МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМ. С.З.ГЖИЦЬКОГО

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему: **“ Реалізація інтуїтивного інтерфейсу користувача для підвищення ефективності CRM-системи ”**

Виконав: ст. гр. Іт-62_____

Спеціальності 126 – «Інформаційні системи та технології»_____
(шифр і назва)

Городечний Володимир Володимирович_____
(Прізвище та ініціали)

Керівник: к.т.н., доцент Луб Павло Миронович
(Прізвище та ініціали)

Рецензент: _____
(Прізвище та ініціали)

ДУБЛЯНИ-2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМ. С.З.ГЖИЦЬКОГО

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Другий (магістерський) рівень вищої освіти
126 – «Інформаційні системи та технології»

"ЗАТВЕРДЖУЮ"
Завідувач кафедри_____
д.т.н., проф. А.М. Тригуба
"____"_____2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу студентці

Городечний Володимир Володимирович

1. Тема роботи: «Реалізація інтуїтивного інтерфейсу користувача для підвищення ефективності CRM-системи»

Керівник роботи: Луб Павло Миронович, к.т.н., доцент.
Затверджені наказом по університету №140/к-с від 28.02.2025.

2. Строк подання студентом роботи 01.12.2025 р.

3. Початкові дані до роботи: 1. Правила створення інтуїтивного інтерфейсу користувача. 2. Архітектура CRM-системи. 3. Monday work platform. 4. Методика тестування ПЗ.

4. Зміст розрахунково-пояснювальної записки:

1. Аналіз систем управління взаємовідносинами з клієнтами
2. Підходи у проектуванні інтерфейсу користувача
3. Методика проектування додатку з інтуїтивним інтерфейсом користувача
4. Використання додатку та комплексне тестування
5. Охорона праці та безпека в надзвичайних ситуаціях
Висновки та пропозиції.
Бібліографічний список.
Додатки.

5. Перелік презентаційного матеріалу 1–2 – Тема, мета, завдання дипломної роботи; 3 – Призначення та аналіз завдань; 4 – Можливості CRM-системи ; 5 – Аналіз вес-сайтів; 6 – Використання інструментарію Monday Work Platform; 7 – Технічне завдання; 8 – Реалізація структури додатку; 9 – Розробка функціональності; 10 – Практичне використання додатку; 11 – Автоматизоване тестування UI; 12 - Головні висновки.

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 4	*Луб П.М., доцент кафедри інформаційних технологій*		
5	*Городецький І.М., доцент кафедри інженерної механіки*		

Дата видачі завдання 28.02.2025.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Етапи виконання кваліфікаційної роботи	Строки виконання роботи	Приміт-ка
1.	*Написання першого розділу та означення головних завдань роботи*	*01.03-01.05.25*	
2.	*Виконання другого розділу та опис інформаційних технологій для виконання завдань роботи*	*01.03-01.05.25*	
3.	*Виконання третього розділу, методика вирішення завдань та елементи наукових досліджень*	*01.05-01.07.25*	
4.	*Виконання четвертого розділу, представлення результатів та їх узагальнення*	*01.05-01.07.25*	
5.	*Написання розділу: «Охорона праці та безпека в надзвичайних ситуаціях»*	*01.07-01.09.25*	
6.	*Завершення оформлення розрахунково-пояснювальної записки та презентаційних матеріалів*	*01.09-01.11.25*	
7.	*Завершення роботи в цілому*	*01.11-01.12.25*	

Студент _____ Городечний В.В.
(підпис)

Керівник роботи _____ Луб П.М.
(підпис)

УДК: 004.8:004.9

Кваліфікаційна робота: 66 с. текст. част., 25 рис., 6 табл., 12 слайдів, 19 джерел.

Реалізація інтуїтивного інтерфейсу користувача для підвищення ефективності CRM-системи. Городечний В.В. Кафедра ІТ. – Дубляни, Львівський НУВМБ, 2025.

Визначено основні завдання CRM-систем та їх ролі у підвищенні ефективності взаємодії з клієнтами. Виконано аналіз можливостей сучасних CRM-рішень, класифікацію їх за видами та функціональними особливостями.

Розглянуто популярні веб-сайти та сервіси, що надають CRM-функціонал, із метою оцінки зручності та практичної користі для бізнесу.

Наведено принципи створення дружніх для користувача інтерфейсів (<user friendly>), а також виявляються переваги й недоліки інтерфейсів існуючих CRM-систем. Представлено інструментарій платформи Monday work platform для проектування ефективного та інтуїтивного користувацького інтерфесу.

Вибрано технологію та інструментарій для створення додатку. Визначено технічне завдання. Подано підхід до реалізації структури додатку, розробки макетів та основних функціональних модулів.

Виконано та наведено методику автоматизованого тестування UI за допомогою Jest і React Testing Library. Описано практичне застосування додатку з інтуїтивним інтерфейсом, що дозволяє користувачам ефективно взаємодіяти з CRM-даними та оцінити стабільність, продуктивність і зручність реалізованого рішення.

Ключові слова: інтерфейс користувача, CRM, інтеграція, програмування, тестування, UI.

ЗМІСТ

ВСТУП

У сучасних умовах український бізнес стикається з численними викликами, пов’язаними як із економічною нестабільністю, так і з наслідками воєнного конфлікту, що значно ускладнює ведення підприємницької діяльності та оптимізацію внутрішніх процесів. Незважаючи на це, аграрний сектор залишається однією з ключових рушійних сил економіки України, демонструючи високий потенціал для зростання та впровадження інновацій.

Одним із ефективних засобів підвищення продуктивності та конкурентоспроможності компаній є впровадження сучасних IT-сервісів, зокрема систем управління взаємовідносинами з клієнтами (CRM), які дозволяють оптимізувати процеси продажу, аналітики та взаємодії з партнерами.

Цифровізація бізнес-процесів стає не просто трендом, а необхідністю для адаптації до динамічного ринку, зменшення витрат часу та ресурсів і підвищення ефективності управління. У цьому контексті особливо важливим є розроблення інтуїтивного інтерфейсу користувача, який забезпечує зручність і швидкість роботи з CRM-системою, сприяючи ефективнішому прийняттю рішень і підвищенню продуктивності персоналу.

Мета роботи – реалізувати інтуїтивний інтерфейс користувача в CRM-системі завдяки розробці персоналізованого додатку для онлайн-платформи організації роботи компаній і команд.

Завдання роботи:

- проаналізувати системи управління взаємовідносинами з клієнтами;
- розкрити підходи у проектуванні інтерфейсу користувача;
- навести методику та виконати проектування додатку з інтуїтивним інтерфейсом;
- застосувати додаток та здійснити його комплексне тестування.

Об’єкт роботи – CRM-система та процеси управління взаємовідносинами з клієнтами в цифровому середовищі, зокрема взаємодія користувача з даними і функціональні можливості інтеграційних додатків.

Предмет роботи – інтуїтивний інтерфейс користувача для CRM-системи, який забезпечує автоматизацію збирання та обробки даних, підвищення ефективності взаємодії користувачів із системою та оптимізацію бізнес-процесів у платформі monday.com.

Новизна результатів:

- запропоновано багатофазну архітектуру синхронізації даних (Snapshot, Sync, Finalize) для надійної обробки великих обсягів інформації;
- реалізовано інтеграцію з сервісом Crunchbase для автоматичного збагачення CRM-даних актуальною бізнес-інформацією;
- розроблено інтуїтивний інтерфейс користувача з кроковим налаштуванням «рецептів», що спрощує роботу та зменшує ризик помилок;
- використано автоматизоване тестування UI (Jest + React Testing Library) для забезпечення стабільності та продуктивності додатку.

РОЗДІЛ 1
АНАЛІЗ СИСТЕМ УПРАВЛІННЯ ВЗАЄМОВІДНОСИНАМИ З КЛІЄНТАМИ

1.1. Призначення та аналіз головних завдань CRM

Система управління взаємовідносинами з клієнтами (CRM – *Customer Relationship Management*) – це комплекс програмних рішень, методик і організаційних підходів, що використовуються для систематичного керування взаємодією компанії з поточними та потенційними користувачами її продуктів чи послуг [3, 5].

Головна функція CRM-платформи полягає в тому, щоб об’єднати всі дані про клієнтів в єдиному інформаційному середовищі, автоматизувати процеси продажу, маркетингові активності та сервісну підтримку, а також забезпечити узгоджену комунікацію між співробітниками, які працюють із клієнтськими запитами.

CRM-системи виступають базовим інструментом для компаній, що орієнтуються на клієнтоцентричну модель ведення бізнесу. Вони дають змогу накопичувати, структурувати та інтерпретувати дані про клієнтів, що, у свою чергу, підтримує прийняття обґрунтованих управлінських рішень у сферах маркетингу, продажів та післяпродажного супроводу.

Серед ключових функцій CRM-систем можна виокремити (рис. 1.1) [5]:

- *централізоване зберігання даних про клієнтів.* Усі контакти, історія комунікацій, здійснені покупки та звернення акумулюються в єдиній базі даних;

- *підвищення продуктивності відділу продажів.* CRM дає змогу відстежувати стадії опрацювання угод, налаштовувати автоматичні нагадування та забезпечувати своєчасну реакцію на запити клієнтів;

- *автоматизацію типових рутинних операцій.* Це зменшує операційне навантаження на персонал і дозволяє зосередитися на більш складних і пріоритетних задачах;

- *інструменти аналітики та формування звітності.* Система генерує докладні аналітичні зрізи, що допомагає виявляти проблемні ділянки бізнес-процесів та коригувати стратегію розвитку компанії;

- *підвищення рівня сервісу*. Завдяки швидкому доступу до повної історії взаємодії з клієнтом, компанія може оперативно реагувати на звернення, персоналізувати комунікацію та суттєво покращувати якість обслуговування.

Рисунок 1.1 – Ключові функції CRM-Системи

Особливо відчутну цінність така програма дає тоді, коли база контактів стрімко зростає, а внутрішні процеси команди потрібно зробити прозорими, керованими та прогнозованими. Розглянемо кілька ключових причин, чому доцільно інтегрувати CRM-систему в роботу компанії.

Централізоване зберігання історії взаємодії з клієнтом. Уся інформація про контакт – від першого звернення до закриття угоди – фіксується в CRM.

Перемовини, інтерес до певних продуктів чи сервісів, уточнення умов – усе структуровано й доступно для всієї команди. Це особливо важливо в ситуаціях, коли менеджер звільняється або переходить на іншу посаду: дані не губляться в особистих блокнотах чи месенджерах, а залишаються в системі.

Автоматизація нагадувань і задач. CRM-система бере на себе значну частину рутинної роботи: нагадує про заплановані дзвінки та зустрічі, сигналізує про важливі дати, формує підказки щодо наступних дій у межах угоди. Завдяки цьому менеджер може більше часу інвестувати у живу комунікацію з клієнтом, а не в ручне ведення календарів і списків завдань.

Єдина стрічка комунікацій. CRM надає можливість переглядати всю історію контакту із замовником у вигляді послідовної таймлайнової стрічки: коли було надіслано електронний лист, про що йшлося на останній зустрічі, які комерційні пропозиції чи документи передавалися. Це спрощує роботу як для окремого менеджера, так і для керівника, який відстежує статус угод і якість опрацювання лідів.

Орієнтири на кожному етапі воронки продажів. CRM виступає як інтерактивний «навігатор» для менеджера: веде по етапах воронки, підказує стандартні наступні кроки, формує типові сценарії дій. Такий підхід однаково корисний як для нових співробітників, які тільки входять у процес, так і для досвідчених фахівців, що прагнуть систематизувати роботу та мінімізувати ризик пропуску важливих дій.

Окремим напрямом розвитку є інтеграція в CRM-системи технологій штучного інтелекту та машинного навчання. Вони використовуються для прогнозування поведінки клієнтів, виявлення закономірностей у даних та автоматизації рутинних операцій. Інтелектуальні рекомендації, скоринг лідів, автоматичне сегментування аудиторії, побудова прогнозних моделей продажів – усе це підвищує точність управлінських рішень і рівень персоналізації сервісу.

У контексті розробки спеціалізованого програмного компонента для CRM-системи особливої актуальності набуває орієнтація на зручний графічний інтерфейс користувача (GUI). Чітка візуальна структура, інтуїтивна логіка

переходів, можливість гнучкого налаштування під конкретні бізнес-процеси та ролі користувачів істотно впливають на продуктивність роботи з системою та готовність співробітників активно її використовувати [17].

Отже, сучасний розвиток CRM-технологій охоплює не лише нарощування функціональних можливостей, а й системне покращення користувацького досвіду. Це формує практичний запит на створення спеціалізованих інтерфейсів, адаптованих до конкретної CRM-платформи та сценаріїв її використання, що й визначає предмет кваліфікаційної роботи.

1.2. Можливості CRM-систем та їх види

У сучасних програмних системах додаток – це самостійний програмний компонент, що реалізує певну функцію або набір функцій у межах конкретної платформи чи операційної системи. Додатки можуть працювати як окремі програми або бути частиною більшої екосистеми, забезпечуючи користувачеві доступ до сервісів та даних інформаційної інфраструктури підприємства.

У контексті хмарних CRM-рішень, зокрема платформи Monday.com [14], додаток – це розширення (часто позначається як "*app*" або "*integration*"), яке підключається до базового ядра системи, додає нові можливості, дозволяє взаємодію з зовнішніми API або реалізує індивідуальну бізнес-логіку на рівні коду.

Сучасні програми здатні суттєво посилити маркетинг, покращити сервіс і створити єдину екосистему для бізнесу.

Сегментація клієнтів і персоналізований підхід. За допомогою CRM можна розбивати базу клієнтів на сегменти — за інтересами, стадіями воронки, джерелами трафіку чи історією покупок. Це дозволяє надсилати релевантні пропозиції саме тим, кому вони цікаві.

Масові розсилки й автоворонки. З CRM можна запускати email або SMS-розсилки без сторонніх сервісів. А ще — будувати автоматизовані ланцюжки

листів (автоворонки), які самостійно ведуть клієнта від знайомства з брендом до покупки.

Інтеграція з іншими системами. CRM легко під'єднується до вашого сайту, систем обліку, телефонії, платіжних рішень, сервісів доставки тощо. Це створює єдиний робочий простір, де дані синхронізовані, а процеси — узгоджені.

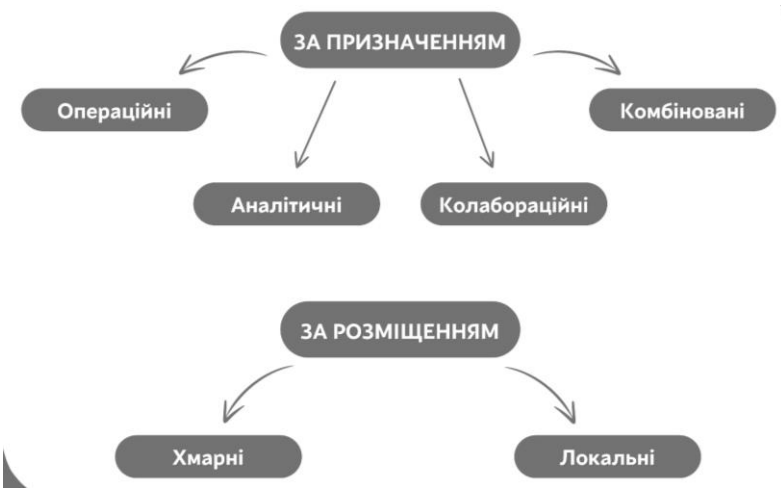
Підтримка клієнтів і сервіс. CRM система може включати модулі підтримки: створення звернень, фіксацію проблем, контроль термінів відповіді. Це важливо для компаній, які цінують сервіс і прагнуть тримати якість обслуговування на високому рівні.

Мобільність і доступ з будь-якої точки світу. Більшість CRM мають мобільні додатки. Це означає, що менеджер може працювати з клієнтами, створювати задачі чи переглядати статус угоди навіть у дорозі — все, що потрібно, завжди під рукою.

Безпека та контроль доступу. Програми управління дозволяють чітко налаштувати, хто і до якої інформації має доступ. Це захищає комерційні дані та дає змогу контролювати дії співробітників у системі.

Гнучкість під бізнес. Деякі CRM легко налаштовуються під ваші процеси. Можна додавати власні поля, статуси, сценарії роботи — без складного програмування. А якщо потрібно, інтегрувати з кастомними рішеннями.

CRM може зростати разом із бізнесом, адаптуватися під нові задачі й допомагати вам масштабуватись без хаосу й втрат.



У середовищі CRM-платформ додатки виконують роль механізму кастомізації та масштабування стандартного функціоналу, даючи змогу налаштувати систему під конкретні процеси компанії. Вони можуть бути внутрішніми (розробленими IT-командою самої організації) або зовнішніми (створеними

незалежними розробниками чи офіційними партнерами платформи).

Розуміння класифікації та типів додатків дає змогу більш усвідомлено проєктувати архітектуру CRM-рішення, обирати відповідні технологічні стеки та визначати оптимальні способи інтеграції потрібного функціоналу.

Таблиця 1.1 – Порівняльна таблиця типів додатків у CRM-платформах

Тип додатку	Призначення	Приклади використання
1	2	3
Інтеграційний	З'єднання з зовнішніми системами або базами даних	Підключення до Crunchbase, Google Sheets, Salesforce
Автоматизаційний	Реалізація реакцій на події або умовні сценарії	Авто-зміна статусу, надсилання листів, оновлення полів
Інтерфейсний	Надання користувачеві візуальних елементів для взаємодії	Дашборди, форми, таблиці з виводом додаткових даних
Системний	Розширення внутрішніх можливостей платформи	Авторизація, логування, контроль доступу

До основних типів CRM-систем відносять [5]:

1. **Операційні.** Служать для автоматизації щоденних процесів: ведення клієнтської бази, обробки лідів, комунікацій, виставлення рахунків, супроводу угод. Їх головна мета – забезпечити структуровану й контрольовану роботу з клієнтами.

2. **Аналітичні.** Орієнтовані на збір і аналіз даних. Дозволяють формувати звіти та візуалізації, досліджувати поведінку клієнтів, ефективність каналів, сегментувати базу, будувати прогнози й ухвалювати рішення на основі метрик.

3. **Колабораційні.** Забезпечують взаємодію між підрозділами, синхронізуючи роботу продажів, маркетингу й підтримки. Надають спільний доступ до профілю клієнта та історії комунікацій, що допомагає уникати дублювання й будувати довгострокові відносини.

4. **Хмарні та локальні.** Хмарні CRM працюють онлайн на інфраструктурі провайдера, доступні через браузер або мобільний застосунок. Локальні встановлюються на сервери компанії, не залежать від сторонніх сервісів, але потребують більше технічної підтримки.

5. **Галузеві.** Створюються для конкретних сфер (медицина, освіта, нерухомість, автосалони тощо) і вже містять типові сутності, бізнес-сценарії та функції, адаптовані під робочі процеси галузі.

6. **Комбіновані.** Поєднують можливості кількох типів: операційний блок, аналітику та колаборацію. Дають змогу вести контакти, автоматизувати операції, формувати звіти, працювати командно, налаштовувати воронки, запускати розсилки та інтегруватися з іншими системами.

Перевага комбінованих CRM у тому, що їх можна масштабувати разом із компанією: почати з базового набору функцій, а згодом підключати додаткові модулі й інтеграції без повної заміни платформи. Найбільш корисною система буде для бізнесів, які прагнуть мати єдине рішення для кількох напрямів діяльності (продажі, маркетинг, сервіс) і планують поетапно розвивати свою CRM-інфраструктуру.

1.3 Аналіз веб-сайтів із сервісами CRM

Сьогодні представлено велику кількість CRM-рішень, орієнтованих на автоматизацію процесів управління взаємодією з клієнтами. Серед найбільш поширених можна виокремити такі платформи, як Monday CRM, HubSpot CRM, Zoho CRM, Salesforce та Pipedrive. Для зазначених систем у цьому розділі буде виконано порівняльний аналіз за такими ключовими параметрами [8, 14]: 1) функціональність; 2) інтерфейс; 3) можливості кастомізації; 4) орієнтація на кінцевого користувача.

У таблиці 1.2 подано зіставлення функціональних можливостей обраних CRM-платформ.

Аналіз доступних інструментів цих систем показує, що більшість із них надають розширений набір засобів для роботи з лідами, автоматизації воронки продажів та аналітичної обробки даних.

Таблиця 1.2 – Порівняння функціональності CRM платформ

№ п/п	Платформа	Основні модулі	Автоматизація без коду	Аналітика та звіти	AI / ML-можливості
1	Monday CRM	Продажі, маркетинг, підтримка, управління проєктами	Правила «якщо-те», шаблони, інтеграції	Візуальні дашборди, кастомні KPI	Обмежені (рекомендації на дошках)
2	HubSpot CRM	Контакти, угоди, email-маркетинг	Робочі процеси у платних «Hubs»	Вбудована вирва продажів, маркетингові метрики	AI-інструменти для контенту (Marketing Hub)
3	Zoho CRM	Продажі, маркетинг, сервіс	Сценарії, макроси, правила	Гнучкі звіти, Canvas-дашборди	Zia AI-помічник (прогнози, чати)
4	Salesforce	Sales, Service, Marketing, Commerce	Flow Builder, Process Builder	Розширена аналітика, Tableau CRM	Einstein AI (прогнози, боти)
5	Pipedrive	Продажі	Простенькі тригери та шаблони email	Базові звіти, воронка	AI-прогноз імовірності угоди

Платформи рівня Salesforce та Zoho CRM надають розширений, «ентерпрайзний» функціонал, однак їхнє впровадження зазвичай вимагає глибших технічних компетенцій або участі розробників, що ускладнює їх використання в невеликих компаніях із обмеженими ресурсами. Натомість HubSpot CRM та Pipedrive орієнтовані на простоту експлуатації, але пропонують суттєво скромніші можливості в частині кастомізації та масштабування функціоналу.

У контексті розробки додатку поверх Monday Work OS її системна архітектура надає достатній рівень гнучкості та стабільний API, що дозволяє підключати власні модулі, розширення та інтерфейси без порушення базової логіки платформи. Завдяки цьому Monday Work OS виступає придатним середовищем для реалізації CRM-рішень із кастомним графічним інтерфейсом (GUI).

Важливою складовою ефективності будь-якої CRM-системи є UX/UI-рівень, тобто зручність та логічність взаємодії користувача з інтерфейсом. Інтуїтивно

зрозуміла, візуально гнучка платформа сприяє зростанню продуктивності персоналу та скорочує час інтеграції інструменту в існуючі бізнес-процеси.

З метою візуального зіставлення було проаналізовано п’ять популярних CRM-рішень: Monday CRM, HubSpot, Zoho CRM, Salesforce та Pipedrive.

На рисунку 1.2 наведено оцінки за двома основними критеріями: 1) *візуальна гнучкість* – наскільки просто змінювати подання полів, дошок, дашбордів та конфігурацій без написання коду; 2) *простота навчання* – обсяг часу та зусиль, потрібних новому користувачу для впевненого старту роботи в системі (вища оцінка відповідає легшому освоєнню).

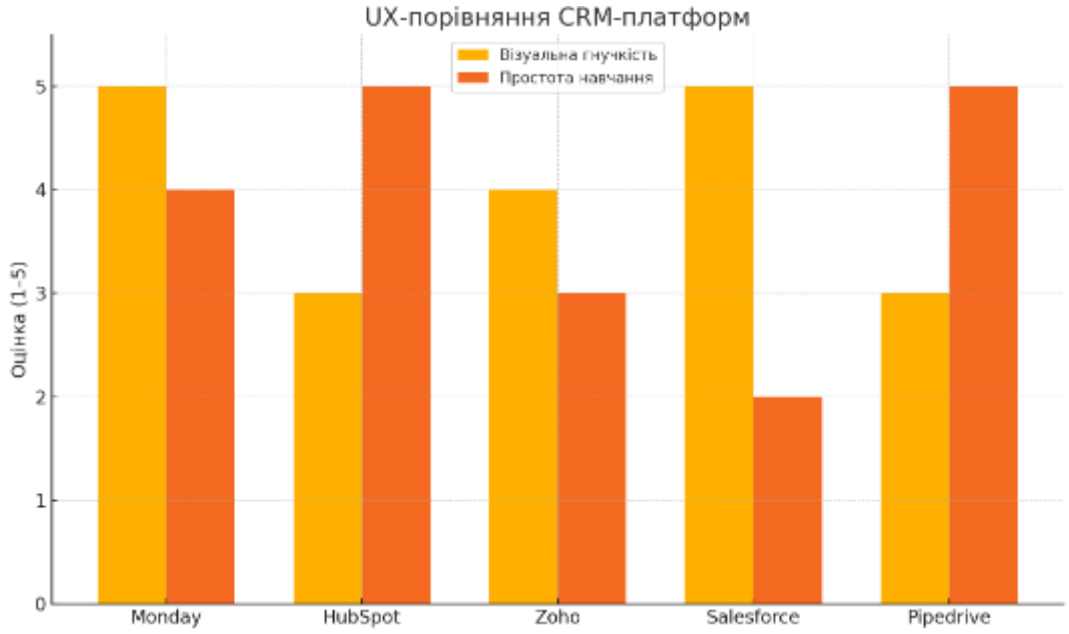


Рисунок 1.2 – UX-порівняння CRM-платформ [5]

Monday CRM демонструє максимальну гнучкість (5) при порівняно низькій кривій навчання (4). Drag-and-drop та яскравий UI дозволяють новим користувачам швидко адаптуватися.

HubSpot CRM орієнтований на новачків (5), проте обмежений у кастомізації інтерфейсу (3), що робить його оптимальним для команд «з коробки».

Zoho CRM пропонує баланс між гнучкістю (4) та середнім порогом входу (3). Інтерфейс багатий функціями, але щільний за елементами.

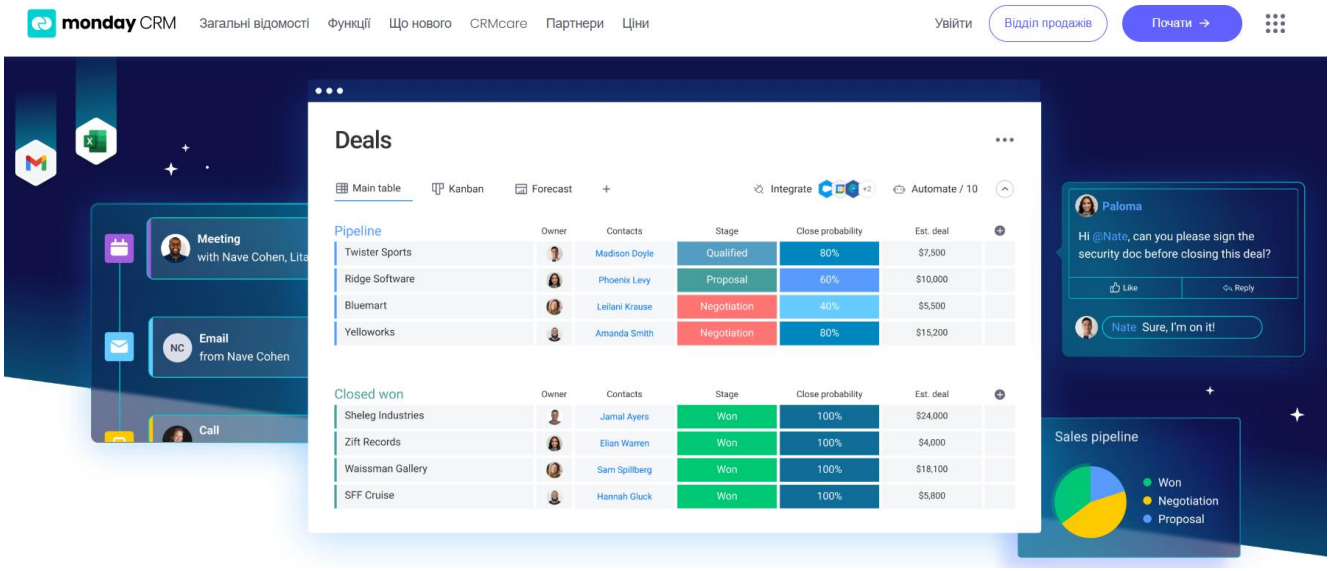


Рисунок 1.3 – Головне вікно Monday CRM

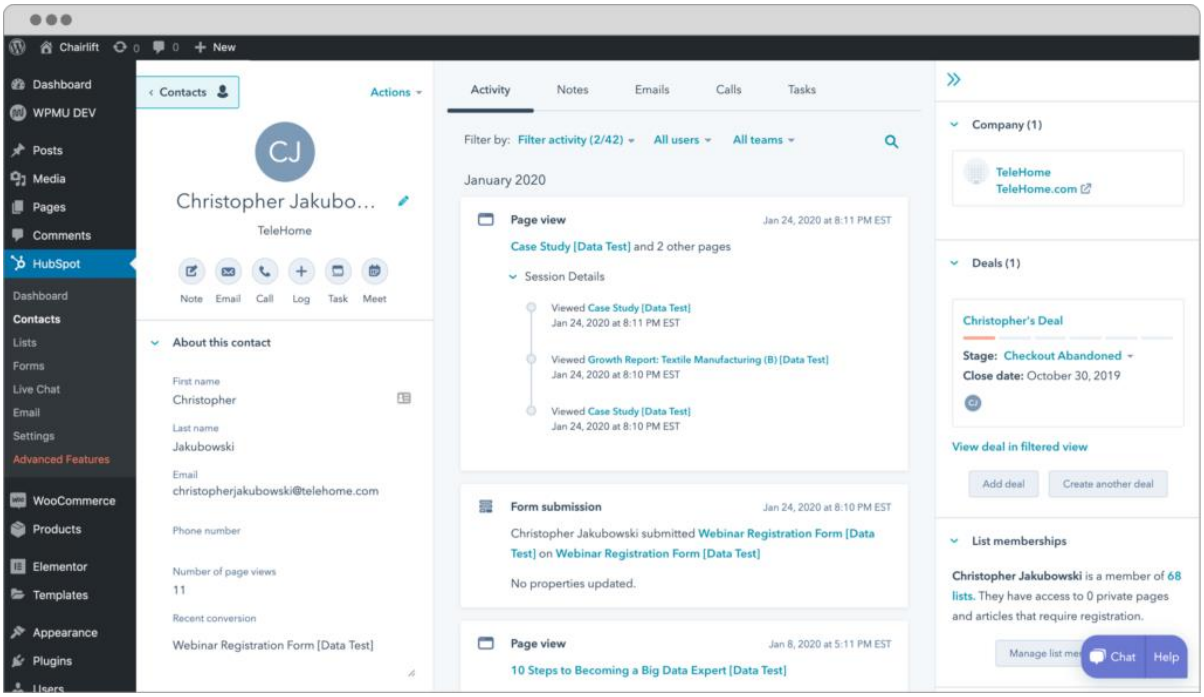


Рисунок 1.4 – Головне вікно HubSpot CRM

Salesforce лідирує за кастомізацією (5), однак має високий поріг входу (2) через складність меню та необхідність освоєння Apex і LWC.

Pipedrive дуже простий для освоєння (5) завдяки фокусі на воронці продажів, але пропонує базові можливості візуального налаштування (3).

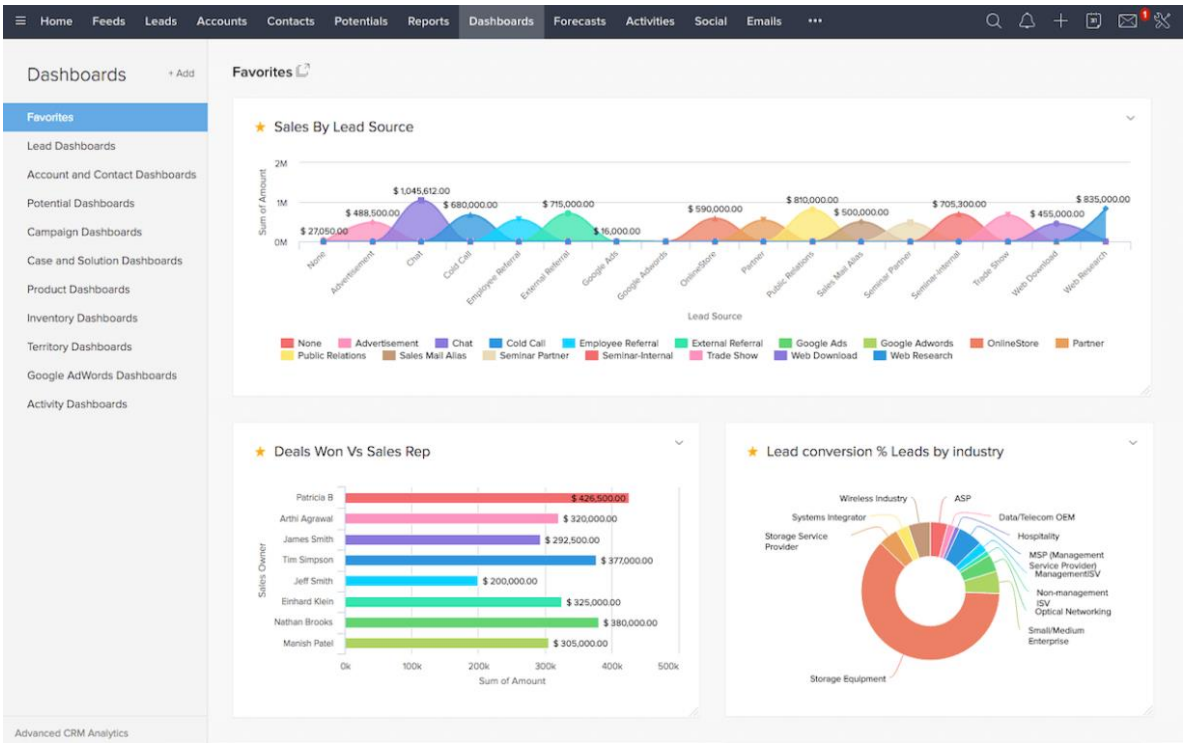


Рисунок 1.5 – Головне вікно Zoho CRM

Результати порівняння підтверджують доцільність використання Monday CRM як базової платформи для розробки GUI-додатку.

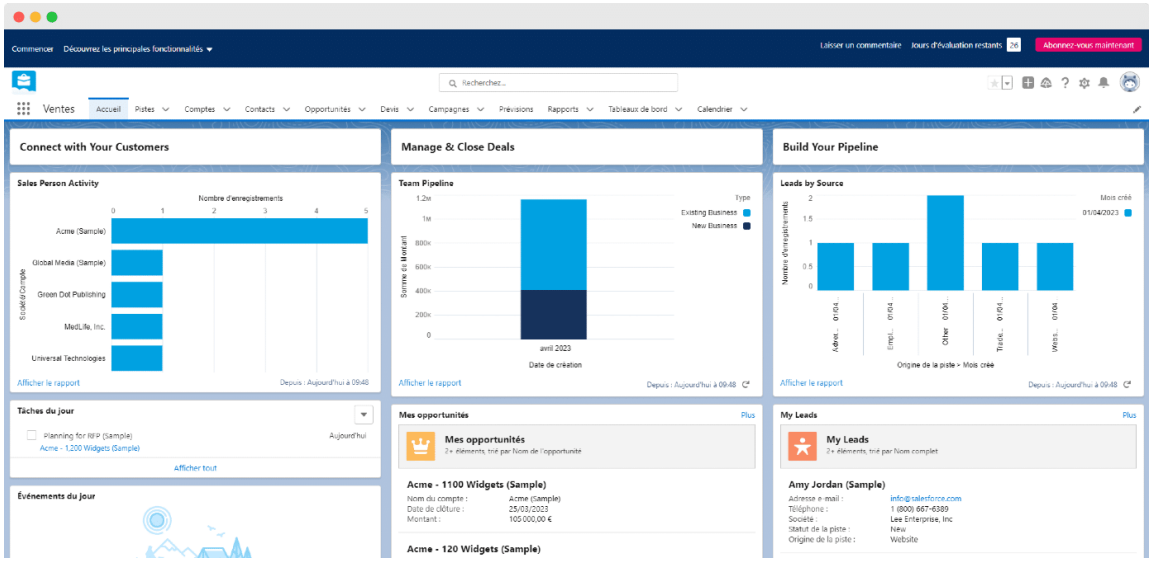


Рисунок 1.6 – Головне вікно Salesforce CRM

Завдяки поєднанню високої адаптивності інтерфейсу та дружності до користувача, майбутній інтерфейс повинен:

- забезпечувати швидке освоєння без технічної підготовки;
- базуватися на знайомих елементах взаємодії (таблиці, статуси, індикатори);
- підтримувати гнучке налаштування логіки, етапів, полів та візуального представлення даних;
- відповідати стилістиці Monday Work OS (кольорові статуси, drag-and-drop, інтерактивні віджети).

Таким чином, аналіз інтерфейсів CRM-систем формує основу для проектування власного графічного інтерфейсу під розробку CRM-додатку на Monday Work OS, що є предметом даної кваліфікаційної роботи.

РОЗДІЛ 2
ПІДХОДИ У ПРОЕКТУВАННІ ІНТЕРФЕЙСУ КОРИСТУВАЧА

2.1. Правила <*user friendly*> інтерфейсу користувача

Інтерфейс – це такий собі «місток» між користувачем і системою. За допомогою інтерфейсу користувач зможе пояснити системі, чого він хоче від неї, а система це виконає.

Ось так поводяться інтернет-користувачі за даними Online Marketing Institute [5]:

- 85% можуть піти з сайту, якщо їм не сподобається дизайн інтерфейсу;
- 83% покинуть сайт, якщо будуть змушені робити багато кліків, щоб знайти те, що їм потрібно;
- 40% ніколи не повернуться на сайт, якщо їм було важко використовувати його вперше.

Принципи хорошого інтерфейсу однакові що у вебсайтів, що й у програм, що й у сервісів. Ось 17 основних правил <*user friendly*> інтерфейсу користувача [6]: 1) бути інтуїтивно зрозумілим; 2) бути передбачуваним; 3) бути мінімалістичним; 4) швидко завантажуватись; 5) показувати всі важливі опції; 6) вміти спілкуватися з користувачем; 7) мати різні стилі для кнопок із різними типами дій; 8) бути привабливим; 9) надавати можливість персоналізації; 10) бути лояльним до помилок користувача; 11) говорити мовою користувача; 12) надавати оптимальну кількість варіантів вибору; 13) давати м’які підказки; 14) затемнювати фон під модальними вікнами; 15) мати короткі форми реєстрації; 16) мати найпростіші принципи заповнення полів; 17) надавати варіанти зручного керування.

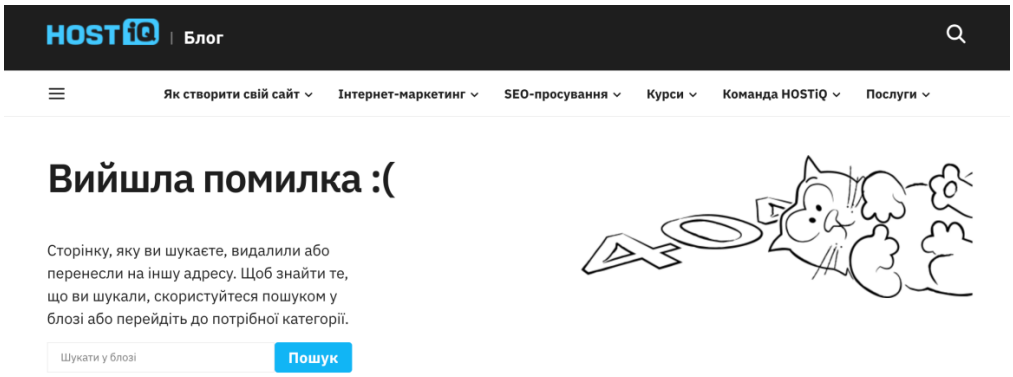
Бути інтуїтивно зрозумілим. Інтерфейс користувача — це те, що має бути максимально зрозумілим для більшості людей. Якщо людина, відкривши програму або зайшовши на сайт, не зрозуміє, як ним користуватися, то

понатискавши кілька секунд на різні кнопки навмання, розчарується та залишить ресурс. Швидше за все, назавжди.

Інтуїтивно зрозумілий інтерфейс — це той, в якому: 1) всі елементи побудовані за принципами елементарної логіки;

2) кнопкам надано зрозумілі позначення;

3) є допомога користувачеві, якщо він заблукав.



Бути передбачуваним. В інших ситуаціях передбачуваність може бути нудною та нецікавою характеристикою, але не щодо інтерфейсу. Користувач, глянувши на той чи інший елемент інтерфейсу повинен відразу зрозуміти, як він поведеться у разі взаємодії.

Бути мінімалістичним. Прагнучи розмістити в інтерфейсі якнайбільше категорій, меню, кнопок тощо, завдається тільки величезної шкоди. Надто захаращений інтерфейс – це велика перешкода для його розуміння користувачем.

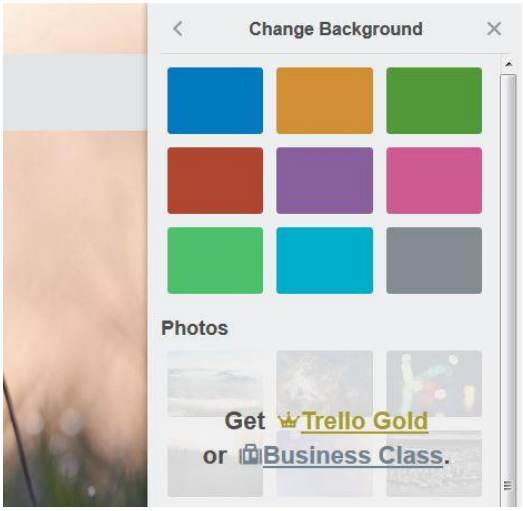
Все, що може бути описано однією фразою, не повинно бути завдано трьома. Зайві елементи та підкатегорії на головній сторінці теж ні до чого.

Швидко завантажуватись. Повільне завантаження інтерфейсу дратуватиме та відштовхуватиме користувача, викликаючи в ньому все більшу неприязнь до ресурсу. Слід переконатися, що швидкість завантаження є оптимальною для комфортного використання. Великою мірою це пов'язано із попереднім пунктом – адже що менше на сайті «важких» елементів, то швидше він завантажуватиметься.

Показувати всі важливі опції. Використовувати списки, що випадають, і меню краще тільки там, де цього уникнути неможливо. В інших випадках намагайтеся відразу показати користувачеві всі його можливості.

Вміти спілкуватися з користувачем. Йдеться про те, що користувач повинен розуміти, яка його дія зараз обробляється системою. Процес надсилання повідомлення повинен супроводжуватися виведенням на екран фрази «повідомлення надсилається», а закінчення цього процесу – «повідомлення надіслано».

Мати різні стилі для кнопок із різними типами дій. У будь-якому інтерфейсі кожна кнопка має своє призначення: перейти кудись, розкрити меню, відкрити в новому вікні, скачати тощо. Щоб не плутати користувачів, слід дотримуватись правил: 1) клікабельні та неклікабельні елементи повинні бути різні; 2) виділяйте кольором або стилем той розділ меню, в якому зараз знаходиться користувач.

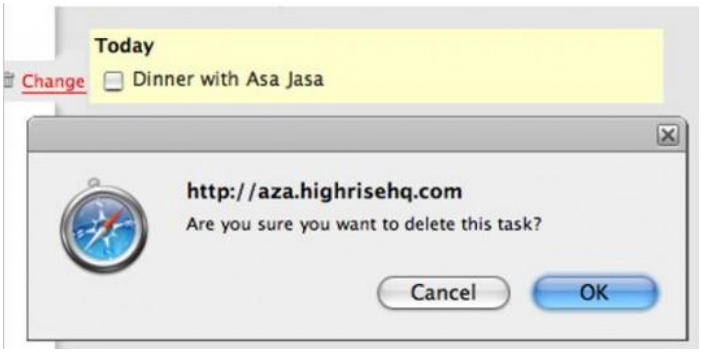


Бути привабливим. Функціональність та зручність – це добре, але є ще й елементарне «гарно/не гарно». Будь-якому користувачеві буде набагато приємніше мати справу з інтерфейсом, який, до всього іншого, ще й тішить око.

Надавати можливість персоналізації.

Персоналізація — це можливість налаштувати щось під себе. Найчастіше цю функцію можна зустріти у програмах, сервісах і додатках. Користувач може змінити, наприклад, колір шрифту, стиль іконок, фонове зображення, розмір текстових блоків так, як йому сподобається (вибираючи із запропонованих варіантів, звичайно).

Бути лояльним до помилок користувача. І все ж завжди будуть користувачі, які не зможуть відразу зрозуміти той чи інший момент при роботі з інтерфейсом. Вони будуть робити помилкові дії, і тут перед інтерфейсом постає завдання зробити так, щоб ці помилки могли бути швидко виправлені.

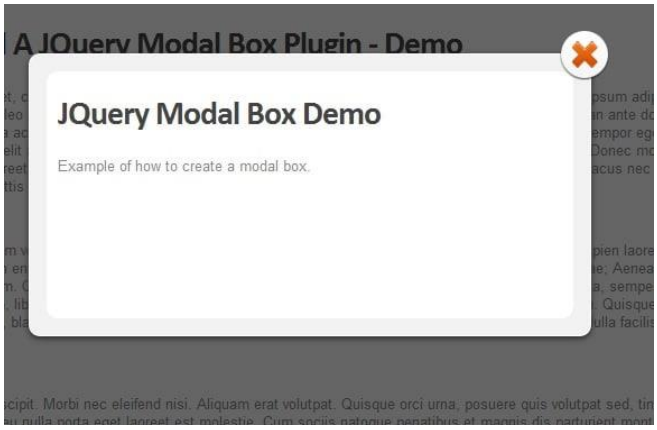


Говорити мовою користувача. Весь текст інтерфейсу, будь-які позначення, повинні бути створені під цільову аудиторію ресурсу.

Надавати оптимальну кількість варіантів вибору. Чим більше варіантів дій ви запропонуєте користувачеві, тим менше ймовірність, що він взагалі зробить хоч щось. Якщо варіантів дійсно багато, і скоротити їх не можна (наприклад, каталог товарів), то використовуйте функцію рекомендацій.

Давати м'які підказки. Підказки — це дуже добре. Вони допомагають користувачеві до кінця розібратися, в чому він не зміг розібратися сам.

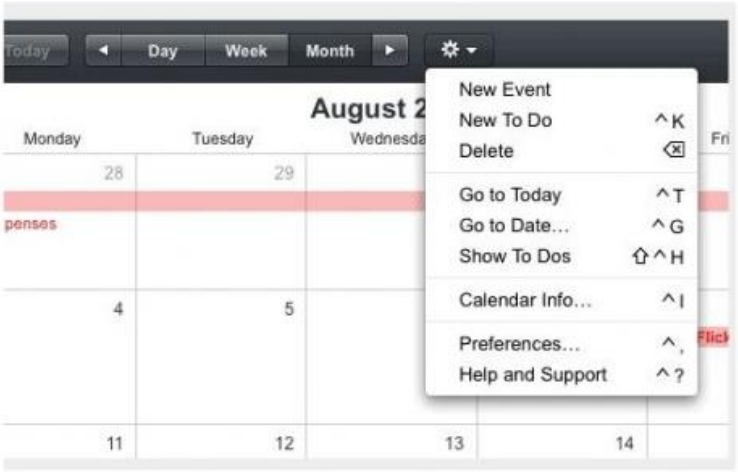
Затемнювати фон під модальними вікнами. Модальне вікно блокує роботу користувача, доки він не закриє це вікно або не зробить у ньому будь-яку дію. Таке вікно потрібно якось виділяти в загальній картині. Найкраще це зробити за допомогою затемнення фону, що знаходиться під ним.



Мати короткі форми реєстрації. Зазвичай люди намагаються уникати реєструватись там, куди вони не збираються заходити регулярно, або ще не впевнені в цьому. А часто навіть відмовляються від покупки в інтернет-магазині, якщо для цього обов'язкова реєстрація, та шукають свій товар на інших сайтах.

Мати найпростіші принципи заповнення полів. Майже кожен сайт пропонує користувачам щось на ньому заповнити. Крім реєстрації на сайті це може бути форма замовлення товару, прохання залишити свої контактні дані або пройти будь-яке опитування.

Надавати варіанти зручного керування. Якщо навігація сайтах в основному виконується мишкою, то в програмах і додатках найчастіше буває зручнішим керування з клавіатури. Доцільно давати користувачам вибирати той вид керування, який їм більше сподобається та показувати яка функція за якої комбінації клавіш включається.



2.2. Пошук переваг у діючих CRM-системах

Для повноцінного впровадження CRM-системи в існуючу бізнес-інфраструктуру важливо, щоб платформа не лише підтримувала базові функції управління клієнтами, але й дозволяла гнучке налаштування (кастомізацію) та інтеграцію з іншими сервісами -наприклад, електронною поштою, системами

обліку, календарями, платформами аналітики тощо [3].

У таблиці нижче представлено порівняння популярних CRM-рішень за наступними критеріями: 1) *Кастомізація* - можливість змінювати логіку роботи системи, структуру елементів, поля, статуси та бізнес-процеси без програмування або з мінімальним залученням розробника. 2) *Інтеграції* - кількість доступних сторонніх сервісів для підключення, підтримка API, доступ до внутрішніх автоматизацій та маркетплейсу додатків.

Відкритість API - наявність документації, REST API, підтримка вебхуків та SDK для розробників.

У таблиці 2.1 представлена порівняльна таблиця для аналізу можливостей кастомізації та інтеграції обраних для дослідження CRM платформ. Цей аналіз дозволяє обґрунтовано оцінити потенціал кожної системи для адаптації під потреби бізнесу або створення власних рішень на її основі - зокрема, розробки графічного інтерфейсу, що є предметом цієї дипломної роботи.

Таблиця 2.1 – Порівняльний аналіз можливостей кастомізації та інтеграції

Платформа	No-code налаштування	Кастомізація з кодом	Маркетплейс / API
Monday CRM	Поля, стани, автоматизації, табло	Немає потреби в коді (складні сценарії через API)	5000+ інтеграцій, відкритий REST API
HubSpot CRM	Поля, pipeline, email-шаблони	JavaScript у «CMS Hub»	1 000+ додатків в App Marketplace
Zoho CRM	Layout-редактор, Canvas, правила	Deluge Script, Functions	500+ інтеграцій, OpenAPI
Salesforce	Flow, Page Layouts, Lightning	Apex, LWC, Heroku	3 500+ компонентів AppExchange
Pipedrive	Поля, етапи, email-шаблони	Webhooks, простий API	~400 інтеграцій у Marketplace

Порівняльний аналіз CRM-систем за рівнем кастомізації та інтеграційних можливостей показує, що Monday CRM є одним із найбільш гнучких та відкритих до зовнішніх інтеграцій рішень. Платформа забезпечує широкий спектр налаштувань без потреби в програмуванні, включаючи модифікацію полів, статусів, форм подання інформації та автоматизацію робочих процесів за

допомогою графічного конструктора.

У той час як Salesforce пропонує ще вищий рівень кастомізації, він передбачає використання власної мови програмування (Apex) та значні технічні ресурси, що ускладнює її впровадження для невеликих або нетехнічних команд. HubSpot і Pipedrive мають обмежену кастомізацію, але натомість вирізняються простотою використання. Zoho CRM знаходиться посередині, пропонуючи помірний рівень налаштувань та доступ до великої кількості інтеграцій.

Таким чином, Monday CRM доцільно розглядати як оптимальну платформу для створення власного графічного інтерфейсу CRM-додатку. Вона забезпечує баланс між технічною доступністю, розширюваністю та простотою реалізації інтегрованих рішень через API та внутрішній маркетплейс додатків. Це повністю відповідає цілям дипломної роботи, орієнтованої на розробку кастомного графічного інтерфейсу в рамках Monday Work OS.

У таблиці 2.2 наведено порівняльний аналіз п’яти популярних CRM-систем - Monday CRM, HubSpot, Zoho CRM, Salesforce та Pipedrive - з урахуванням: 1) орієнтованості платформи на тип бізнесу (B2B, малі команди, великі корпорації тощо); 2) наявності безкоштовного тарифного плану; 3) стартової вартості платного доступу; 4) особливостей ліцензування на користувача.

Ці фактори впливають на вибір технологічної платформи для створення власного рішення, а також визначають потенційні обмеження для цільових користувачів майбутнього додатку.

Аналіз моделей ціноутворення та цільової аудиторії показує, що різні CRM-системи орієнтовані на різні сегменти бізнесу -як за розміром команди, так і за бюджетними можливостями. Наприклад, HubSpot CRM приваблює користувачів безкоштовним базовим функціоналом, проте значно обмежує доступ до розширених функцій без підписки на дорогі тарифи. Salesforce, навпаки, орієнтований переважно на великі корпоративні структури з відповідними ресурсами, а його тарифна модель малопридатна для малого та середнього бізнесу.

Monday CRM у цьому контексті вирізняється гнучким позиціонуванням:

пропонує як доступні тарифи для малих команд, так і потужні можливості масштабування для середнього та великого бізнесу. Це робить платформу зручною для використання на різних етапах розвитку компанії. Крім того, ціна за користувача в Monday CRM є конкурентною відносно обсягу функцій, що надаються вже у стартових планах.

Таблиця 2.2 – Порівняльний аналіз ціноутворення CRM-сервісів

Платформа	Цільові користувачі	Безкоштовний тариф	Базова ціна, $/кор./міс	Масштабованість
Monday CRM	Стартапи, креативні та IT-команди (5-500 осіб)	Тест 14 днів	10–24	До ~1 000 користувачів (Enterprise)
HubSpot CRM	Малий + середній бізнес, маркетологи	Так	0 (Free), → 18+ (Starter)	Без обмежень користувачів
Zoho CRM	SMB → великі компанії	Ні	14–45	Висока, модульні адд-они
Salesforce	Середній + корпоративний сегмент	Ні	25–300+	Практично необмежена
Pipedrive	Команди продажів 3-100 осіб	Ні	14–99	Обмежена (менше 500 користувачів)

У контексті кваліфікаційної роботи, що передбачає розробку власного додатку з графічним інтерфейсом на базі Monday Work OS, така тарифна модель і цільова гнучкість дозволяють орієнтуватися на ширший сегмент потенційних користувачів: від стартапів до стабільних середніх компаній. Це підвищує практичну цінність створеного рішення.

2.3. Використання інструментарію Monday work platform

Monday Work Platform – це хмарна платформа для управління робочими процесами, що надає користувачам можливість створювати адаптовані рішення для планування, комунікації та взаємодії в командах. В основі платформи лежить

концепція Work OS – операційної системи для роботи, яка дозволяє будувати індивідуальні цифрові робочі простори [6].

Платформа monday стала популярною завдяки своїй візуальній простоті, гнучкості в налаштуваннях та розвинутим інтеграціям із зовнішніми сервісами, зокрема Slack, Google Workspace, Microsoft Teams, Zoom, HubSpot, Salesforce, Zapier та багатьма іншими. Вона активно використовується компаніями для управління проєктами, завданнями, маркетинговими кампаніями, клієнтськими взаємодіями (CRM) та навіть внутрішніми HR-процесами.

Серед ключових характеристик платформи можна виділити: 1) базову структуру у вигляді дощок (boards), які можуть бути налаштовані під будь-який робочий процес – від простих списків до повноцінних CRM-систем; 2) гнучку систему статусів, стовпців і форм вводу, яку можна адаптувати під специфіку бізнесу; 3) автоматизацію дій за допомогою вбудованого конструктора сценаріїв (автоматичне призначення задач, надсилання повідомлень, зміна статусів тощо); 4) підтримку додатків і розширень (Apps), що дозволяє створювати власні віджети, інтеграції та автоматизації; 5) доступ до відкритого API, який надає розробникам змогу створювати кастомні рішення, підключати зовнішні сервіси або будувати повноцінні інтерфейси взаємодії.

Особливістю Monday Work Platform є її орієнтація на no-code та low-code підхід, що дозволяє бізнес-користувачам самостійно модифікувати робочі середовища без глибоких технічних знань (рис. 1.1.). Водночас, для складніших задач розробники можуть використовувати Monday Apps Framework на базі React.js, GraphQL або будь які інші веб технології [16].

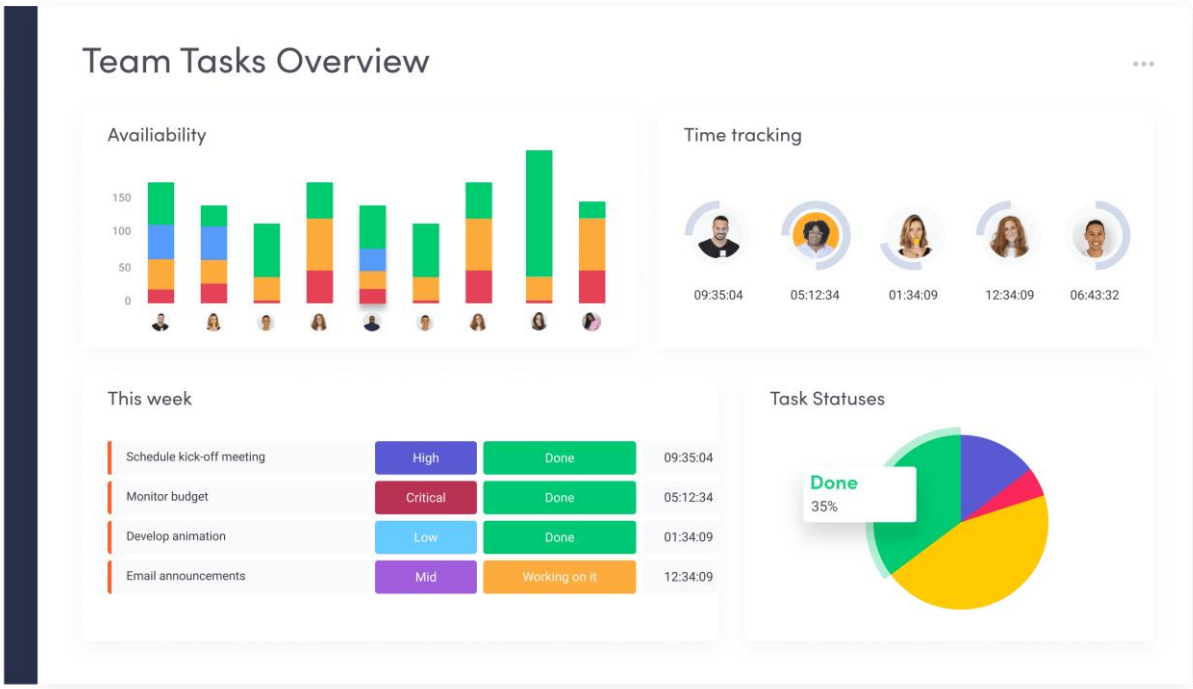


Рисунок 2.1 – Task management в Monday Work Platform

У контексті кваліфікаційної роботи саме Monday Work Platform обрана як база для розробки CRM-додатку, завдяки її технічній відкритості, підтримці кастомних елементів інтерфейсу та широким можливостям для інтеграції з сторонніми ресурсами.

Що робить Monday Work OS «платформою», а не просто CRM – роботу з їхнім GraphQL API та гнучкістю бордів (дощок). Нижче наведемо кусок коду, який: 1) через API Monday створює елемент (item) на дошці; 2) заповнює кастомні стовпці (наприклад, статус і дату); 3) демонструє, що ми працюємо не з «готовим CRM», а з універсальною робочою платформою, де все – борди, стовпці, статуси – можна ліпити під свій процес.

Приклад на Node.js (типовий бекенд для інтеграцій з Monday) [10]:

```
// Приклад: створення задачі на борді Monday Work OS через GraphQL API

import fetch from "node-fetch";

const MONDAY_API_URL = "https://api.monday.com/v2";
const MONDAY_API_TOKEN = process.env.MONDAY_API_TOKEN; // зберігаємо токен у змінній середовища
```

```
// ID борду, на якому працює застосунок (CRM-пайплайн)
const BOARD_ID = 123456789;

// Дані, які приходять з кастомного інтерфейсу (GUI)
const newLeadPayload = {
 name: "Новий відвідувач – ТОВ 'Городечний В.'",
 status: "New",
 nextFollowUpDate: "2025-11-20"
};

async function createLeadItemOnMonday(payload) {
 const query = `
  mutation ($boardId: Int!, $itemName: String!, $columnValues: JSON!) {
   create_item (
    board_id: $boardId,
    item_name: $itemName,
    column_values: $columnValues
   ) {
    id
    name
   }
  }
 `;

 // column_values – JSON-об'єкт, який відповідає кастомним колонкам на борді
 const columnValues = {
  status: { label: payload.status },          // статус-колонка
  date4: { date: payload.nextFollowUpDate }      // колонка типу "date" (наприклад, next follow-up)
  // можна додати інші поля: email, phone, текст, people, tags тощо
 };

 const variables = {
  boardId: BOARD_ID,
  itemName: payload.name,
  columnValues: JSON.stringify(columnValues)
 };

 const response = await fetch(MONDAY_API_URL, {
  method: "POST",
  headers: {
   "Content-Type": "application/json",
   Authorization: MONDAY_API_TOKEN
  },
  body: JSON.stringify({ query, variables })
 });

 const data = await response.json();
```

```
  if (data.errors) {
    console.error("Monday API error:", data.errors);
    throw new Error("Failed to create item in Monday");
  }

  console.log("Створено елемент на борді Monday:", data.data.create_item);
  return data.data.create_item;
}

// Виклик функції – усередині REST/GraphQL бекенду
createLeadItemOnMonday(newLeadPayload)
  .then(() => console.log("Лід успішно створено"))
  .catch(console.error);
```

У межах кваліфікаційної роботи саме цей продукт буде слугувати прикладом для розробки графічного інтерфейсу CRM-додатку, з урахуванням реальної структури та логіки обробки клієнтських даних.

РОЗДІЛ 3
МЕТОДИКА ПРОЕКТУВАННЯ ДОДАТКУ З ІНТУЇТИВНИМ ІНТЕРФЕЙСОМ КОРИСТУВАЧА

3.1 Інструментарій для створення додатку

Для розробки CRM-додатку в екосистемі Monday Work Platform було задіяно низку сучасних технологій, які забезпечили швидкий цикл розробки, можливість подальшого масштабування, стабільну інтеграцію з платформою Monday та гнучке керування даними. Обраний стек технологій поєднує інструменти для фронтенду, бекенду, баз даних, інфраструктури та тестування, що дало змогу реалізувати повноцінну інтеграцію стороннього сервісу з Monday CRM та підтримати подальший розвиток функціональності без критичних змін архітектури [10, 13, 15].

Для ефективної реалізації CRM-рішення важливо сформувати стабільне, передбачуване та зручне середовище розробки, яке підтримує кросплатформені інструменти, командну взаємодію та автоматизацію рутинних операцій. У цьому проєкті було використано операційне середовище з підтримкою Linux-функціоналу всередині Windows, а також систему контролю версій, що дозволила зберігати історію змін, організувати паралельну розробку в гілках та забезпечити безперервність процесу розробки.

WSL2 (*Windows Subsystem for Linux 2*) – забезпечив повноцінне Linux-середовище всередині Windows-системи, що дало можливість використовувати переваги Linux-командного рядка, пакетних менеджерів та скриптів автоматизації, не змінюючи основну ОС розробника [4].

Git – основний інструмент для контролю версій та організації командної роботи. Застосовувався для відстеження змін у кодовій базі, управління гілками, реалізації feature- та hotfix-флоу, а також синхронізації роботи кількох розробників. Завдяки інтеграції з GitHub було

налаштовано зручний pull request-цикл для перегляду, рев'ю та валідації змін перед потраплянням у основну гілку проєкту [5].

Під час розробки CRM-додатку виникла потреба створити компактний графічний інтерфейс, який дозволяв користувачу взаємодіяти з додатком та передавати свій персональний *Crunchbase API key*. Для цього були застосовані сучасні frontend-технології, що забезпечили швидку розробку, зручну стилізацію, реактивність компонентів та ефективне управління станом додатку. Такий підхід поєднує простоту реалізації з гнучкістю, що дозволило інтегрувати UI у загальну архітектуру додатку без перевантаження зайвим функціоналом.

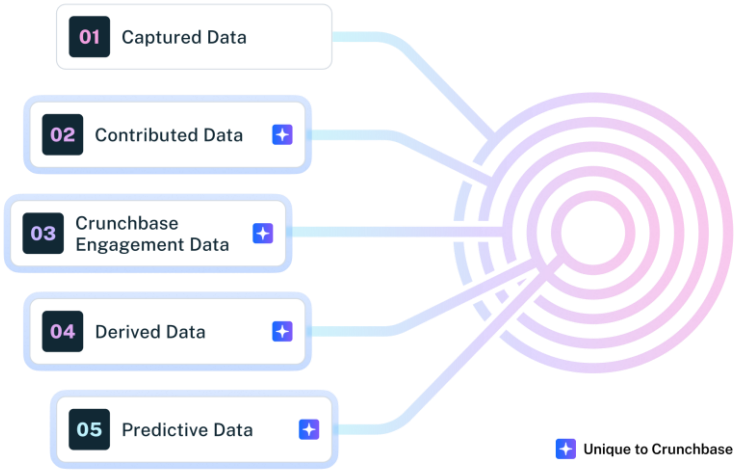


Рисунок 3.1 – Дані та прогнози в Crunchbase API

React обрано як основну бібліотеку для побудови компонентного, модульного інтерфейсу, що дозволило швидко створювати інтерактивні UI-компоненти, зручні для масштабування та повторного використання [617].

Vite забезпечив миттєву перезбірку, гаряче оновлення модулів (HMR) та швидкий старт проєкту.

Sass застосовано для гнучкої стилізації, підтримки змінних, вкладеності та міксинів, що полегшило масштабування стилів великого UI [17].

MobX забезпечив реактивне керування станом, автоматизуючи оновлення інтерфейсу при зміні даних, що підвищило продуктивність і спростило код [13].

Серверна частина додатку реалізована з акцентом на продуктивність,

масштабованість і сувору типізацію, що зменшило кількість помилок та спростило підтримку. Архітектура бекенду побудована на *JavaScript*/*TypeScript* із легким *HTTP*-фреймворком та інструментами для роботи з *SQL*, що забезпечило швидке створення надійного *API* для взаємодії між усіма компонентами системи [18]. Axios використано для *HTTP*-запитів між *frontend* і *backend* та взаємодії з *Crunchbase API*. *Node.js* у поєднанні з *TypeScript* дозволили створити надійну бекенд-архітектуру з чіткою типізацією [14]. *Fastify* обрано як швидкий і легкий фреймворк для *RESTful API* з плагінною архітектурою, що спрощує підключення логів, валідації та *CORS*. *Knex*.js спростив побудову *SQL*-запитів, управління міграціями та підвищив безпечність роботи з різними СУБД.

Для розробки та тестування використовувалися сучасні інструменти: *Visual Studio Code* як основне середовище з розширеннями для форматування, автодоповнення та інтеграції з *Git*; *Postman* для ручного тестування API-запитів, перевірки авторизації та побудови колекцій запитів [16].

Додаток розгорнуто в екосистемі monday.com із використанням її внутрішніх інструментів для стабільної роботи та інтеграції: *Monday Code* забезпечив хостинг серверної частини всередині платформи, підвищену безпеку та інтеграцію з внутрішнім API; *Monday Queue* реалізував оновлення дошечки порціями, обходячи обмеження ранінг-тайму; *Monday Storage* зберігає дані про структуру дошечки, відповідність полів *Crunchbase* і колонок; *Integration for Sentence Builder* дозволив безпосередньо інтегрувати функціонал додатку у середовище *Monday*.

Для зберігання даних про обробку айтемів та інтеграцію з *Crunchbase* використано *PostgreSQL* як реляційну СУБД, *Docker* для

локального розгортання, *AWS RDS* для тестового середовища та *Neon* для продакшн-бази з гнучким масштабуванням і безперервним збереженням змін [15].

Завдяки такому технологічному стеку CRM-додаток забезпечує 1) зручний, користувацько-орієнтований інтерфейс; 2) пряме інтегрування з Mo*nday Work OS* через її інструменти та API; 3) масштабовану архітектуру для розширення функціоналу; 4) стійкість і надійність при роботі з великими обсягами даних; 5) простоту обслуговування, оновлення та розгортання нових версій у production-середовищі.

2.3 Виокремлення мети та технічного завдання

Для досягнення основної цілі – побудови інтуїтивного інтерфейсу користувача в CRM-системі з автоматичним отриманням та синхронізацією даних із сервісу Crunchbase – було сформовано технічне завдання, що включає функціональні та нефункціональні вимоги, технологічні обмеження, а також типові сценарії використання системи.

1. Загальна мета: Головна мета розробки полягає у створенні додатку, який автоматизує оновлення інформації про компанії на дошці Monday.com через інтеграцію з Crunchbase API. Такий софт має зменшити ручну роботу та спростити аналітику для користувачів, що відстежують зміни у фінансовому, організаційному або ринковому стані вибраних компаній.

2. Функціональні вимоги: Додаток повинен реалізовувати такі основні функції: 1) авторизація користувача шляхом введення персонального Crunchbase API key; 2) перевірка коректності ключа під час початкового налаштування; 3) динамічний мапінг полів, отриманих із Crunchbase, до відповідних колонок на бордах Monday; 4) обробка подій, зокрема створення нового айтема або зміни назви чи вебсайту компанії, з подальшим оновленням даних; 5) автоматична щоденна синхронізація всіх записів із Crunchbase у фоновому режимі; 6) обробка помилок (наприклад, некоректний API key або відсутність компанії у Crunchbase)

з відображенням інформативних повідомлень для користувача; 7) робота в умовах обмежень на кількість запитів до Crunchbase API за рахунок реалізації черги з поступовою обробкою записів.

3. Нефункціональні вимоги: Для забезпечення стабільної роботи система має відповідати таким критеріям: 1) висока надійність та відмовостійкість при типових збоях мережі або зовнішніх сервісів; 2) підтримка масштабування до приблизно 4000 айтемів на один акаунт без критичної деградації продуктивності; 3) час обробки черги не повинен перевищувати 300 секунд, що відповідає обмеженням на виконання функцій у середовищі Monday Code.

4. Технологічні обмеження: Реалізація додатку відбувається з урахуванням таких умов: 1) використання Monday Code як середовища виконання серверної логіки; 2) застосування Monday Queue для контрольованої обробки подій і недопущення перевищення лімітів зовнішніх та внутрішніх API; 3) інтеграція з Crunchbase API через бібліотеку Axios у середовищі Node.js; 4) збереження службових даних у PostgreSQL із поділом середовищ на development, testing та production (Docker, AWS RDS, Neon); 5) фронтенд-застосунок побудований на React, MobX і Vite та використовується виключно для безпечної передачі Crunchbase API key та базових налаштувань.

5. Сценарії використання: Додаток підтримує такі типові сценарії взаємодії: 1) користувач додає або оновлює інформацію про компанію в Monday, після чого система автоматично виконує запит до Crunchbase і оновлює відповідні поля на дошці; 2) виконується щоденна фонова синхронізація всіх компаній із зовнішнім сервісом для підтримання актуальності даних; 3) у разі виникнення помилки (наприклад, проблеми з API або пошуком компанії) відповідний опис записується в спеціальну колонку на дошці, а користувач отримує повідомлення про проблему.

3.3. Реалізація структури додатку та створення макету

На етапі проєктування структури додатку основний акцент було зроблено на архітектурі взаємодії між його компонентами, побудові внутрішніх процесів синхронізації даних та загальній подієвій логіці в межах платформи Monday.com. З урахуванням обмежень середовища виконання Monday Code та вимог до масштабованості, додаток спроєктовано як набір взаємопов'язаних фаз обробки, що працюють на основі черги повідомлень (Monday Queue) та поетапного оновлення даних.

Для формалізації структури та поведінки системи було розроблено низку UML-діаграм послідовностей (sequence diagrams), які описують ключові сценарії взаємодії компонентів додатку [12]:

– Recipe Install + Snapshot Phase – початкове встановлення інтеграційного рецепту та перше зчитування вмісту дошки (snapshot);

– Sync Phase – основна фаза збагачення даних за допомогою Crunchbase API;

– Daily Sync – щоденне оновлення інформації про айтеми на борді;

– Finalize Phase – завершальна фаза синхронізації, яка перевіряє, чи всі необхідні рядки були оновлені, і, у разі успіху, робить відповідні позначки про завершення циклу оновлення.

Надалі кожна з діаграм розглядається окремо з поясненням її функціонального призначення та ролі в загальній логіці роботи системи.

На рис. 3.2 наведено діаграму послідовностей, що описує логіку, спроєктовану для початкового встановлення рецепту та першого зчитування даних з дошки (snapshot). Дана діаграма відображає повний ланцюжок дій, який виконується під час першої інсталяції інтеграційного рецепту користувачем у Monday Work Platform – від взаємодії з UI до запуску першої фази синхронізації (Snapshot Phase), яка фіксує поточний стан дошки для подальшої обробки.

На рис. 3.3 показано діаграму, що описує щоденне оновлення даних айтемів на борді.

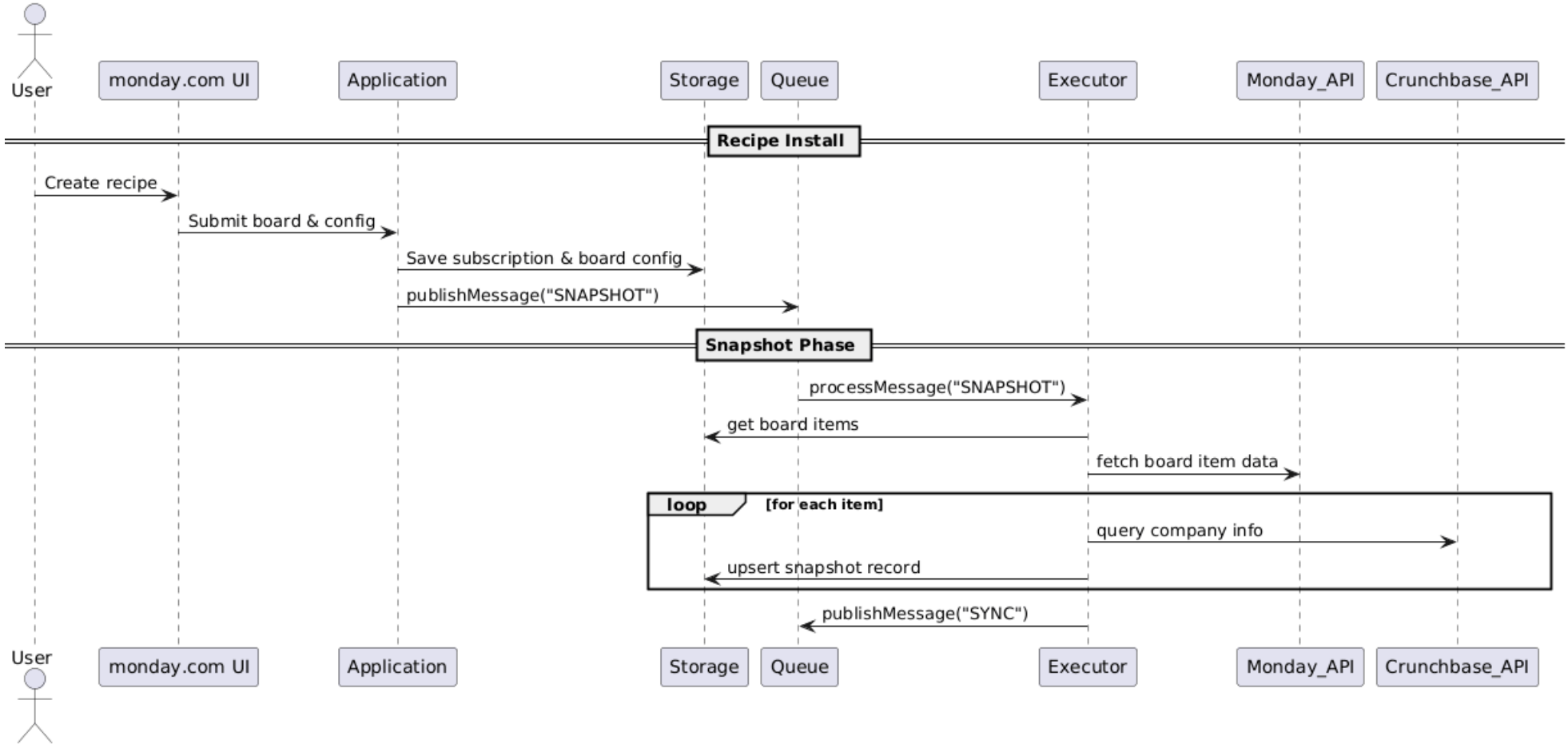


Рисунок 3.2 – Recipe Install + Snapshot Phase Sequence Diagram

Вона моделює процес автоматичної ініціалізації синхронізації, який виконується раз на 24 години для кожного акаунта з активованим інтеграційним рецептом. Основне завдання цього сценарію – забезпечити регулярне збагачення CRM-даних, навіть за відсутності ручних змін з боку користувача.

Процес включає такі базові кроки:

– Запуск планувальника – синхронізація ініціюється зовнішнім тригером (наприклад, AWS Lambda або іншим scheduler-сервісом), який надсилає POST-запит на спеціальний endpoint бекенду додатку;

– Визначення цільових підписок. Додаток відбирає всі активні підписки (Subscriptions), для яких за останні 24 години не виконувалась синхронізація, і для кожної з них формує чергове повідомлення типу SNAPSHOT;

– Обробка SNAPSHOT-повідомлення. QueueProcessor зчитує повідомлення з черги, ідентифікує boardId та accountId, отримує актуальний список айтемів із monday.com і порівнює його з попередньо збереженим snapshot;

– Оновлення структури Snapshot. Фіксуються зміни (нові, змінені або видалені айтеми), після чого відповідні записи у базі оновлюються. Додатково оновлюється службова інформація про етап виконання та поточний статус обробки (SyncState);

– Публікація SYNC-повідомлень. Якщо в результаті оновлення snapshot виявлено айтеми, що потребують збагачення даними, вони розбиваються на «порції», і для кожної порції у чергу додається окреме повідомлення типу SYNC;

– Контроль обсягу та часу. Під час обробки враховуються обмеження у 4000 айтемів на акаунт та максимальний час виконання (до 300 секунд). Якщо ліміти перевищено, обробка решти айтемів переноситься на наступний цикл;

– Оновлення Subscription. Після завершення або вимушеного переривання циклу обробки оновлюється поле lastCycleCompletedAt, яке використовується як мітка часу останньої успішної або часткової синхронізації.

Така щоденна процедура гарантує, що навіть без додаткових дій з боку користувача дані на борді залишаються актуальними, а інтеграція з Crunchbase стабільно виконується у фоновому режимі.

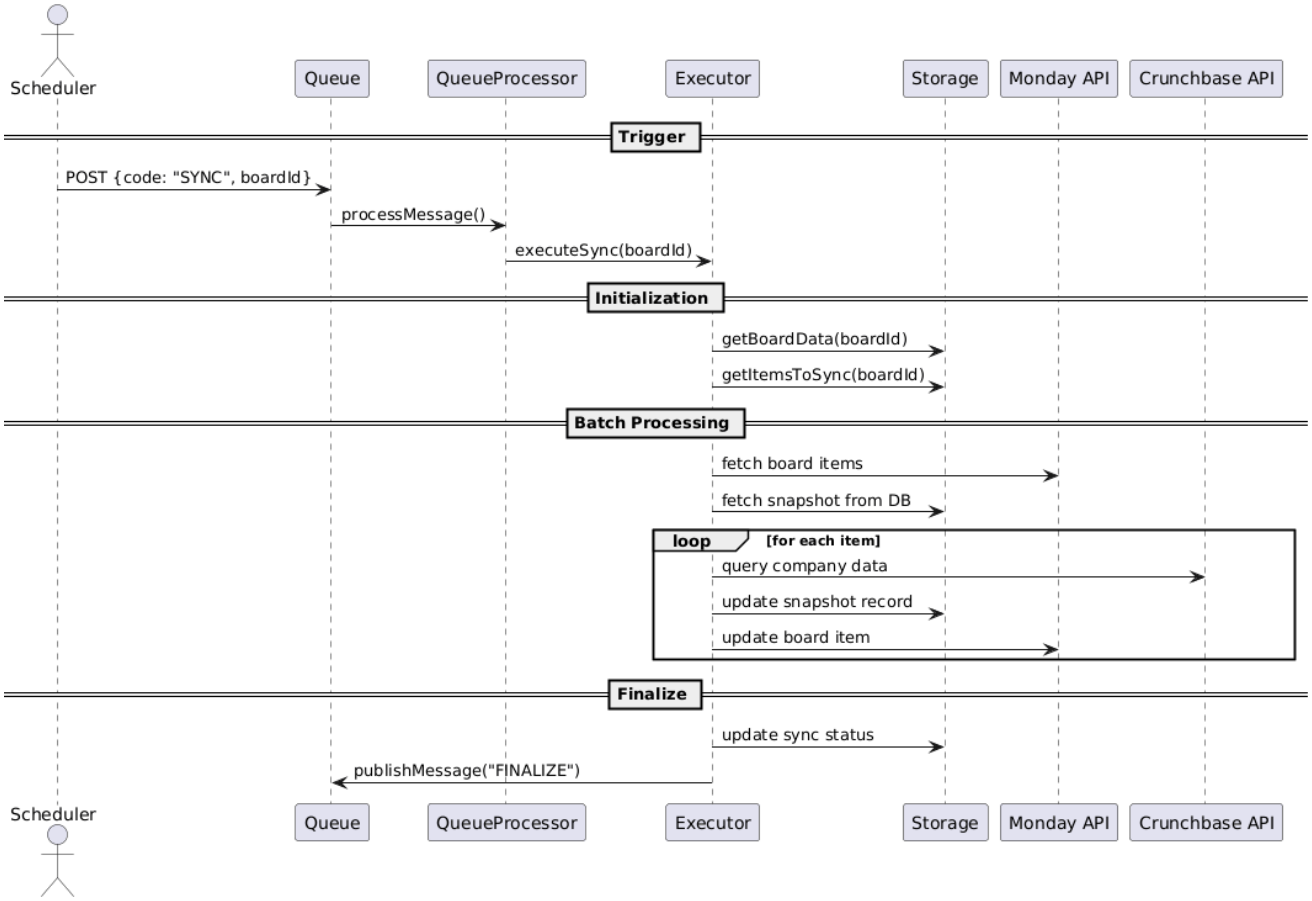


Рисунок 3.3 – Daily Sync Sequence Diagram

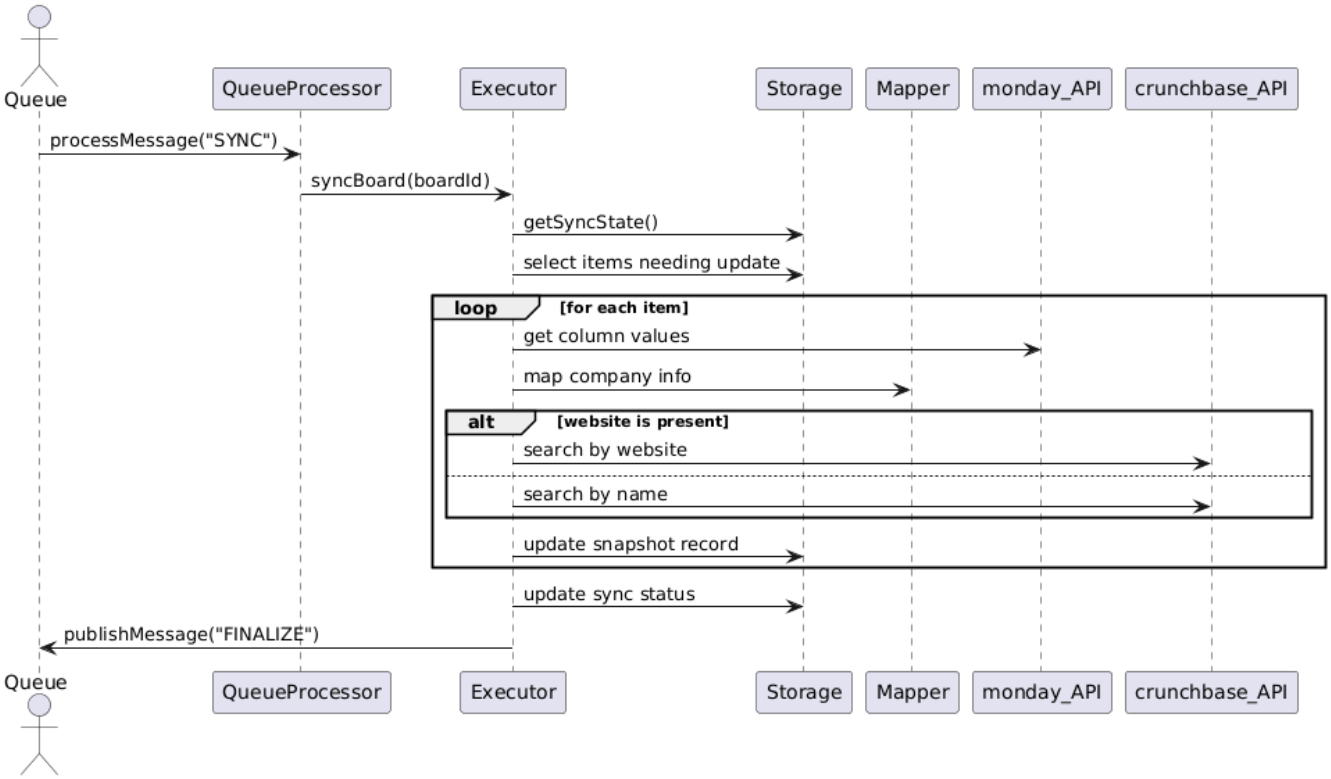


Рисунок 3.4 – Sync Phase Sequence Diagram

На рис. 3.4 наведено діаграму, що відображає основну фазу збагачення даних із використанням Crunchbase API. Вона описує один із ключових сценаріїв – фазу синхронізації (SYNC Phase), яка входить до загального циклу оновлення даних про компанії та відповідає за оновлення item-ів на дошці користувача актуальною інформацією з Crunchbase. Запуск цієї фази здійснюється через чергу (Monday Queue), яка генерує та надсилає повідомлення типу SYNC.

На рис. 3.5 зображено діаграму, що описує фінальну фазу синхронізації, яка перевіряє, чи всі необхідні рядки були успішно оновлені, і, у разі коректного завершення процесу, встановлює відповідні мітки про закриття поточного циклу оновлення.

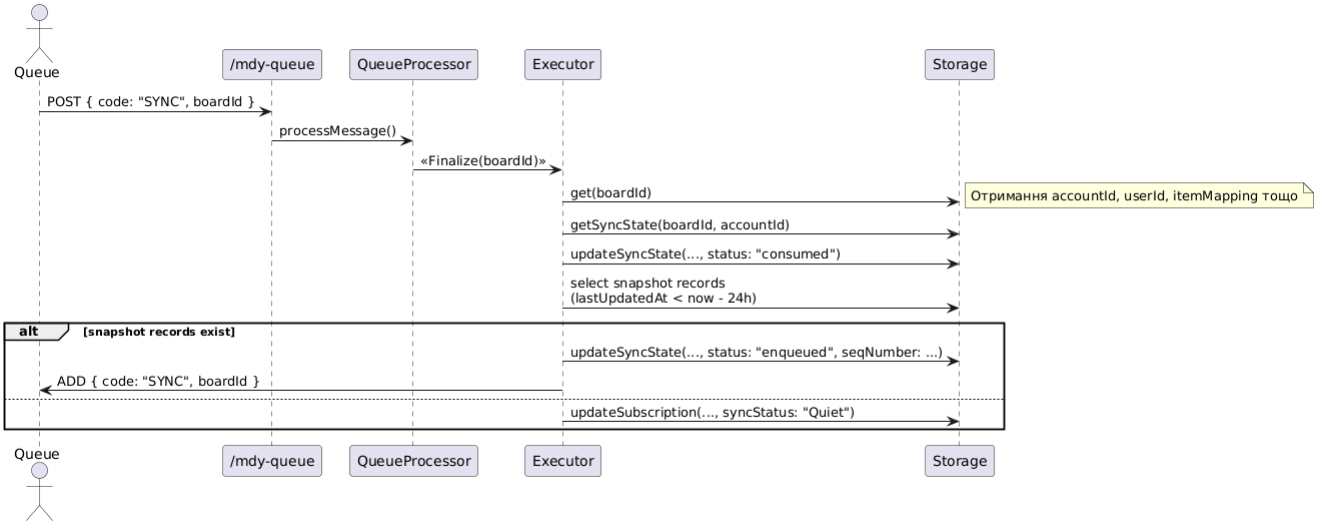


Рисунок 3.5 – Finalize Phase Sequence Diagram

Ця діаграма відображає логіку роботи завершальної стадії синку, яка запускається після того, як для конкретної дошки були оброблені всі заплановані повідомлення типу SYNC. Ключове призначення цієї фази – зафіксувати факт завершення циклу, проаналізувати наявність неоновлених або пропущених айтемів і, за потреби, ініціювати новий етап обробки або додатковий прохід по даних.

3.4. Розробка функціональності додатку

Розробка функціональності додатку побудована на модульній архітектурі з чітким розподілом відповідальності між окремими компонентами. Основний бізнес-код написаний мовою TypeScript із використанням фреймворку Fastify для серверної частини та бібліотеки Knex для роботи з базою даних PostgreSQL. Взаємодія з monday.com та Crunchbase API реалізована через HTTP-запити з використанням клієнта Axios [13-19].

Кожен ключовий процес – встановлення рецепту, формування snapshot, синхронізація даних, обробка черги – винесений в окремий клас або сервіс. Це спрощує масштабування, рефакторинг і подальший супровід проєкту. У цьому підрозділі описуються основні програмні модулі, з яких складається система, а також наведено фрагменти найбільш показового функціоналу.

Рисунок 3.6 – Структура файлів проекту

На рис. 3.6 наведено скріншот структури файлів проєкту. У кореневому каталозі розташований конфігураційний файл app.ts, який відповідає за ініціалізацію серверу Fastify, реєстрацію плагінів, маршрутів та middleware-компонентів. Така організація коду дає змогу чітко розділити логіку на рівні API, синхронізації, шару доступу до даних та інтеграцій із зовнішніми системами. Це позитивно впливає як на масштабованість рішення, так і на зручність його тестування та підтримки.

Каталог public/ містить статичні ресурси, що віддаються бекендом (Fastify) у відповідь на запити користувача. Такий підхід дозволяє обійтися без окремого фронтенд-хостингу й спрощує деплой. Хоча основна логіка додатку зосереджена на серверній частині, для зручності взаємодії з користувачем було реалізовано невеликий клієнтський інтерфейс, єдина задача якого – надати користувачу можливість безпечно ввести та передати свій Crunchbase API key.

Окрему увагу приділено реалізації класу “ExecutorFactory” (рис. 3.7). Його призначення – організація обробки вхідних повідомлень, що надходять до додатку через чергу “monday queue”. Клас відповідає за динамічне створення об’єктів-виконавців (executors) залежно від типу задачі, вказаного в повідомленні.

```
export class ExecutorFactory {
  private dbStateManager: DbStateManager;
  private queueManager: QueueManager;

  constructor() {
    this.dbStateManager = new DbStateManager();
    this.queueManager = new QueueManager(queueService);
  }

  /**
   * Creates an executor based on the provided code.
   * @param code The message code that determines which executor to create
   * @returns An instance of the appropriate executor
   * @throws Error if the code is not recognized
   */
  createExecutor(code: SyncStateCode): any {
    switch (code) {
      case SyncStateCode.SNAPSHOT:
        return new SnapshotExecutor(this.dbStateManager, this.queueManager, EXECUTOR_TIME_LIMIT_MS);
      case SyncStateCode.SYNC:
        return new SyncExecutor(this.dbStateManager, this.queueManager, EXECUTOR_TIME_LIMIT_MS);
      case SyncStateCode.FINALIZE:
        return new FinalizeExecutor(this.dbStateManager, this.queueManager, EXECUTOR_TIME_LIMIT_MS);
      default:
        logger.error(`Unknown executor code`, { code });
        throw new Error(`Unknown executor code: ${code}`);
    }
  }
}
```

Рисунок 3.7 – Реалізація класу ExecutorFactory

Такий підхід ґрунтується на патерні Factory Method і дозволяє інкапсулювати логіку створення обробників, відокремлюючи ініціалізацію конкретних executor-ів (SnapshotExecutor, SyncExecutor, FinalizeExecutor) від решти системи. Завдяки цьому додаток легко розширюється – для додавання

нової фази або обробника достатньо оновити відповідний switch, не змінюючи базову логіку роботи черги.

DbStateManager інкапсулює логіку доступу до бази даних під час виконання задачі: читає й оновлює стан синхронізації, інформацію про item-и та інші службові дані.

QueueManager відповідає за постановку нових задач у чергу після завершення поточної (наприклад, додавання подій SYNC або FINALIZE після обробки SNAPSHOT). Константа EXECUTOR_TIME_LIMIT_MS передається в кожен executor і визначає максимальний час виконання задачі в мілісекундах, щоб дотримуватися обмежень середовища Monday Code щодо тривалості виконання функцій.

РОЗДІЛ 4
ВИКОРИСТАННЯ ДОДАТКУ ТА КОМПЛЕКСНЕ ТЕСТУВАННЯ

4.1. Тестування та валідація коректності роботи додатку

Після реалізації основної логіки додатку було проведено комплексне тестування та виконано низку оптимізацій, спрямованих на підвищення стабільності, масштабованості та відповідності обмеженням середовища виконання Monday Code. Для ручної перевірки API-запитів активно застосовувався Postman, а також були створені тестові дошки в monday.com з різними сценаріями використання (заповнені та порожні поля, некоректні API-ключі, відсутність компаній у Crunchbase тощо).

Для валідації коректності роботи всіх фаз циклу збагачення борди було розроблено власний симулятор черги, який емітує поведінку Monday Queue та дає змогу запускати обробку повідомлень у контрольованому середовищі. Ключовий фрагмент відповідного коду наведено на рис. 4.1.

```
// Enqueue endpoint
app.post('/enqueue', async (req, res) => {
    const message = req.body;
    console.log('\nReceived message:', JSON.stringify(message, null, 2));

    const shouldDispatch = process.env.AUTO_DISPATCH === 'true' ? true : await promptForDispatch(message);

    res.json({ status: 'dispatched', message });

    if (shouldDispatch) {
        await dispatchMessage(message);

    } else {
        console.log('Message dispatch cancelled');
    }
});
```

Рисунок 4.1 – Ключовий фрагмент коду власного симулятору черги

Цей фрагмент демонструє реалізацію кастомного симулятора черги, який використовувався для інтеграційного тестування роботи додатку. Завдяки такому підходу розробник отримав можливість вручну або автоматизовано керувати надсиланням повідомлень, що в реальному середовищі формуються чергою

monday.com. Це дало змогу поетапно перевірити коректність виконання кожної з фаз обробки – SNAPSHOT, SYNC та FINALIZE – у контрольованих умовах, що суттєво спростило виявлення та усунення помилок на ранніх стадіях розробки.

```
it('should not get subscriptions with sync errors', async () => {
  const twoDaysAgo = new Date(Date.now() - 2 * 24 * 60 * 60 * 1000);

  // Create test subscriptions
  const subscriptions = [
    // Account 1: QUIET with error
    {
      id: '1',
      accountId: 7860841,
      boardId: 1,
      userId: 1,
      websiteColumnId: 'col1',
      webhookUrl: 'http://test1.com',
      lastCycleCompletedAt: twoDaysAgo,
      syncStatus: SyncStatus.QUIET,
      syncError: 'API error occurred',
    },
    // Account 2: QUIET without error
    {
      id: '2',
      accountId: 7860842,
      boardId: 2,
      userId: 2,
      websiteColumnId: 'col2',
      webhookUrl: 'http://test2.com',
      lastCycleCompletedAt: twoDaysAgo,
      syncStatus: SyncStatus.QUIET,
      syncError: null,
    },
  ];

  await db(TABLE_NAMES.Subscriptions).insert(subscriptions);

  // Test with limit 2
  const result = await mondaySubscriptionsDao.getSubscriptionsToSync(2);

  // Should only get Account 2 (no error) and not Account 1 (has error)
  expect(result).toHaveLength(1);
  expect(+result[0].accountId).toBe(7860842);
});
});
```

Рисунок 4.2 – Модульний тест одного з методів Subscriptions DAO

Окремий акцент було зроблено на логіці вибору підписок (subscriptions) для запуску циклу синхронізації. Зокрема, реалізовано модульний тест, який перевіряє, що система коректно відфільтровує записи з помилками або некоректним станом (рис. 4.2), гарантуючи, що до обробки потрапляють лише валідні підписки.

4.2 Практичне використання додатку з інтуїтивним інтерфейсом користувача

Розроблений додаток дає змогу автоматично збагачувати CRM-дані на дошці платформи monday.com актуальною інформацією з Crunchbase, що є особливо корисним для бізнес-аналітики, моніторингу партнерів та відстеження змін у клієнтських компаніях.

Після встановлення додатку та початкового налаштування інтеграції користувач отримує можливість сконфігурувати так званий «рецепт» – набір правил, за якими система буде автоматично заповнювати колонки на дошці. Для цього необхідно:

– вказати колонку, в якій зберігається веб-сайт компанії;

– обрати, які саме поля з Crunchbase (наприклад, локація, дата залучення інвестицій тощо) мають відображатися на дошці;

– визначити колонку, в яку записуватимуться повідомлення про помилки або технічні статуси.

Інтеграція спрацьовує у таких випадках:

– під час створення нового рядка (item) на дошці;

– при зміні назви компанії або оновленні її веб-сайту в існуючому рядку;

– один раз на добу в межах щоденного синхронізаційного циклу, мета якого – актуалізація даних без участі користувача.

Під час початкового налаштування користувач має ввести валідний Crunchbase API key. Додаток виконує перевірку його коректності та рівня доступу. Якщо ключ є тестовим (trial) або базовим (basic) і не підтримує необхідний функціонал, система відобразить повідомлення про помилку та не дозволить завершити конфігурацію. Заміна ключа після первинної інсталяції рецепту передбачена лише шляхом повного перевстановлення додатку.

Нижче описано повний цикл взаємодії користувача з додатком Crunchbase CRM Enrichment у середовищі monday.com – від стартової інсталяції до фактичного збагачення дошки бізнес-даними:

– користувач обирає робочий простір і конкретну дошку, на якій планується активувати інтеграцію;

– із переліку доступних інтеграцій користувач вибирає шаблон, що дозволяє шукати компанії в Crunchbase та автоматично переносити їхні атрибути на дошку (рис. 4.3);

– система запитує необхідні permissions на доступ до даних обраних бордів, веб-хуків та можливість змінювати значення колонок (рис. 4.4), після чого користувач підтверджує надані права;

– далі користувач вводить особистий API-ключ Crunchbase; якщо ключ недійсний, прострочений або має пробний тариф, система блокує завершення налаштування та показує відповідне повідомлення (рис. 4.5);

– на наступному кроці користувач налаштовує мапінг між полями Crunchbase (наприклад, City, Acquired Date тощо) і відповідними колонками на дошці monday.com (рис. 4.6);

– після завершення конфігурації система автоматично запускає початкову синхронізацію, під час якої існуючі рядки на дошці збагачуються актуальними даними з Crunchbase.

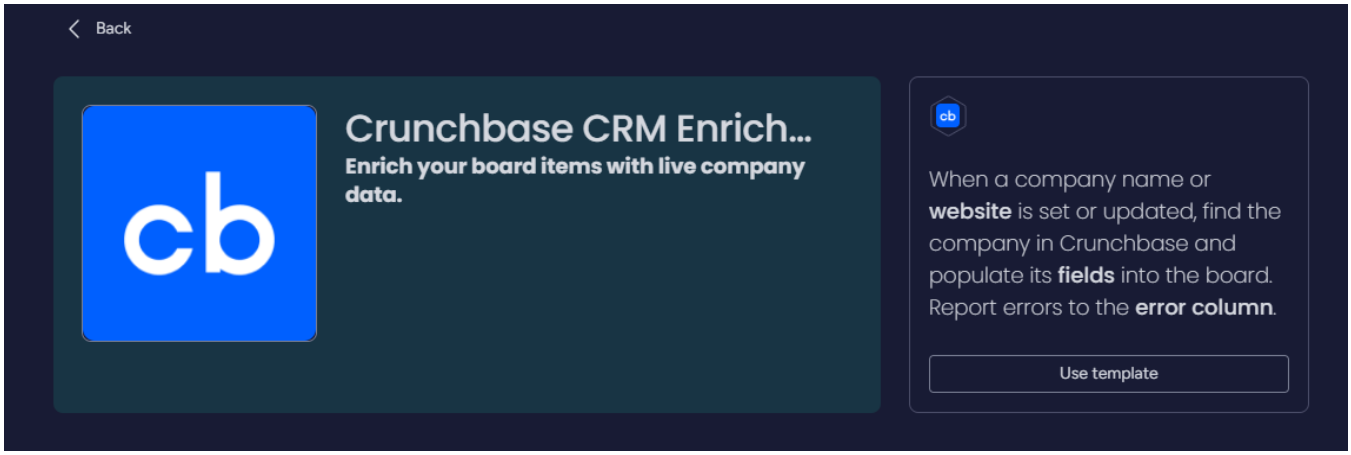


Рисунок 4.3 – Вибір шаблону автоматизації

Усі знайдені компанії автоматично заповнюються на дошці актуальними даними (рис. 4.7).

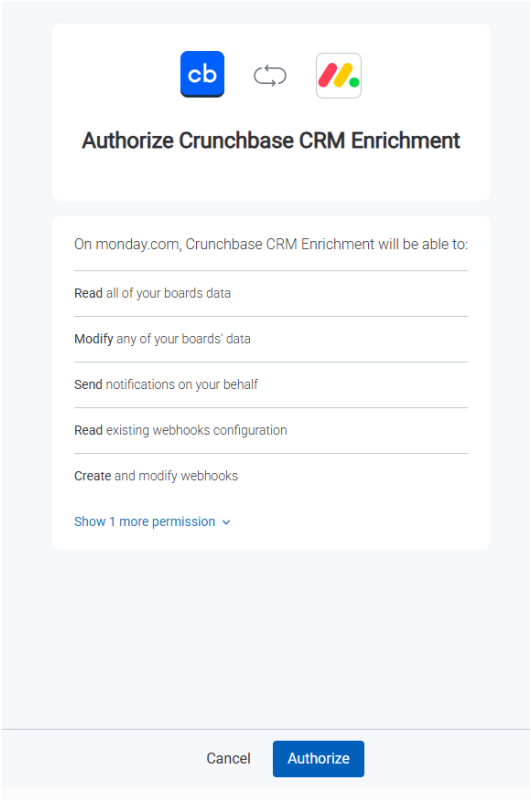


Рисунок 4.4 – Підтвердження доступу до необхідних даних

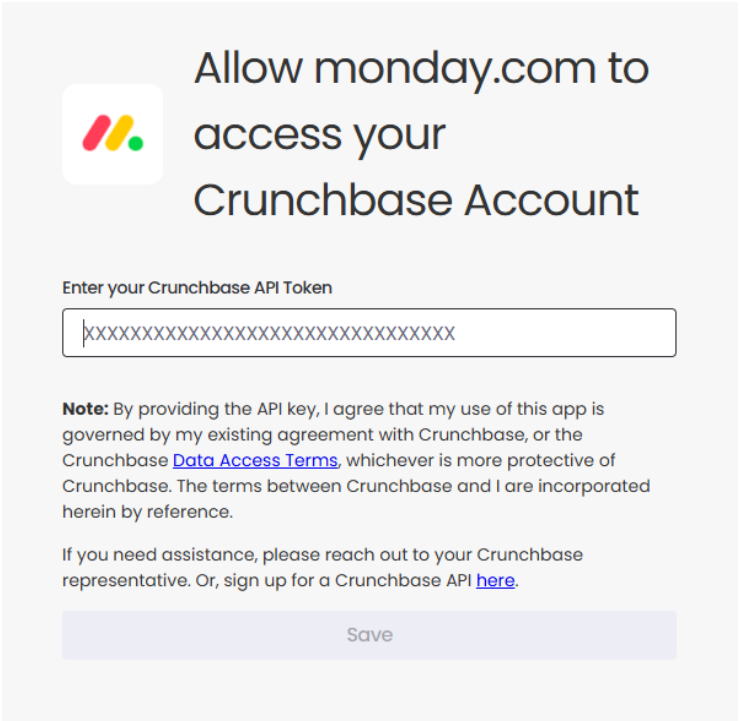


Рисунок 4.5 – Сторінка для передачі особистого ключа до Crunchbase API

Рисунок 4.6 – Задання відповідності між полями Crunchbase та відповідними колонками на дошці

Як видно з рисунку, після заповнення полів, обраних користувачем під час налаштування інтеграції, додаток самостійно підтягує та підставляє значення, отримані з Crunchbase API.

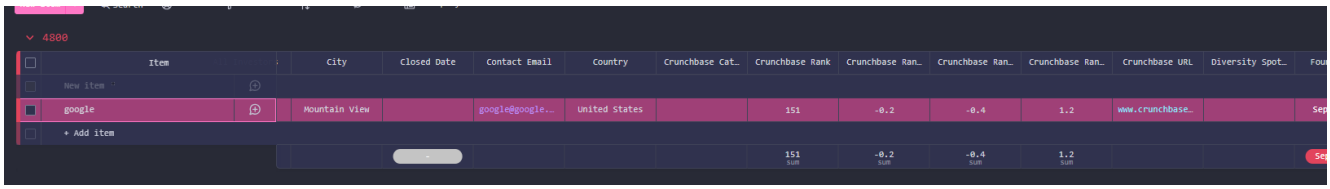


Рисунок 4.7 – Результат збагачення рядку на дошці

Зручність інтерфейсу, описаного в роботі, полягає насамперед у його автоматизованості та наочності. Користувачеві не потрібно вручну збирати або перевіряти інформацію про компанії – додаток самостійно отримує актуальні дані з Crunchbase і підставляє їх у відповідні колонки на дошці. Це скорочує час на рутинні операції та мінімізує людські помилки.

Код React-компонента з використанням MobX для стану, який реалізує базовий UI наведений в Додатку А.

Інтерфейс інтуїтивний, бо налаштування інтеграції виконуються через логічну послідовність кроків: вибір робочого простору і дошки, підтвердження доступу, введення API-ключа, зіставлення полів між Crunchbase і дошкою. Користувач бачить результат відразу після синхронізації, включно з усіма ключовими атрибутами компанії (місто, країна, email, категорії, рейтинг, URL), що дозволяє швидко орієнтуватися в бізнес-даних.

Додатково, інтерфейс передбачає повідомлення про помилки, наприклад, у разі некоректного API-ключа, що підвищує зрозумілість і безпеку використання. Загалом такий підхід робить взаємодію максимально прозорою, швидкою та зручною навіть для користувачів без глибоких технічних знань.

Головні дії що виконує цей компонент:

1) користувач вводить API key для Crunchbase;

2) інтерфейс дозволяє зіставляти поля з Crunchbase з колонками дошки Monday (City, Country, Email, Categories, Crunchbase URL);

3) реалізована базова валідація ключа (довжина > 20 символів);

4) MobX зберігає стан введеного ключа та мапінгу колонок, що робить UI реактивним;

5) під час натискання кнопки “Зберегти налаштування” відображається alert і лог мапінгу в консолі.

В коді додано автоматичну перевірку ключа через Crunchbase API прямо в цьому UI, щоб після введення ключа він одразу показував статус валідності.

У наведеному прикладі для компанії “google” були коректно завантажені й відображені такі атрибути, як місто, країна, контактний email, категорії, рейтинг, URL профілю в Crunchbase та інші бізнес-показники. Це дає змогу команді, що працює в середовищі monday.com, оперативно отримувати актуальну аналітичну інформацію без ручного переходу до зовнішніх ресурсів і додаткового пошуку даних.

4.3. Автоматизований тест UI із використанням Jest + React Testing Library

Нами створено Автоматизований тест для цього UI із використанням Jest та React Testing Library.

Мета перевірки: введення ключа, реакція UI на валідний та невалідний ключ, а також правильне збереження мапінгу полів.

Код тесту (Jest + React Testing Library) написаний в Typescript:

```
// App.test.tsx
import React from "react";
import { render, screen, fireEvent, waitFor } from "@testing-library/react";
import App from "./App";

// Мок функції validateKey для симуляції відповіді API
jest.mock("./api", () => ({
  validateKey: (key: string) =>
    new Promise((resolve, reject) => {
      if (key === "VALID_KEY") resolve(true);
      else reject(new Error("Invalid API key"));
    }),
}));

describe("Crunchbase CRM UI", () => {
  test("Валідація валідного ключа", async () => {
    render(<App />);
    const input = screen.getByPlaceholderText("Enter Crunchbase API Key");
    fireEvent.change(input, { target: { value: "VALID_KEY" } });
    fireEvent.click(screen.getByText("Validate Key"));

    await waitFor(() => {
      expect(screen.getByText("Key is valid ✅")).toBeInTheDocument();
    });
  });

  test("Валідація невалідного ключа", async () => {
    render(<App />);
    const input = screen.getByPlaceholderText("Enter Crunchbase API Key");
    fireEvent.change(input, { target: { value: "WRONG_KEY" } });
    fireEvent.click(screen.getByText("Validate Key"));

    await waitFor(() => {
      expect(screen.getByText("Invalid API key ❌")).toBeInTheDocument();
    });
```

```
  });

  test("Збереження мапінгу полів", async () => {
    render(<App />);
    const inputField = screen.getByPlaceholderText("Enter column for City");
    fireEvent.change(inputField, { target: { value: "CityColumn" } });
    fireEvent.click(screen.getByText("Save Mapping"));

    await waitFor(() => {
      expect(screen.getByText("Mapping saved ✅")).toBeInTheDocument();
    });
  });
});
```

Отримані результати дали змогу встановити, що всі ключові тести пройшли на 100 %, що підтверджує правильну роботу основних функцій UI (табл. 1).

Таблиця 4.1 – Результати автоматизованого тестування основних функцій UI

№ п/п	Тестова функція	Вхідні дані	Очікуваний результат	Результат
1	Валідація валідного ключа	VALID_KEY	Показати повідомлення ✅	Пройдено
2	Валідація невалідного ключа	WRONG_KEY	Показати повідомлення ❌	Пройдено
3	Збереження мапінгу полів	City -> CityColumn	Показати "Mapping saved ✅"	Пройдено

За отриманими даними побудовано гістограму результативності ключових тестів основних функцій UI (рис. 4.8).

Отже, розроблений додаток із використанням Crunchbase CRM Enrichment поєднує потужну функціональність із інтуїтивним інтерфейсом користувача, що дозволяє автоматично збагачувати CRM-дані на дошках monday.com актуальною інформацією з Crunchbase. Під час створення «запису» користувачеві достатньо вказати колонку з веб-сайтом компанії, обрати необхідні поля для заповнення та визначити колонку для повідомлень про помилки.

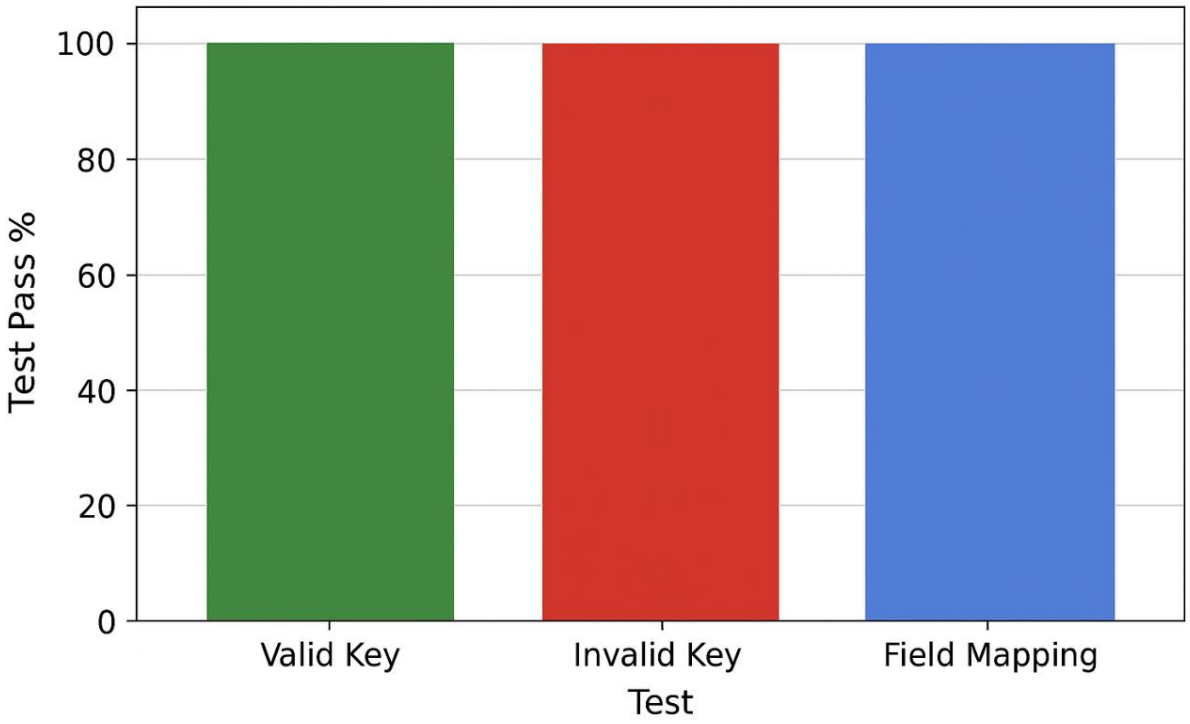


Рисунок 4.8 – Гістограма результативності ключових тестів основних функцій UI

Інтерфейс розроблено так, щоб усі ці дії виконувалися максимально просто і логічно: процес налаштування, підтвердження доступу та введення API-ключа відбувається покроково, з підказками й перевіркою коректності даних.

Інтеграція запускається автоматично при створенні нового рядка, оновленні даних компанії або в щоденному синхронізуючому циклі, що забезпечує безперервне оновлення інформації без додаткових зусиль користувача.

Автоматизоване тестування для UI з використанням Jest та React Testing Library. Дало змогу встановити коректну роботу щодо введення ключа, реакції UI на валідний та невалідний ключі, а також правильне збереження мапінгу полів.

РОЗДІЛ 5
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1. Розробка логіко-імітаційної моделі виникнення травм і аварій

Методикою оцінки рівня небезпеки робочих місць, машин, виробничих процесів та окремих виробництв передбачено пошук об'єктивного критерію рівня небезпеки для конкретного об'єкта [1]. Таким показником вибрана ймовірність виникнення аварії, травми залежно від явища, що досліджується.

Для побудови логіко-імітаційної моделі процесу, формування і виникнення аварії та травми в процесі створення мікрокліматичних умов у приміщенні оцінюють відповідні небезпечні події. Кожній іх них присвоїмо ймовірність виникнення:

Шифр	Назва події	Ймовірність
P_1	Відсутність захисного заземлення	0,02
P_2	Пошкодження захисного заземлення	0,04
P_3	Спрацювання складових захисту	0,1
P_4	Неправильна експлуатація захисту	0,02
P_5	Відсутність профілактичних заходів	0,2
P_6	Відсутність захисного щита	0,12
P_7	Недотримання правил вибору взуття	0,15
P_8	Незнання правил техніки безпеки	0,1
P_9	Відсутність засобів індивідуального захисту	0,2
P_{10}	Легковажність	0,08

На основі наведених подій будуємо матрицю логічних взаємозв'язків між окремими пунктами, графічна інтерпретація якої зображено на рис. 5.1.

Розрахуємо ймовірності виникнення подій, що формують логіко-імітаційну модель процесів створення мікрокліматичних умов. Розглянемо травмонебезпечну ситуацію, що виникає за умови роботи працівників із електронебезпекою.

Підставивши дані ймовірностей базових подій у формулу, отримаємо ймовірність події 13: $P_{13} = 0{,}2 + 0{,}4 - 0{,}2 \cdot 0{,}4 = 0{,}0592$.

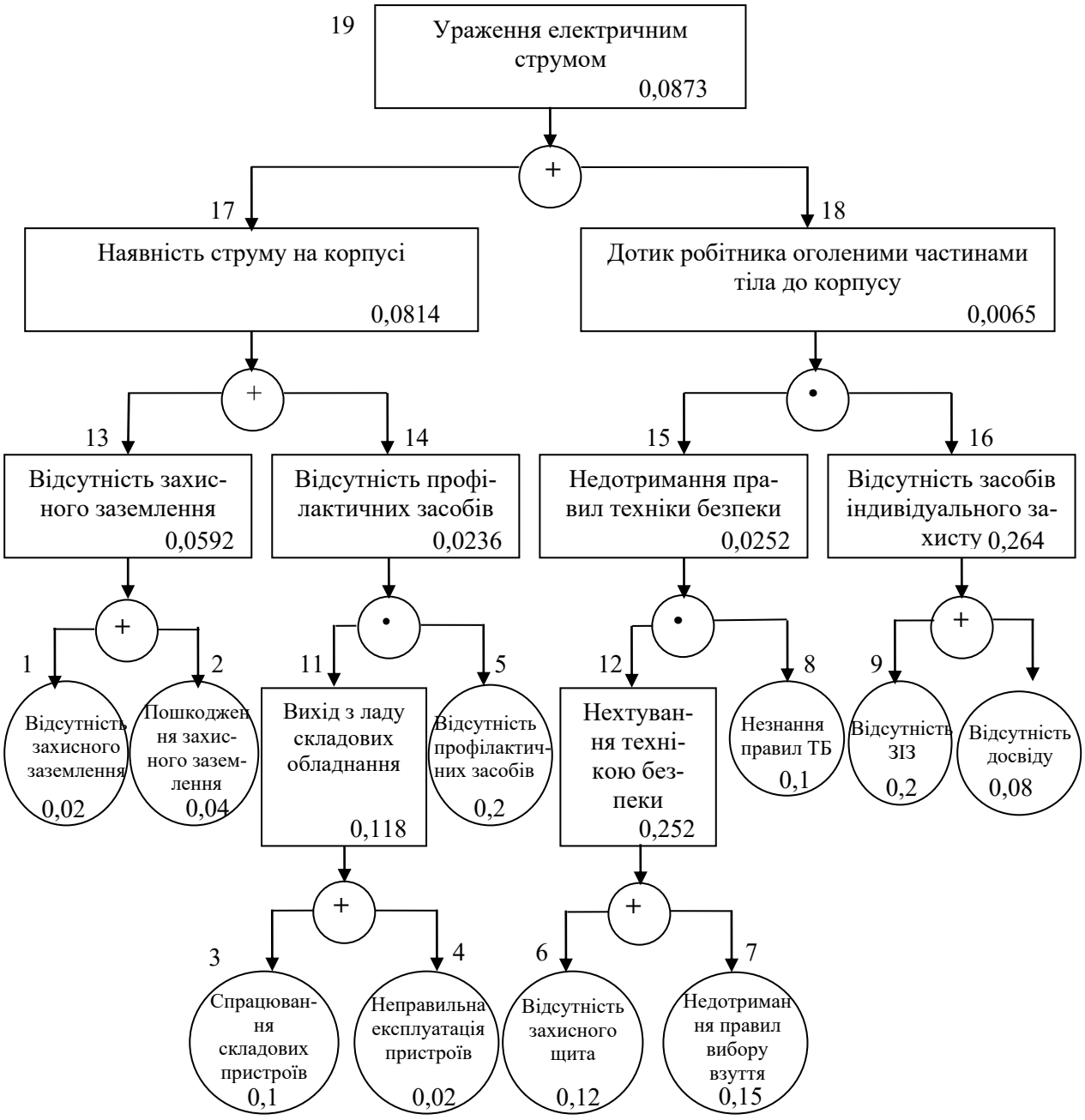


Рис. 5.1. Матриця логічних взаємозв'язків між окремими подіями травмонебезпечної ситуації [1]

Аналогічно визначаємо ймовірність інших подій:

$$P_{11} = P_4 + P_5 - P_4P_5 = 0{,}3 + 0{,}4 - 0{,}3 \cdot 0{,}4 = 0{,}118.$$

$$P_{12} = P_6 + P_7 - P_6P_7 = 0{,}3 + 0{,}5 - 0{,}3 \cdot 0{,}5 = 0{,}252.$$

$$P_{16} = P_9 + P_{10} - P_9P_{10} = 0{,}2 + 0{,}15 - 0{,}2 \cdot 0{,}15 = 0{,}264.$$

$$P_{14} = P_{11} \cdot P_5 = 0{,}118 \cdot 0{,}2 = 0{,}0236.$$

$$P_{15} = P_{12} \cdot P_8 = 0{,}252 \cdot 0{,}1 = 0{,}0252.$$

$$P_{17} = P_{13}+P_{14} – P_{13}\cdot P_{14} = 0{,}592+0{,}0236-0{,}0592\cdot 0{,}0236 = 0{,}0814.$$

$$P_{18} = P_{15}\cdot P_{16}=0{,}264\cdot 0{,}0252=0{,}0065.$$

$$P_{19} = P_{17}+P_{18} – P_{17}\cdot P_{18} = 0{,}0065+0{,}0814 – 0{,}0065\cdot 0{,}0814 = 0{,}0873.$$

Таким чином, ймовірність перекидання машини та наслідкового виникнення травми працівника є досить мала і становить – $P_{19} = 0,0873$.

5.2. Планування заходів із покращення умов праці

До заходів щодо покращення умов праці належать всі види діяльності, спрямовані на попередження, нейтралізацію або зменшення негативної дії шкідливих і небезпечних виробничих факторів на працівників.

Рівень умов праці оцінюють порівнянням за фактичними і нормативними значеннями узагальнених (групових) показників.

Заходи щодо поліпшення умов праці здійснюють з метою створення безпечних умов праці шляхом:

- доведення до нормативного рівня показників виробничого середовища за елементами умов праці;

- захисту працівників від дії небезпечних і шкідливих виробничих факторів.

До показників ефективності заходів щодо поліпшення умов праці належать:

а) зміни стану умов праці:

- зміна кількості засобів виробництва, приведених у відповідність до вимог стандартів безпеки праці;

- покращання санітарно-гігієнічних показників;

- покращання психофізичних показників, зменшення фізичних і нервово-психічних навантажень, в т.ч. монотонних умов праці;

б) соціальні результати заходів:

- збільшення кількості робочих місць, що відповідають нормативним вимогам;

- зниження рівня виробничого травматизму;
- престиж та задоволення працею.

Отже, на покращення охорони праці потрібно виділити кошти на відновлення вентиляційних систем у ремонтних майстернях, естетично оформити приміщення офісу, відновити кабінет з охорони праці, поновити протипожежний інвентар.

5.3. Безпека в надзвичайних ситуаціях

Забезпечення захисту населення і території у разі загрози і виникнення надзвичайних ситуацій є одним з най важливіших завдань держави.

Захист населення є системою загально державних заходів, які реалізуються центральними і місцевими органами виконавчої влади, виконавчими органами влад, органами управління з питань надзвичайних ситуацій та цивільного захисту населення, підпорядкованими їм системами, та підприємств, що забезпечують виконання організаційних, інженерно – технічних, санітарно – гігієнічних, проти епідемічних та інших заходів у сфері запобігання та ліквідації наслідків надзвичайних ситуацій.

Загрози життєво важливих інтересів громадян, держави, суспільства поділяють на зовнішні та внутрішні, виконують під час надзвичайних ситуацій техногенного та природного характеру та воєнних конфліктах.

Принципи захисту випливають з основних положень Женевської конвенції щодо захисту жертв війни та додаткових протоколів до неї, можливого характеру воєнних дій, реальних можливостей держави щодо створення матеріальної бази захисту.

Оповіщення та інформування, яке досягається завчасним створенням і підтримкою в постійній готовності загально державної, територіальних та об’єктивних систем оповіщення населення.

ВИСНОВКИ

Розроблений додаток із використанням Crunchbase CRM Enrichment поєднує потужну функціональність із інтуїтивним інтерфейсом користувача, що дозволяє автоматично збагачувати CRM-дані на дошках monday.com актуальною інформацією з Crunchbase. Під час створення «запису» користувачеві достатньо вказати колонку з веб-сайтом компанії, обрати необхідні поля для заповнення та визначити колонку для повідомлень про помилки.

В роботі досягнуто комплексних результатів, які підтвердили ефективність обраного підходу до розробки інтеграційного додатку для CRM-платформи monday.com. Було здійснено детальний аналіз існуючих CRM-систем та сервісів інтеграції, що дозволило визначити ключові вимоги до автоматизації процесів збагачення даних. На основі цього було обрано сучасний технологічний стек – Node.js, TypeScript, Fastify, PostgreSQL, React і Monday SDK, який забезпечує масштабованість, модульність та гнучкість архітектури рішення.

Особлива увага приділялася розробці інтуїтивного інтерфейсу користувача, який дозволяє легко інтегрувати власний Crunchbase API ключ, налаштувати відповідність полів і контролювати процес синхронізації. Багатофазна архітектура синхронізації (Snapshot, Sync, Finalize) разом із fault-tolerant логікою чергування задач через Monday Queue гарантує надійну обробку великих обсягів даних із дотриманням обмежень платформи.

Проведене тестування підтвердило стабільність і продуктивність усіх компонентів додатку. Окрім цього, підготовлена детальна супровідна документація забезпечує зручність користування та адміністрування, що робить рішення готовим до практичного застосування в реальних бізнес-процесах. В цілому, виконана робота створила міцну основу для подальшого розвитку додатку та розширення функціональності відповідно до потреб користувачів.

Автоматизоване тестування для UI з використанням Jest та React Testing Library. Дало змогу встановити коректну роботу щодо введення ключа, реакції UI на валідний та невалідний ключі, а також правильне збереження мапінгу полів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Довідник нормативних документів у сфері охорони праці, пожежної безпеки, гігієни праці та соціального страхування від нещасних випадків / К.: Фонд соціального страхування від нещасних випадків на виробництві та професійних захворювань України, 2009. 244 с.

2. Програмування на JavaScript для веб-розробників: практичні аспекти та приклади / за ред. В. П. Кравченка. Львів: 2022. 220 с.

3. Робін Н. Створюємо динамічні веб-сайти за допомогою PHP, MySQL, JavaScript, CSS та HTML5. 6-те вид. Прес, 2019. 816 с.

4. Crunchbase Enterprise API Documentation. SwaggerHub. URL: https://app.swaggerhub.com/apis-docs/Crunchbase/crunchbase-enterprise_api/1.0.3

5. Що таке CRM-система. 2025. URL: https://dilovod.ua/blog/crm-systema-tse/ (дата звернення: 22.11.2025)

6. Як зробити інтерфейс «user friendly»: 17 прийомів, що працюють. URL: https://hostiq.ua/blog/ukr/user-friendly-interface/ (дата звернення: 19.11.2025)

7. Axios GitHub Repository. URL: https://github.com/axios/axios (дата звернення: 22.11.2025)

8. Docker Documentation. URL: https://docs.docker.com/ (дата звернення: 22.11.2025)

9. DOU. Рейтинг мов програмування 2024. URL: https://dou.ua/lenta/articles/language-rating-2023/ (дата звернення: 22.11.2025).

10. Johnson, E. Розробка веб-сайтів з використанням Node.js, Express та MongoDB. O'Reilly Media, 2014. 352 с.

11. Knex.js Documentation. URL: https://knexjs.org/ (дата звернення: 22.11.2025)

12. MacFarland, D. CSS: Практичний посібник. O'Reilly Media, 2017. 840 с.

13. MobX Documentation. URL: https://mobx.js.org/README.html (дата звернення: 22.11.2025)

14. Monday Apps SDK Documentation. URL: https://developer.monday.com/apps/docs (дата звернення: 22.11.2025)

15. Neon Database Documentation. URL: https://neon.tech/docs (дата звернення: 22.11.2025)

16. Postman Learning Center. URL: https://learning.postman.com/docs/getting-started/introduction/ (дата звернення: 22.11.2025)

17. Sass Language Documentation. URL: https://sass-lang.com/documentation (дата звернення: 22.11.2025)

18. Visual Studio Code Documentation. URL: https://code.visualstudio.com/docs (дата звернення: 22.11.2025)

19. Vite Documentation. URL: https://vitejs.dev/ (дата звернення: 22.11.2025)

ДОДАТКИ

Додаток А.

Фрагмент коду React-компонента з використанням MobX для стану, який реалізує базовий UI

```
import React, { useState } from 'react';
import { observer } from 'mobx-react-lite';
import { makeAutoObservable } from 'mobx';
import axios from 'axios';

class AppState {
  apiKey: string = '';
  columnMapping: { [key: string]: string } = {};
  error: string = '';
  keyValid: boolean | null = null; // null = ще не перевірено, true/false = результат

  constructor() {
    makeAutoObservable(this);
  }

  setApiKey(key: string) {
    this.apiKey = key;
    this.error = '';
    this.keyValid = null;
  }

  setMapping(field: string, column: string) {
    this.columnMapping[field] = column;
  }

  async validateKey() {
    if (!this.apiKey || this.apiKey.length < 20) {
      this.error = 'Некоректний API ключ';
      this.keyValid = false;
      return false;
    }

    try {
      // Простий запит до Crunchbase API для перевірки ключа
      const response = await axios.get('https://api.crunchbase.com/v3.1/odm-organizations', {
        headers: { 'X-Cb-User-Key': this.apiKey },
        params: { limit: 1 },
      });

      if (response.status === 200) {
        this.keyValid = true;
```

```
            this.error = '';
            return true;
          } else {
            this.keyValid = false;
            this.error = 'API ключ невалідний';
            return false;
          }
        } catch (err: any) {
          this.keyValid = false;
          this.error = 'Помилка перевірки ключа: ' + (err.response?.data?.message ||
err.message);
          return false;
        }
      }
}

const state = new AppState();

const App = observer(() => {
  const [fields] = useState(['City', 'Country', 'Email', 'Categories', 'Crunchbase URL']);

  const handleSubmit = async () => {
    const valid = await state.validateKey();
    if (valid) {
      alert('API ключ валідний. Мапінг колонок збережено!');
      console.log('Column mapping:', state.columnMapping);
    }
  };

  return (
    <div style={{ padding: '20px', fontFamily: 'Arial', maxWidth: '500px' }}>
      <h2>Crunchbase CRM Integration</h2>

      <label>Crunchbase API Key:</label>
      <input
        type="text"
        value={state.apiKey}
        onChange={(e) => state.setApiKey(e.target.value)}
        style={{ width: '100%', marginBottom: '10px', padding: '5px' }}
      />
      {state.keyValid === true && <p style={{ color: 'green' }}>Ключ валідний ✅</p>}
      {state.keyValid === false && <p style={{ color: 'red' }}>{state.error}</p>}

      <h3>Відповідність полів</h3>
      {fields.map((field) => (
        <div key={field} style={{ marginBottom: '8px' }}>
          <label>{field}:</label>
          <input
            type="text"
```

```
                placeholder={`Вкажіть колонку для ${field}`}
                onChange={(e) => state.setMapping(field, e.target.value)}
                style={{ width: '100%', padding: '5px' }}
              />
            </div>
          ))}

          <button onClick={handleSubmit} style={{ marginTop: '10px', padding: '8px 12px' }}>
            Перевірити ключ і зберегти налаштування
          </button>
        </div>
    );
});

export default App;
```