клієнт не потребує встановлення ПЗ | Легке оновлення, доступ з будь-якого пристрою з інтернетом, централізоване зберігання | Залежність від інтернету, обмеження щодо офлайн-функціональності |
| Мобільний застосунок | Встановлюється на смартфон/планшет, має доступ до апаратних функцій пристрою | Зручність користувачу, офлайн-режим, push-сповіщення | Потрібна окрема реалізація для Android та iOS, складне оновлення, вища вартість |
| Десктопний застосунок | Встановлюється на комп’ютер, має повний доступ до ресурсів ОС | Висока продуктивність, не залежить від інтернету | Необхідність встановлення, важке оновлення, обмеження у кросплатформеності |

Критерії вибору:

* відсутність вимог до специфічних функцій пристрою (GPS, камера);
* потреба у доступності на різних пристроях (ПК, смартфон, планшет);
* простота оновлення функціоналу;
* централізоване управління даними та доступами;
* обмежені ресурси команди розробників.

На основі проведеного порівняння було обрано вебзастосунок як найоптимальніший тип для реалізації системи. Він забезпечує легкий доступ для користувачів, зручну підтримку та масштабованість рішення. Усі основні функції системи (пошук рецептів, управління інгредієнтами, формування списку покупок) можуть бути реалізовані у браузері без втрати якості користувацького досвіду.

## 3.2 Вибір формату системи керування базами даних

Для швидкого збереження, обробки та доступу до структурованих даних у програмному забезпеченні критично важливим є обґрунтований вибір системи керування базами даних (СКБД).

У табл. 3.2 будуть розглянуті три найпоширеніші СКБД, які можуть бути використані для зберігання даних у подібних системах.

Таблиця 3.2 – Порівняння систем управління базами даних

|  |  |  |  |
| --- | --- | --- | --- |
| СКБД | Тип СКБД | Переваги | Недоліки |
| MySQL | Реляційна | Висока продуктивність, широке використання, велика спільнота | Менш гнучка підтримка складних транзакцій, GPL-ліцензія |
| PostgreSQL | Реляційна | Повна підтримка SQL-стандартів, розширюваність, підтримка JSON | Складніша в налаштуванні та адмініструванні |
| MongoDB | Документо-орієнтована | Гнучкість структури даних, висока швидкість запису | Не реляційна, складніше забезпечити цілісність зав’язків |

Критерії вибору:

* наявність чіткої структури даних та зав’язків між ними (рецепти, продукти, категорії, користувачі);
* підтримка складних запитів (JOIN, агрегації);
* висока надійність та перевіреність на практиці;
* документованість та відсутність обмежень ліцензії на використання в навчальних проектах.

PostgreSQL було обрано як основну СКБД, з огляду на:

* підтримку транзакцій, зовнішніх ключів, процедур і тригерів;
* розширюваність за рахунок можливості створення власних типів, індексів, функцій;
* наявність повноцінної англомовної документації;
* відкриту ліцензію (PostgreSQL License).

PostgreSQL найкраще відповідає вимогам предметної області проекту. Вона забезпечує високу продуктивність, стабільність, підтримку складних типів даних та транзакцій. Розвинена англомовна документація та сумісність із сучасними фреймворками робить PostgreSQL придатною для розробки та подальшого масштабування вебзастосунку.

## 3.3 Опис мови програмування та фреймворків

У рамках розробки серверної частини програмного забезпечення для системи підбору рецептів було обрано мову програмування Java у поєднанні з фреймворком Spring Boot.

Java – це об’єктно-орієнтована, кросплатформна мова програмування загального призначення, яка вже понад два десятиліття використовується для розробки веб, десктопних та мобільних застосунків. Однією з ключових особливостей Java є наявність віртуальної машини Java (JVM), яка забезпечує платформонезалежність та захищене виконання коду.

Основні переваги Java для розробки серверної частини:

* підтримка багатопотоковості, що дозволяє обробляти велику кількість одночасних запитів від користувачів;
* статична типізація і система компіляції, що зменшує кількість помилок на ранньому етапі;
* розвинена екосистема бібліотек і фреймворків, які покривають більшість задач сучасної веб-розробки;
* широке використання в комерційних проектах, що забезпечує наявність великої кількості навчальних ресурсів та спільноти.

Spring Boot – це високорівневий фреймворк для Java, який спрощує створення мікросервісів та REST API шляхом автоматичної конфігурації, вбудованого вебсервера (Tomcat) та інтеграції з базами даних, системами безпеки, шаблонізаторами тощо.

Технічні особливості Spring Boot:

* автоконфігурація: автоматично підбирає параметри конфігурації на основі залежностей, що підключені до проекту;
* вбудований сервер: не потребує зовнішнього сервера для запуску (Tomcat);
* підтримка REST-контролерів: дозволяє швидко реалізувати application programming interface (API)-запити;
* підтримка безпеки (Spring Security): дозволяє гнучко реалізовувати автентифікацію та авторизацію;
* легка інтеграція з базами даних через Spring Data JPA, що спрощує роботу з SQL-запитами.

У реалізації даного веб-застосунку Java використовується як основна мова для серверної логіки, а Spring Boot – як фреймворк, що забезпечує архітектурну основу сервісів, безпеку, доступ до даних, обробку запитів від клієнта та створення REST API.

Мова програмування Java у поєднанні з Spring Boot є перевіреним та надійним технологічним стеком для створення сучасних веб-застосунків. Завдяки широкому інструментарію, стабільності та потужній спільноті, Java + Spring Boot дозволяє реалізувати продуктивну, масштабовану та безпечну серверну частину.

## 3.4 Опис засобів розробки програмного забезпечення

Для реалізації проекту як основне інтегроване середовище розробки (IDE) було обрано IntelliJ IDEA Ultimate. Вибір цього середовища був обумовлений наявними технологічними обмеженнями, а також необхідністю повної підтримки стеку Java + Spring Boot + PostgreSQL, що використовується у проекті.

IntelliJ IDEA – це професійне середовище розробки, розроблене компанією JetBrains. Воно підтримує мови Java, Kotlin, JavaScript, SQL та інші, і має глибоку інтеграцію з основними фреймворками (Spring, Hibernate, JPA, Maven тощо).

Основні можливості IntelliJ IDEA Ultimate:

* розумне автодоповнення коду, яке аналізує контекст змінних, типи, виклики методів та пропонує релевантні варіанти;
* повна інтеграція зі Spring Boot, включно з автоматичним розпізнаванням конфігурацій, контролерів, шаблонів та профілів;
* зручна панель для підключення до баз даних, можливість виконання SQL-запитів без сторонніх інструментів;
* візуалізація структури проекту, контроль залежностей, генерація коду;
* вбудовані інструменти для роботи з Git/GitLab, що дозволяє зручно управляти версіями, комітами, гілками та злиттями;
* підтримка юніт-тестування (JUnit).

Використання IntelliJ IDEA Ultimate дозволяє значно скоротити час розробки за рахунок автоматизації повторюваних дій, покращення якості коду за допомогою вбудованих підказок та зручного дебагінгу. Крім того, повна інтеграція з Maven спрощує управління залежностями та конфігураціями.

IntelliJ IDEA Ultimate – це сучасне середовище розробки для створення вебзастосунків на основі Java. У контексті даного проекту воно дозволяє повною мірою реалізувати усі вимоги до серверної частини та забезпечує зручну роботу з базою даних та системою контролю версій.

## 3.5 Визначення компонентів та розробка діаграми розгортання

Програмна система для підбору кулінарних рецептів реалізована як трирівнева веб-архітектура, яка включає:

1. рівень клієнта (frontend) – інтерфейс користувача, який взаємодіє з користувачем через браузер;
2. рівень прикладної логіки (backend) – серверна частина, яка обробляє запити, виконує бізнес-логіку та забезпечує доступ до даних;
3. рівень бази даних – централізоване сховище даних, яке містить інформацію про користувачів, рецепти, продукти, категорії, інгредієнти тощо.

Для візуалізації взаємодії між апаратними та програмними компонентами було розроблено діаграму розгортання (Deployment Diagram) у нотації Unified Modeling Language (UML), яка наведена на рис. 3.1.

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

Вміст, створений ШІ, може бути неправильним.

Рисунок 3.1 – Діаграма розгортання

У результаті аналізу вимог було побудовано компонентну архітектуру та визначено основні програмні та апаратні вузли розгортання системи. Розроблена діаграма розгортання відображає фізичне розміщення програмних елементів, їхню взаємодію та канали обміну даними. Така структуризація дозволяє забезпечити високу надійність, масштабованість і зручність супроводу програмного забезпечення.

# 4 Розділ 4

В розділі 4 пропонується навести:

1. порядок підготовки розробленого програмного забезпечення (ПЗ) до застосування. Зокрема, порядок розгортання бази даних (як у 2.5), складу інсталяційного пакету, процедури деплоймента необхідних компонентів на сервері (рис.3.44);
2. інструкції для різних категорій користувачів для застосування розробленого ПЗ за призначенням.

Порядок застосування описується у довільному вигляді. Під час викладення опису доцільно орієнтуватися на вигляд діаграм використання для відповідних категорій користувачів. Також доцільно навести відомості про особливості вводу даних та команд на різних сторінках (формах, екранах) інтерфейсу.

Виходячи із методичних вказівок, цей розділ є необов’язковим, якщо студент виконав в ході курсового проектування тільки розробку проекту ПЗ.

# Висновки

Висновки мають містити такі елементи:

1. нагадування по те, яка задача вирішувалась. Та ціль, для досягнення якої виконана робота;
2. основні результати, які отримані в ході роботи;
3. висновки (не менш трьох), які можна зробити з позиції питань програмної інженерії, які можна зробити за результатами розробки ПЗ;
4. практична цінність отриманих результатів;
5. можливі напрямки вдосконалення (продовження) розробки або досліджень (опційно).

Вимоги СТЗВО-ХПІ-3.01-2021

5.7 Висновки

5.7.1 У висновках повинні бути наведені стислі результати виконаної роботи та пропозиції щодо її використання, а також надана оцінка техніко-економічної ефективності результату роботи та її впровадження.

# Список джерел інформації

1. СТЗВО-ХПІ-3.01-2021. Текстові документи у сфері навчального процесу. Загальні вимоги до виконання. – Введ. 09.12.2021.
2. ДСТУ ГОСТ 7.1.:2006. Бібліографічний запис, бібліографічний опис. Загальні вимоги та правила складання. – Введ. 01.07.2007.
3. Дейт К. Дж.. Введение в системы баз данных; пер.с англ. / К. Дж. Дейт. – 8 изд. – М. : Издательский дом "Вильямс", 2006. – 1328 с.
4. Кнут Дональд Эрвин. Искусство программирования: классический труд: в 3т./ Д. Э. Кнут; общ. ред.; пер. с англ. Ю. В. Козаченко. – 2-е изд., испр. и доп. – М. : Издательский дом "Вильямс", 2004 .– Т. 3. -823 с.
5. Томашевський В.М. Моделювання систем: підручник. / В.М. Томашевський – К.: Видавнича група BHV, 2005. – 352 с.
6. Кібернетика в сучасних економічних процесах : зб. текстів виступів на республік. міжвуз. наук.-практ. конф. / Держкомстат України, Ін-т статистики, обліку та аудиту. – Київ : ІСОА, 2002. – 147 с.
7. Павлов А.А. Принятие решений на основе метода анализа иерархий / А.А. Павлов, Е.И. Лищук // Вестн. НТУ «ХПИ». Сборник научных трудов. Тематический выпуск «Системный анализ, управление и информационные технологии» .– Х.: НТУ «ХПИ» .– 2007 .– №41. – С.69-76
8. Тітов П.С. Аналіз та синтез динамічних процесів у вібраційних пристроях для роботів : дипл. робота магістра : 7.080303 : захищено 12.02.09 / Тітов Петро Семенович. – Х., 2009. – 104 с. – І-13А.05.
9. Гнатуш А. Реинжиниринг: многое в малом // http://citforum.ru/programming/case/gnatush/lot.shtml, 08.11.2018.
10. Кельтон В. Имитационное моделирование. Классика CS. / В. Кельтон. – 3-е изд. – СПб. : Питер ; К. : Издательская группа BHV, 2004 .– 847 с.
11. Язык имитационного моделирования GPSS // http://gpssmodelling.narod.ru/index.html, 21.09.2018.
12. Гультяев А.К. Matlab 5.2. Имитационное моделирование в среде Windows: практ. пособие / А. К. Гультяев. - СПб. : КОРОНА принт, 1999. – 288 с.
13. Arena – система имитационного моделирования // http://www.interface.ru/sysmod/arena.htm, 12.11.2018.
14. A structured approach to enterprise modeling and analysis // http://www.idef.com/, 05.09.2018.

# Додаток А Приклад горизонтального розташування рисунків

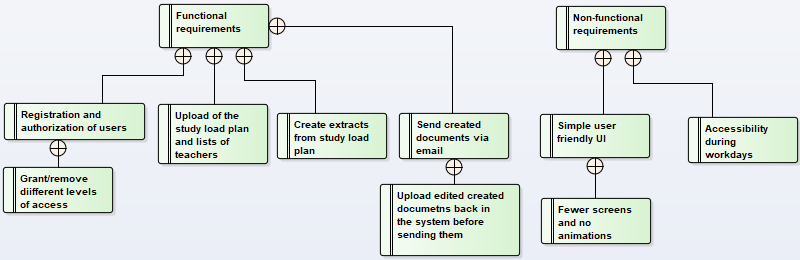


Рисунок Б.1 – Діаграма вимог до програмного рішення для формування витягів з розподілу навчального навантаження

# Додаток Б Приклад додатку із структурними розділами, таблицями та рисунками

Б.1 Порядок проведення експерименту

Послідовність дій під час роботи з програмою більш детально було описано у попередніх розділах. Скорочений варіант послідовності є наступним:

1. завантажити план навчального навантаження на викладачів;
2. завантажити перелік викладачів та детальних даних про них;
3. завантажити перелік викладачів які є керівниками студентів з курсовими або дипломними роботами призначеними до них;
4. натиснути кнопку «Сформувати відомість учбових доручень»;
5. натиснути кнопку «Сформувати індивідуальні плани».;
6. натиснути кнопку «Сформувати розподіл навантаження по викладачах».

Формування витягів з використанням програми було вирішено додатково провести не тільки на тих самих даних, що використовувались для формування витягів без використання програми, але ще й на спеціально підготовлених планах навчального навантаження на різні кількості викладачів для демонстрування переваг для різних розмірів кафедр.

Б.2 Результати формування витягів з використанням програми

Б.2.1 Вимірювання часу витраченого на завантаження плану

Символами D будуть позначатись різні умовні кафедри (англ. Department) з різною кількістю викладачів на кожній із них, номери з 1 по 6 позначають кафедри з 59, 33, 21, 17, 10 та 7 викладачами відповідно. Вимірювання часу витраченого на завантаження плану навчального навантаження у програму у відповідності до кількості викладачів наведено у таблиці Б.1.

Таблиця Б.1 – Зчитування плану навчального навантаження програмою

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № експерименту | D1 | D2 | D3 | D4 | D5 | D6 |
| 1 | 29.23 | 17.01 | 12.01 | 9.67 | 7.73 | 7.53 |
| 2 | 28.69 | 17.99 | 11.28 | 8.60 | 6.94 | 6.22 |
| 3 | 30.43 | 17.00 | 11.95 | 8.92 | 7.94 | 5.32 |
| 4 | 29.49 | 18.14 | 10.74 | 10.74 | 7.78 | 7.22 |
| 5 | 28.54 | 16.61 | 11.90 | 10.59 | 7.03 | 4.99 |
| 6 | 28.18 | 18.21 | 12.06 | 9.95 | 8.70 | 6.26 |
| 7 | 28.50 | 16.63 | 12.41 | 10.06 | 6.79 | 5.34 |
| 8 | 29.78 | 18.77 | 10.46 | 10.11 | 7.08 | 5.46 |
| 9 | 29.89 | 16.51 | 11.17 | 9.08 | 6.78 | 6.50 |
| 10 | 28.88 | 18.61 | 12.61 | 10.88 | 8.29 | 4.91 |
| Сума | 23.64 | 12.32 | 6.21 | 4.46 | 2.65 | 1.42 |

Б.2.2 Вимірювання часу витраченого на завантаження плану

Виміри зчитування переліку викладачів наведено у таблиці Б.2.

Таблиця Б.2 – Час витраченний на зчитування переліку викладачів

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № експерименту | D1 | D2 | D3 | D4 | D5 | D6 |
| 1 | 5.47 | 5.14 | 7.20 | 5.43 | 4.76 | 5.57 |
| 2 | 4.88 | 7.14 | 5.35 | 7.43 | 4.48 | 4.34 |
| 3 | 3.69 | 6.16 | 5.92 | 3.52 | 4.97 | 4.02 |
| 4 | 5.30 | 5.57 | 6.40 | 6.79 | 5.57 | 5.40 |
| 5 | 5.99 | 4.18 | 3.92 | 4.35 | 5.64 | 7.18 |
| 6 | 7.48 | 5.63 | 5.47 | 5.02 | 8.18 | 5.26 |
| 7 | 4.45 | 3.84 | 6.49 | 6.45 | 4.08 | 6.96 |
| 8 | 7.19 | 5.13 | 4.41 | 6.81 | 5.53 | 5.04 |
| 9 | 3.69 | 6.39 | 6.49 | 3.79 | 7.62 | 5.75 |
| 10 | 7.50 | 5.67 | 5.89 | 5.65 | 4.38 | 6.03 |

Б.2.3 Вимірювання часу витраченого на завантаження переліку викладачів

Вимірювання часу витраченого на завантаження переліку викладачів що є керівниками у програму наведено у таблиці Б.3.

Таблиця Б.3 – Час витрачений на зчитування переліку викладачів та дипломників

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № експерименту | D1 | D2 | D3 | D4 | D5 | D6 |
| 1 | 6.37 | 3.96 | 4.11 | 4.28 | 5.71 | 6.22 |
| 2 | 4.28 | 6.25 | 4.05 | 4.44 | 5.61 | 3.61 |
| 3 | 5.01 | 6.41 | 4.16 | 4.06 | 6.26 | 5.10 |
| 4 | 5.75 | 4.76 | 6.01 | 6.18 | 5.19 | 5.46 |
| 5 | 4.26 | 3.93 | 5.04 | 5.26 | 4.46 | 4.86 |
| 6 | 5.53 | 3.90 | 6.37 | 4.53 | 5.68 | 5.29 |
| 7 | 4.62 | 3.53 | 4.43 | 4.76 | 4.37 | 4.19 |
| 8 | 5.90 | 3.57 | 3.60 | 5.49 | 4.70 | 4.17 |
| 9 | 5.68 | 4.81 | 5.87 | 5.72 | 4.19 | 4.09 |
| 10 | 4.04 | 4.28 | 3.59 | 4.42 | 4.49 | 5.80 |

…

Б.5 Візуалізація результатів експериментів

Графіки залежності загального часу формування витягів від кількості викладачів на кафедрі представлені на рисунках Б.1 та Б.2.

Рисунок Б.1 – Графік залежності часу формування від кількості викладачів з використанням програми

Оцінка отриманого графіку дозволяє стверджувати, що автоматизоване формування витягів з плану навчального навантаження здійснюється набагато швидше, порівнюючи з поточним підходом до його отримання. Безсумнівним цей висновок для кафедри із кількістю посад 59. Прогнозований час формування витягів з розподілу навчального навантаження без використання програми приблизно складає більше 10 годин. Розроблений застосунок дозволяє отримати більш якісний результат менш чим за 2 хвилини.

# Додаток В Приклад змісту пояснювальної записки до курсового проєкту

В додатку представлений варіант структури розділів пояснювальної записки до курсового проєкту на прикладі роботи на тему «Проектування програмного забезпечення для системи аудиту програмного забезпечення кафедри».

Вступ

Перелік познак та скорочень

Розділ 1. Обґрунтування необхідності автоматизації обліку програмного забезпечення кафедри

1.1 Аналіз сучасного стану організації обліку програмного забезпечення кафедри

Представлений аналіз існуючого стану бізнес-процесів із детальним описом та побудовою моделі процесів планування розподілу аудиторій, в якому використовуються відомості про комп’ютери та встановлене програмне забезпечення, із застосуванням BPMN.

Розроблена доменна модель предметної області та глосарій.

1.2 Аналіз сучасних систем аудиту програмного забезпечення із розробкою пропозицій щодо впровадження на кафедрі

Описані сучасні вимоги користувачів, які зацікавлені у відомостях про оснащення комп’ютерів у аудиторіях, запропоновані зміни до системи обліку, представлена модифікована модель процесу планування розподілу аудиторій, що враховує оснащення комп’ютерною технікою із застосуванням BPMN.

1.3 Розробка системи вимог до системи аудиту програмного забезпечення

1.3.1 Опис призначення системи

1.3.2 Визначення функціональних та нефункціональних вимог до розроблюваного програмного забезпечення

1.3.3 Опис етапів розробки програмного забезпечення

Розділ 2. Проектування програмного забезпечення системи

2.1 Розробка діаграм поведінки

2.1.1 Розробка діаграм варіантів використання для різних категорій користувачів

2.1.2 Уточнення реалізації варіантів використання із розробкою діаграм взаємодії

2.2 Розробка бази даних для зберігання інформації

2.2.1 Визначення бізнес-правил предметної області

2.2.2 Розробка структури бази даних

Розділ 3 Обґрунтований вибір стеку технологій для створення системи

3.1 Вибір архітектури системи

3.2 Вибір формату системи керування базами даних

3.3 Опис мови програмування та технологій для створення систем

3.4 Визначення засобів розробки програмного забезпечення

3.5 Визначення компонентів та розробка діаграми розгортання

Розділ 4 Застосування програмного забезпечення для введення інформацію про комп’ютери та програмне забезпечення кафедри

4.1 Робота інженерів кафедри в ході введення даних

4.2 Робота викладачів і відомостями про оснащення комп’ютерних класів

4.3 Формування IT-паспорту аудиторії

Висновки

Список джерел інформації