Метадані

ДОКУМЕНТ

Заголовок

2026_Осадчук_В.Ю._ФІТ_ІТТС_ІСТ-22-1.

Автор	Науковий керівник / Експерт	ІД документу
Осадчук В.Ю.	**Зікратий С.В.**	**334340102**

ОРГАНІЗАЦІЯ

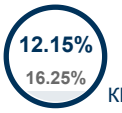
Назва організації	підрозділ
Ivano-Frankivsk National Technical University of Oil and Gas	**Каф. ІТТС**

ЗВІТ

Дата звіту	Дата редагування
6/16/2026	**6/16/2026**

Обсяг знайдених подібностей

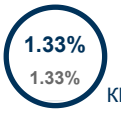
Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

13844

Кількість слів



КЦ

122544

Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	11
Парафрази (SmartMarks)	a	127

Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз	Колір тексту

#	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2026_Неймет_В.Д._ФІТ_ІТТС_ІСТ-22-1 [redacted] 6/12/2026	87 (0.63 %)

	Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)	
2	2026_Неймет_В.Д._ФІТ_ІТТС_ІСТ-22-1 [redacted] 6/12/2026 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)	31 (0.22 %)
3	2025_ПІ_БД_ПЗПІ-24-4_Шевцов_М_Д_скорочений [redacted] 1/15/2026 Kharkiv National University of Radio Electronics (каф. ПІ)	29 (0.21 %)
4	2026_Неймет_В.Д._ФІТ_ІТТС_ІСТ-22-1 [redacted] 6/12/2026 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)	26 (0.19 %)
5	2025_ПІ_БД_ПЗПІ-24-4_Шевцов_М_Д_скорочений [redacted] 1/15/2026 Kharkiv National University of Radio Electronics (каф. ПІ)	26 (0.19 %)
6	http://194.8.51.241/bitstream/123456789/9654/1/BR_Piaternia.pdf	25 (0.18 %)
7	2025_ПІ_БД_ПЗПІ-24-4_Шевцов_М_Д_скорочений [redacted] 1/15/2026 Kharkiv National University of Radio Electronics (каф. ПІ)	24 (0.17 %)
8	База даних аеропорта [redacted] 5/29/2026 National University Chernihiv Politechnika (NUCP) 2 (Інформаційний центр запобігання та виявлення плагіату)	24 (0.17 %)
9	2025_ПІ_БД_ПЗПІ-24-4_Шевцов_М_Д_скорочений [redacted] 1/15/2026 Kharkiv National University of Radio Electronics (каф. ПІ)	22 (0.16 %)
10	ІПЗ-51м Ковтунець Владислав Курсова Робота.pdf [redacted] 6/4/2026 National University of Water and Environmental Engineering (National University of Water and Environmental Engineering)	21 (0.15 %)

Домашня база даних (4.84 %)

#	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2026_Грицак_О.О._ФІТ_ІТТС_ІСТ-22-1 [redacted] 6/11/2026 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)	446 (46) (3.22 %)
2	2026_Неймет_В.Д._ФІТ_ІТТС_ІСТ-22-1 [redacted] 6/12/2026 **Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)**	**175 (8) (1.26 %)**
3	ПЗ Маковійчук (1) [redacted] 6/10/2025 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. КСМ)	26 (4) (0.19 %)
4	ПЗ_БР_Дем'яник_02 [redacted] 6/3/2025 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. КСМ)	13 (2) (0.09 %)
5	ПЗ_БР_Копко [redacted] 5/31/2025 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. КСМ)	5 (1) (0.04 %)
6	2025_Бондар Н.В._ФІТ_ІТТС_ІСТ-21-1 [redacted] 6/24/2025 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)	5 (1) (0.04 %)

Програма обміну базами даних (6.16 %)

#	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
7	2025_ПІ_БД_ПЗПІ-24-4_Шевцов_М_Д_скорочений [redacted] 1/15/2026 **Kharkiv National University of Radio Electronics (каф. ПІ)**	**497 (41) (3.59 %)**
8	ВКР_Буран [redacted] 2/9/2026 State University of Trade and Economics (Кафедра комп'ютерних наук та інформаційних систем)	56 (8) (0.4 %)
9	Розробка веб-застосунку для управління особистими фінансами [redacted] 3/12/2026 Donbass Institute of Technology and Management (Donbass Institute of Technology and Management)	35 (3) (0.25 %)
10	База даних аеропорта [redacted] 5/29/2026 National University Chernihiv Politechnika (NUCP) 2 (Інформаційний центр запобігання та виявлення плагіату)	33 (2) (0.24 %)
11	Розробка транспортної логістичної вебплатформи з використанням React.js [redacted] 5/24/2023 National University "Zaporizhzhia Polytechnic" (Кафедра "Комп'ютерні системи та мережі")	22 (2) (0.16 %)
12	курсова омельчук [redacted] 5/20/2026 Establishment of the Lviv Regional Council "Sambir Professional Pedagogical College Named after Ivan Fylypchak" (Sambir Professional Pedagogical College named after Ivan Fylypchak)	22 (4) (0.16 %)
13	ІПЗ-51м Ковтунець Владислав Курсова Робота.pdf [redacted] 6/4/2026 National University of Water and Environmental Engineering (National University of Water and Environmental Engineering)	21 (1) (0.15 %)
14	122_Mokhonko_Illia_Kostiantynovych_it2025 [redacted] 6/4/2025 National University of Food Technologies (Кафедра інформаційних технологій, штучного інтелекту і кібербезпеки)	20 (2) (0.14 %)
15	МФІТ бакалаври 2026 [redacted] 6/13/2026 Odessa I.I.Mechnikov National University (Одеський національний університет імені І.І.Мечникова)	17 (2) (0.12 %)
16	2026_ФеС-41_Пасічник_В_О_текст [redacted] 6/10/2026 The Ivan Franko National University (Факультет електроніки та комп'ютерних технологій)	15 (1) (0.11 %)
17	«Розробка веб-додатку "Мій список справ"». Дипломна робота. [redacted] 5/22/2026 Zhytomyr Agricultural Technical Professional College (Zhytomyr Agricultural Technical Professional College)	15 (2) (0.11 %)
18	2026 КвРбак - Горбачьова [redacted] 6/9/2026 Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)	13 (2) (0.09 %)
19	ПЗДП_ІСТ_КПІ_2026_Філь В.О. [redacted] 6/11/2026 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-ра інформаційних систем та технологій)	13 (2) (0.09 %)
20	ІНТЕЛЕКТУАЛЬНИЙ ПОШУК НАВЧАЛЬНИХ МАТЕРІАЛІВ	12 (2) (0.09 %)

	[redacted] 6/1/2026 Taras Shevchenko National University of Kyiv (Факультет комп'ютерних наук та кібернетики)	
21	2025_KRB_SN-42_Liashenko_HO [redacted] 6/17/2025 Ternopil Ivan Pul'uj National Technical University (кафедра комп'ютерних наук)	10 (2) (0.07 %)
22	Плиска Дмитро плагіат [redacted] 6/10/2026 West Ukrainian National University (West Ukrainian National University)	10 (1) (0.07 %)
23	Диплом Богуславський В. І. [redacted] 6/3/2026 Zaporizhzhia National University (Кафедра електроніки, інформаційних систем та програмного забезпечення (спеціальність 121 Інженерія програмного забезпечення))	8 (1) (0.06 %)
24	БР_Козир_плагіат [redacted] 6/5/2026 State University of Intellectual Technologies and Communications (Кафедра комп'ютерної інженерії та інформаційних систем)	7 (1) (0.05 %)
25	Pavlusenko_Donets_Diplom_2026 [redacted] 6/3/2026 V. N. Karazin Kharkiv National University (KKNU) (ННІ комп'ютерних наук та штучного інтелекту - кафедра математичного моделювання та аналізу даних)	6 (1) (0.04 %)
26	Розроблення інформаційної системи футбольного клубу «Оріон» засобами React [redacted] 6/15/2025 National Forestry University of Ukraine (ЦДН НЛТУ України)	6 (1) (0.04 %)
27	3 поверховий житловий будинок для багатодітних сімей [redacted] 12/10/2025 Kamyanets-Podilskyi Instrument Applied College of Construction, Architecture and Design (Kamyanets-Podilskyi Instrument Applied College of Construction, Architecture and Design)	5 (1) (0.04 %)
28	CheperysYS_MCS01_master_2025 [redacted] 12/8/2025 GoIT Neoversity (GoIT Neoversity)	5 (1) (0.04 %)
29	1303-4-06_2026_Б-81 [redacted] 5/27/2026 National University "Lviv Politechnika" (NULP2)	5 (1) (0.04 %)

Інтернет (5.24 %)

#	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
30	https://essuir.sumdu.edu.ua/bitstreams/99888b6a-4f59-47c5-b3af-8e0f7956825d/download	189 (18) (1.37 %)
31	https://codezup.com/securing-nodejs-apis-authentication-authorization/	139 (14) (1 %)
32	**http://194.8.51.241/bitstream/123456789/9654/1/BR_Piaternia.pdf**	**56 (4) (0.4 %)**
33	https://jishuzhan.net/article/2017496585501458433	56 (4) (0.4 %)
34	https://stackoverflow.com/questions/46115993/mean-app-error-expected-object	43 (4) (0.31 %)
35	https://stackoverflow.com/questions/78588048/maximum-update-depth-limit-exceeded-problem-coming-while-using-react-dnd	39 (4) (0.28 %)
36	https://essuir.sumdu.edu.ua/bitstreams/db1a2e0d-ee9b-4bb6-b8c5-77ae4d47ea10/download	28 (3) (0.2 %)
37	https://document.kdu.edu.ua/diplom/2025_bak_6340.pdf	28 (3) (0.2 %)

38	https://teachmeidea.com/oauth2-jwt-session-tokens-explained/	21 (2) (0.15 %)
39	https://openarchive.nure.ua/server/api/core/bitstreams/53b21898-c554-4713-b289-3e87b1df46c4/content	18 (3) (0.13 %)
40	https://1library.net/document/zkwnxvx8-%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D1%8E%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%97-%D1%82%D0%B5%D1%85%D0%BD%D1%96%D0%BA%D0%B8-%D0%B0%D1%84%D0%B5%D0%B4%D1%80%D0%B8-%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B8%D0%B9-%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82-%D1%96%D1%96%D0%BC%D1%84%D1%80%D1%96l%D1%96-%D0%B0%D1%81%D0%B2%D1%96%D0%B4%D1%87%D1%83%D1%8E-%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%BE%D0%BC%D1%83.html	18 (3) (0.13 %)
41	https://essuir.sumdu.edu.ua/bitstream-download/123456789/99550/1/Korchmenko_bac_rob.pdf;jsessionid=DDFB1BC8804AC8E9B5DF8296B4B0CCB0	17 (1) (0.12 %)
42	http://ir.polissiauniver.edu.ua/bitstream/123456789/17076/1/Zaverukha_MM_KR_122_2025.pdf	16 (2) (0.12 %)
43	https://hong-jh.tistory.com/entry/%ED%86%A0%ED%81%B0-%EA%B8%B0%EB%B0%98-%EC%9D%B8%EC%A6%9D-%ED%9A%A8%EC%9C%A8%EC%A0%81%EC%9C%BC%EB%A1%9C-%EA%B4%80%EB%A6%AC%ED%95%98%EB%8A%94-%EB%B2%95	14 (1) (0.1 %)
44	https://ela.kpi.ua/bitstream/123456789/57753/1/Kovalenko_bakalavr.pdf	12 (1) (0.09 %)
45	https://studfile.net/preview/15326912/	11 (1) (0.08 %)
46	https://devtechtutor.com/index.php/2024/06/21/building-a-crm-system-with-node-js-and-express/	9 (1) (0.07 %)
47	https://elartu.tntu.edu.ua/bitstream/lib/45516/1/dyplom_Datsko_2024.pdf	7 (1) (0.05 %)
48	https://zweck.io/jwt-authentication-in-node-js-with-middleware-a-secure-approach-for-web-applications/	5 (1) (0.04 %)

Список прийнятих фрагментів

#	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)
2026_Грицак_О.О._ФІТ_ІТТС_ІСТ-22-1		**421 (3.04%)**
1	А. В. Затверд. ЗАМІХОВСЬКА О. Л	6 (0.04%)
2	1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА	5 (0.04%)
3	Змн. Арк. No докум. Підпис Дата Арк. 9 КБР.ІСТ	10 (0.07%)
4	Змн. Арк. No докум. Підпис Дата Арк. 10 КБР.ІСТ	10 (0.07%)
5	Змн. Арк. No докум. Підпис Дата Арк. 11 КБР.ІСТ	10 (0.07%)
6	Змн. Арк. No докум. Підпис Дата Арк. 13 КБР.ІСТ	10 (0.07%)
7	Змн. Арк. No докум. Підпис Дата Арк. 14 КБР.ІСТ	10 (0.07%)
8	Змн. Арк. No докум. Підпис Дата Арк. 16 КБР.ІСТ	10 (0.07%)

9	Змн. Арк. No докум. Підпис Дата Арк. 17 КБР.ІСТ	10 (0.07%)
10	Змн. Арк. No докум. Підпис Дата Арк. 18 КБР.ІСТ	10 (0.07%)
11	Змн. Арк. No докум. Підпис Дата Арк. 21 КБР.ІСТ	10 (0.07%)
12	Змн. Арк. No докум. Підпис Дата Арк. 22 КБР.ІСТ	10 (0.07%)
13	Змн. Арк. No докум. Підпис Дата Арк. 23 КБР.ІСТ	10 (0.07%)
14	Змн. Арк. No докум. Підпис Дата Арк. 24 КБР.ІСТ	10 (0.07%)
15	Змн. Арк. No докум. Підпис Дата Арк. 25 КБР.ІСТ	10 (0.07%)
16	Змн. Арк. No докум. Підпис Дата Арк. 26 КБР.ІСТ	10 (0.07%)
17	Змн. Арк. No докум. Підпис Дата Арк. 27 КБР.ІСТ	10 (0.07%)
18	Змн. Арк. No докум. Підпис Дата Арк. 28 КБР.ІСТ	10 (0.07%)
19	Змн. Арк. No докум. Підпис Дата Арк. 29 КБР.ІСТ	10 (0.07%)
20	Змн. Арк. No докум. Підпис Дата Арк. 30 КБР.ІСТ	10 (0.07%)
21	Змн. Арк. No докум. Підпис Дата Арк. 31 КБР.ІСТ	10 (0.07%)
22	Змн. Арк. No докум. Підпис Дата Арк. 32 КБР.ІСТ	10 (0.07%)
23	Змн. Арк. No докум. Підпис Дата Арк. 34 КБР.ІСТ	10 (0.07%)
24	Змн. Арк. No докум. Підпис Дата Арк. 35 КБР.ІСТ	10 (0.07%)
25	Змн. Арк. No докум. Підпис Дата Арк. 36 КБР.ІСТ	10 (0.07%)
26	Змн. Арк. No докум. Підпис Дата Арк. 37 КБР.ІСТ	10 (0.07%)
27	Змн. Арк. No докум. Підпис Дата Арк. 38 КБР.ІСТ	10 (0.07%)
28	Змн. Арк. No докум. Підпис Дата Арк. 39 КБР.ІСТ	10 (0.07%)
29	Змн. Арк. No докум. Підпис Дата Арк. 40 КБР.ІСТ	10 (0.07%)
30	Змн. Арк. No докум. Підпис Дата Арк. 41 КБР.ІСТ- 07.00.00.000 ПЗ Рисунок 3.1	13 (0.09%)
31	Змн. Арк. No докум. Підпис Дата Арк. 42 КБР.ІСТ	10 (0.07%)
32	Змн. Арк. No докум. Підпис Дата Арк. 44 КБР.ІСТ	10 (0.07%)
33	Змн. Арк. No докум. Підпис Дата Арк. 45 КБР.ІСТ	10 (0.07%)
34	Змн. Арк. No докум. Підпис Дата Арк. 46 КБР.ІСТ	10 (0.07%)
35	Змн. Арк. No докум. Підпис Дата Арк. 47 КБР.ІСТ	10 (0.07%)
36	Змн. Арк. No докум. Підпис Дата Арк. 49 КБР.ІСТ	10 (0.07%)
37	Змн. Арк. No докум. Підпис Дата Арк. 50 КБР.ІСТ	10 (0.07%)
38	Змн. Арк. No докум. Підпис Дата Арк. 52 КБР.ІСТ	10 (0.07%)
39	Змн. Арк. No докум. Підпис Дата Арк. 53 КБР.ІСТ	10 (0.07%)
40	Змн. Арк. No докум. Підпис Дата Арк. 54 КБР.ІСТ	10 (0.07%)

41	Змн. Арк. No докум. Підпис Дата Арк. 55 КБР.ІСТ	10 (0.07%)
42	Змн. Арк. No докум. Підпис Дата Арк	7 (0.05%)
43	Змн. Арк. No докум. Підпис Дата Арк. 58 КБР.ІСТ	10 (0.07%)
ПЗ Маковійчук (1)		**8 (0.06%)**
1	Змн. Арк. No докум. Підпис Дата Арк. 43	8 (0.06%)
ПЗ_БР_Дем'яник_02		**7 (0.05%)**
1	Змн. Арк. No докум. Підпис Дата Арк	7 (0.05%)
2025_ПІ_БД_ПЗПІ-24-4_Шевцов_М_Д_скорочений		**101 (0.73%)**
1	CREATE TABLE room room id` int NOT NULL AUTO_INCREMENT,	8 (0.06%)
2	50)COLLATE utf8mb4_unicode_ci	8 (0.06%)
3	ENGINE=InnoDB AUTO_INCREMENT= 7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicod...	17 (0.12%)
4	CREATE TABLE partner partner id` int NOT NULL AUTO_INCREMENT,	8 (0.06%)
5	100) COLLATE utf8mb4_unicode_ci NOT NULL	10 (0.07%)
6	id`) REFERENCES group group id`) ON DELETE RESTRICT ON UPDATE CASCADE) ENGINE...	26 (0.19%)
7	id` int NOT NULL utility type varchar(100 COLLATE utf8mb4_unicode_ci NOT NULL...	14 (0.1%)
8	start_date` date NOT NULL, agreement end_date` date DEFAULT NULL,	10 (0.07%)
База даних аеропорта		**24 (0.17%)**
1	email` varchar(100) COLLATE utf8mb4_unicode_ci NOT NULL, `password_varchar(255...	24 (0.17%)
ВКР_Буран		**6 (0.04%)**
1	FOREIGN KEY (`manager_id`) REFERENCES `users`	6 (0.04%)

КВАЛІФІКАЦІЙНА
БАКАЛАВРСЬКА РОБОТА КБР.ІСТ- 13.00.00.000 ПЗ Група ІСТ-22-1
Валентина ОСАДЧУК
2026 Івано-Франківський Національний Технічний Університет Нафти і Газу
Інститут Інформаційних Технологій Кафедра інформаційно-телекомунікаційних технологій і систем

Осадчук Валентина Юріївна

УДК 004.9:005.8

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА
Розробка інформаційної системи для управління проєктами компанії

Інформаційні системи та технології
(назва освітньої програми)

126 - Інформаційні системи та технології
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня _____Осадчук В. Ю._____
(підпис, прізвище та ініціали здобувача) Науковий керівник _____Зікратий С. В., к.т.н., доцент_____ (підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника) Допущено до захисту В. о. завідувача кафедри ІТТС к.т.н., доцент Заміховська О. Л. (посада) (підпис) (дата) (прізвище та ініціали) м. Івано-Франківськ – 2026 рік Івано-Франківський національний технічний університет нафти і газу Інститут Інформаційних технологій Кафедра Інформаційно-телекомунікаційних технологій і систем _____ Освітньо-кваліфікаційний рівень бакалавр _____ Спеціальність 126 – Інформаційні системи та технології

ЗАТВЕРДЖУЮ:

В. о. завідувача кафедри ІТТС Заміховська О.Л.
« » _____ 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Осадчук Валентині Юріївні
.
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка інформаційної системи для управління проєктами компанії
керівник проєкту (роботи) Зікратий С. В., к.т.н., доцент ,
(прізвище, ім'я, по батькові)
затверджені наказом вищого навчального закладу від
2. Строк подання студентом проєкту (роботи) червня 2026р.
3. Вихідні дані до роботи Методичні вказівки, технічна література .
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1 Аналіз предметної області та програмних аналогів; 2 Формування
вимог і проєктування інформаційної системи; 3 Розробка клієнтської та серверної частин вебзастосунку, реляційної бази даних і REST API; 4 Реалізація модулів управління проєктами, завданнями, обілку часу, аналітики та звітності; 5 Тестування програмного забезпечення.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
діаграма прецедентів (Use Case); функціональна модель IDEF0 (контекстна діаграма та декомпозиція); діаграми бізнес-процесів (BPMN) та потоків даних

(DFD/IDEF3); діаграма компонентів та діаграма розгортання системи; діаграма класів, діаграми послідовності та діяльності; ER-діаграма та логічна схема бази даних; Екранні форми інтерфейсу системи._____
6. Консультатни розділів роботи
Розділ Консультант Підпис, дата
1- 3 Зікратий С. В.

7. Дата видачі завдання – 22 грудня 2025 року КАЛЕНДАРНИЙ ПЛАН No з/п Назва етапів бакалаврської роботи
Термін виконання
етапів роботи
Примітка
1
Аналіз предметної області та постановка
завдання
Березень, 2026 виконано
2
Розробка структури інформаційної
системи
Квітень, 2026 виконано
3 Розробка програмного забезпечення Травень, 2026 виконано
4 Оформлення пояснювальної записки Червень, 2026 виконано

Студент Осадчук В. Ю.

Керівник роботи Зікратий. С. В.

АНОТАЦІЯ
Бакалаврська робота присвячена розробленню вебзорієнтованої інформаційної системи управління проєктами «ProSync IT». У роботі проаналізовано предметну область управління IT-проєктами за гнучкими методологіями, виконано порівняння наявних програмних аналогів та обґрунтовано доцільність створення власного рішення. Спроєктовано архітектуру системи, структуру реляційної бази даних і побудовано UML-моделі. Програмно реалізовано серверну частину з REST-API та рольовим розмежуванням доступу, а також клієнтський односторінковий застосунок із Kanban-дошкою, обліком трудовитрат, системою сповіщень, аналітикою та генерацією PDF-звітів. Працездатність системи підтверджено комплексним тестуванням.
Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, УПРАВЛІННЯ ПРОЄКТАМИ, SCRUM, KANBAN, ОБЛІК ЧАСУ, REST API, REACT, NODE.JS, EXPRESS, SEQUELIZE, MYSQL.

ANNOTATION
The bachelor's thesis is devoted to the development of the web-based project management information system "ProSync IT". The work analyses the domain of IT project management based on agile methodologies, compares existing software analogues and substantiates the feasibility of building a custom solution. The system architecture and the relational database structure are designed, and UML models are built. The server side is implemented as a REST API with role-based access control, and the client side is a single-page application featuring a Kanban board, time tracking, a notification subsystem, analytics and PDF report generation. The operability of the system is confirmed by comprehensive testing.
Keywords: INFORMATION SYSTEM, PROJECT MANAGEMENT, SCRUM, KANBAN, TIME TRACKING, REST API, REACT, NODE.JS, EXPRESS, SEQUELIZE, MYSQL.

КБР.ІСТ-13.00.00.000 ПЗ
Змн Лист No докум. Підпис Дата
Розроб. ОСАДЧУК В. Ю.

Розробка інформаційної системи для управління проєктами компанії
Літ. Арк. Аркушів
Перевір. ЗІКРАТИЙ С. В. Н 6 102
Реценз.
ІФНТУНГ,ІСТ-22-1 Н. Контр. ВОЗНИЙ **А. В.**
Затверд. ЗАМІХОВСЬКА О. Л.

ЗМІСТ

ВСТУП 9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 12
1.1 Аналіз предметної області та процесів управління IT-проєктами 12
1.2 Огляд програмних аналогів 13
1.3 Призначення, межі та цілі системи 16
1.4 Користувачі системи та розмежування прав доступу 17
1.5 Функціональні та нефункціональні вимоги 18
1.6 Постановка задачі 20
1.7 Висновки до розділу 1 ..... 21 2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА СТРУКТУРИ ІНФОРМАЦІЙНОЇ
СИСТЕМИ 23
2.1 Обґрунтування вибору технологічного стеку 23
2.2 Функціональне та процесне моделювання системи 25
2.3 Проєктування програмної архітектури 29
2.4 Об'єкто-орієнтоване моделювання 32
2.5 Проєктування реляційної бази даних 34
2.6 Висновки до розділу 2 38
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ
СИСТЕМИ «ProSync IT» 40
3.1 Структура та організація програмного коду 40
3.2 Реалізація серверної частини 41
3.3 Реалізація REST-API 45
3.4 Реалізація клієнтської частини 46

3.5 Реалізація ключових функціональних модулів 47
3.6 Тестування системи 53
3.7 Висновки до розділу 3 55

ВИСНОВКИ 56
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 58
ДОДАТОК А 61
ДОДАТОК Б 65
ДОДАТОК В 84
БІБЛІОГРАФІЧНА ДОВІДКА 102

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

API (Application Programming Interface) — програмний інтерфейс застосунку.
CRUD (Create, Read, Update, Delete) — створення, читання, оновлення, видалення.
CSS (Cascading Style Sheets) — каскадні таблиці стилів.
DnD (Drag-and-Drop) — механізм перетягування елементів інтерфейсу.
DTO (Data Transfer Object) — об’єкт передавання даних.
HTTP/HTTPS (HyperText Transfer Protocol (Secure)) — протокол передавання гіпертексту.
JSON (JavaScript Object Notation) — текстовий формат обміну даними.
JWT (JSON Web Token) — токен автентифікації.
KPI (Key Performance Indicator) — ключовий показник ефективності.
MVC (Model-View-Controller) — архітектурний патерн «модель–подання–контролер».
ORM (Object-Relational Mapping) — об’єктно-реляційне відображення.
REST (Representational State Transfer) — архітектурний стиль вебсервісів.
SPA (Single Page Application) — односторінковий застосунок.
SP (Story Points) — умовна оцінка складності завдання.
СКБД — Система керування базами даних.
ІС — Інформаційна система.
ПЗ — Програмне забезпечення.

ВСТУП

У сучасних ІТ-компаніях успішне виконання проєктів залежить не лише від професійних навичок команди, а й від ефективної організації робочих процесів. Навіть за наявності досвідчених розробників і тестувальників відсутність зручного інструменту для координації роботи може призводити до втрати часу, нерівномірного розподілу навантаження та труднощів із контролем виконання завдань. Особливо це помітно під час роботи над декількома проєктами одночасно, коли необхідно постійно відстежувати прогрес, контролювати терміни виконання та аналізувати використання ресурсів.

На практиці багато команд використовують набір окремих інструментів для ведення завдань, обміну повідомленнями, обліку робочого часу та формування звітності. Такий підхід створює додаткові труднощі, оскільки інформація зберігається в різних системах і потребує ручного узгодження. У результаті керівникам проєктів доводиться витрачати значну частину часу на пошук актуальних даних та підготовку звітів, а учасники команди не завжди мають повне уявлення про поточний стан робіт.

Популярні програмні продукти для управління проєктами, такі як Jira, Trello або Asana, пропонують широкий набір можливостей, однак їх використання не завжди є доцільним для невеликих команд. Частина функціоналу може залишатися незатребуваною, а деякі можливості доступні лише у платних версіях. Крім того, використання сторонніх сервісів не завжди

дозволяє повністю контролювати дані та адаптувати систему під особливості конкретного робочого процесу.
У зв'язку з цим актуальним є створення власної інформаційної системи, яка поєднуватиме основні інструменти управління проєктами в єдиному середовищі. Така система повинна забезпечувати планування проєктів і спринтів, роботу із завданнями на Kanban-дошці, облік витраченого часу,

інформування учасників команди про важливі зміни та формування аналітичної звітності.
Метою бакалаврської роботи є проєктування та розроблення вебзорієнтованої інформаційної системи управління проєктами «ProSync IT», яка дозволяє централізовано організовувати роботу команди, контролювати виконання завдань і здійснювати аналіз результатів діяльності в межах єдиного програмного середовища.
Об'єктом дослідження є процеси управління IT-проєктами та організації командної роботи під час розроблення програмного забезпечення.
Предметом дослідження є методи та засоби проєктування вебзорієнтованих інформаційних систем управління проєктами, а також технології реалізації клієнтської та серверної частин програмного забезпечення.
Для досягнення поставленої мети необхідно вирішити такі завдання:
− проаналізувати особливості управління IT-проєктами та дослідити існуючі програмні аналоги;
− визначити функціональні та нефункціональні вимоги до системи;
− спроєктувати архітектуру системи та структуру реляційної бази даних, побудувати UML-моделі;
− обґрунтувати вибір технологічного стеку та реалізувати серверну частину з REST-API і рольовим доступом;
− розробити клієнтський вебзастосунок із підтримкою Kanban-дошки, обліку часу, сповіщень, аналітики та генерації PDF-звітів;
− виконати тестування системи й перевірити відповідність реалізації поставленим вимогам.
Під час виконання роботи використано методи системного аналізу, об'єктно-орієнтованого проєктування, реляційного моделювання даних, а також сучасні підходи до розроблення вебзастосунків.
Практичне значення роботи полягає у створенні працездатної інформаційної системи, яку можна використовувати для управління

проєктами невеликих і середніх команд розробників програмного забезпечення. Запропоноване рішення забезпечує централізоване зберігання даних, підвищує прозорість виконання завдань та спрощує процес формування звітності.
Структура роботи складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатків. У першому розділі розглянуто предметну область та проведено аналіз існуючих рішень. Другий розділ присвячено проєктуванню архітектури системи та структури бази даних. У третьому розділі описано програмну реалізацію та результати тестування розробленої інформаційної системи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області та процесів управління IT-проєктами

Предметною областю роботи є управління проєктами у сфері розроблення програмного забезпечення — діяльність, що охоплює повний життєвий цикл продукту: від збору вимог та ініціації до релізу й супроводу.
Ключове завдання управління в такому середовищі полягає в раціональному

розподілі людських, часових і технічних ресурсів задля досягнення високої продуктивності команди та якісного результату у визначені терміни. На відміну від проєктів із заздалегідь незмінним обсягом робіт, розроблення ПЗ характеризується високою мінливістю вимог, що зумовлює поширення гнучких (Agile) підходів до організації праці.

Найпоширенішими гнучкими методологіями є фреймворк Scrum та підхід Kanban. Згідно з офіційним керівництвом, Scrum організовує роботу у вигляді коротких ітерацій — спринтів, упродовж яких команда виконує запланований обсяг завдань і демонструє придатний до використання приріст продукту [1]. Робота структурується навколо беклогу — упорядкованого за пріоритетом переліку завдань, із яких на кожен спринт відбирається посильний обсяг. Підхід Kanban, своєю чергою, робить акцент на безперервному потоці робіт і візуалізації стану завдань на дошці зі стовпцями-статусами, що дає змогу команді бачити, що зроблено, що в роботі, а що очікує на виконання. Спільним для обох підходів є прагнення до прозорості процесу, регулярного перегляду результатів і швидкого реагування на зміну пріоритетів [5, 6].

Системний аналіз діяльності команд розробки дає змогу виокремити кілька актуальних проблем менеджменту, наприклад низька прозорість розподілу завдань: керівник позбавлений можливості бачити реальну поточну завантаженість виконавців, унаслідок чого одні спеціалісти виявляються перевантаженими, тоді як інші мають надлишок вільного часу. Другою проблемою є відсутність достовірного обліку трудовитрат — без фіксації

фактично витраченого часу неможливо ані об’єктивно оцінити продуктивність, ані спланувати наступні ітерації. Третьою є висока трудомісткість формування звітності, коли менеджер вручну зводить дані з різних джерел, витрачаючи на це час, який міг би бути спрямований на управління. Також використання розрізнених інструментів (електронних таблиць, месенджерів, нотаток) призводить до децентралізації даних, їх дублювання та втрати, а отже — до зривання термінів.

Складність координації стрімко зростає відносно масштабу проєкту, кількості залучених спеціалістів та різнорідності виконуваних завдань. Усе це формує об’єктивну потребу в єдиному цифровому середовищі, яке поєднувало б планування, ведення завдань, облік часу та аналітику, мінімізуючи ручну працю та усуваючи інформаційні розриви між учасниками процесу.

1.2 Огляд програмних аналогів

Для формування об’єктивних вимог до проєктованої системи потрібно дослідити наявні програмні рішення, що є стандартами в галузі управління проєктами. Аналіз сильних і слабких сторін провідних платформ дозволяє запозичити вдалі інженерні рішення та уникнути характерних недоліків. Для розгляду відібрано три найбільш популярні системи, що уособлюють різні підходи: Jira, Trello та Asana.

Першим аналогом є Jira від компанії Atlassian — потужна платформа для гнучкого управління, орієнтована насамперед на команди розробки [2]. Система підтримує Scrum- і Kanban-дошки, ведення беклогу, планування спринтів, оцінку завдань у Story Points, побудову діаграм згоряння та різноманітні звіти, а також інтегрується з великою кількістю зовнішніх сервісів. Сильною стороною Jira є глибина функціоналу та гнучкість налаштування робочих процесів, але саме ця насиченість обертається складністю: платформа має високий поріг входу для нових користувачів, вимагає тривалого налаштування, а вартість ліцензування для великих команд є відчутною. Для невеликої команди значна частина можливостей Jira

залишається незатребуваною, що робить її надлишковою. Зовнішній вигляд інтерфейсу Jira наведено на рисунку 1.1.

Рисунок 1.1 — Інтерфейс системи Jira
Другим аналогом є Trello — інструмент на основі Kanban-дошок, що належить тій самій Atlassian [3]. Trello будується на простій трирівневій ієрархії: дошки містять списки, а списки — картки, які перетягуються між стовпцями за допомогою механізму drag-and-drop. Перевагою Trello є інтуїтивно зрозумілий інтерфейс і мінімальний поріг входу: команда може почати роботу за лічені години. Проте безкоштовна версія обмежена (зокрема кількістю дошок і лише Kanban-поданням), а самій системі бракує вбудованих засобів глибокої аналітики, повноцінного обліку часу та інструментів управління багаторівневими проєктами, через що вона краще пасує невеликим або особистим ініціативам, ніж систематичному управлінню розробкою. Інтерфейс системи Trello наведено на рисунку 1.2.

Рисунок 1.2 — Інтерфейс системи Trello
Третім аналогом є Asana — платформа управління роботою, що поєднує кілька подань завдань (список, дошка, часова шкала, календар) із побудовою інформаційних панелей на основі актуальних даних та інтеграцією з понад 400 сервісів [4]. Asana добре підходить для структурованої координації, оскільки впорядковує роботу за схемою «завдання → проєкти → портфелі». Її слабкими сторонами є обмеженість безкоштовної версії та надмірно активна система сповіщень, яка може відволікати виконавців від роботи. Приклад подання проєкту в Asana наведено на рисунку 1.3.

Рисунок 1.3 — Інтерфейс системи Asana
Узагальнення результатів аналізу показує, що жодне з розглянутих рішень не є оптимальним для невеликої IT-команди: Jira надмірно складна й дорога, Trello — занадто проста й слабка в аналітиці, Asana обмежує безкоштовний функціонал. Спільним недоліком комерційних платформ є залежність від зовнішнього постачальника, неможливість повного контролю над даними та надлишковість функцій, за які доводиться сплачувати. Це

обґрунтовує доцільність розроблення власної, відносно компактної системи, яка інтегрує найнеобхідніше — Kanban-дошку з перетягуванням завдань, планування спринтів, оцінку у Story Points, вбудований облік часу, рольове розмежування доступу, сповіщення та формування PDF-звітів. Зведене порівняння аналогів за ключовими критеріями наведено в таблиці 1.1.
Таблиця 1.1 — Порівняльний аналіз програмних аналогів

Критерій	Jira	Trello	Asana	ProSync IT
Kanban-дошка з drag-and-drop	+	+	+	+
Планування спринтів	+	-	частково	+
Оцінка у Story Points	+	-	частково	+
Вбудований облік часу	частково	-	-	+
Рольова модель доступу	+	частково	+	+
Аналітика та графіки	+	-	+	+
Експорт звітів у PDF	+	-	частково	+
Власне розгортання / контроль даних	-	-	-	+
Поріг входу	високий	низький	середній	низький
Вартість для команди	висока	помірна	помірна	відсутня

1.3 Призначення, межі та цілі системи
Проєктована інформаційна система отримала робочу назву «ProSync IT». Її головним призначенням є створення єдиного інтегрованого цифрового

простору для планування, поточного моніторингу та аналізу виконання завдань IT-проєктів. Система реалізується як веб-додаток, доступний через браузер, що знімає потребу у встановленні клієнтського програмного забезпечення та спрощує спільну роботу команди.
Бізнес-мета впровадження системи полягає у скороченні витрат часу менеджерів на формування аналітичної звітності приблизно на 40 % та зменшенні частки прострочених завдань до показника, що не перевищує 5 % від загального обсягу беклогу за спринт. Межі системи охоплюють внутрішнє

управління проєктами, контроль робочого часу та інформування учасників і замовників; натомість поза межами залишаються модулі бухгалтерського обліку, нарахування заробітної плати та кадрового діловодства. Узагальнені характеристики системи подано у вигляді паспорта (таблиця 1.2).
Таблиця 1.2 — Паспорт інформаційної системи «ProSync IT»
Характеристика Опис
Назва системи Вебзорієнтована інформаційна система управління проєктами «ProSync IT»
Призначення Єдиний цифровий простір для планування проєктів і спринтів, ведення завдань, обліку часу, сповіщення та аналізу виконання робіт
Бізнес-мета та метрики Скорочення часу на звітність менеджерів ≈ на 40 %; зменшення частки прострочених завдань до ≤ 5 % за спринт
Межі (охоплює) Управління проєктами та завданнями, облік робочого часу, сповіщення, аналітика, генерація PDF-звітів
Межі (не охоплює) Бухгалтерський облік, нарахування зарплати, кадрове діловодство
Архітектура Багаторівнева клієнт-серверна, REST-взаємодія у форматі JSON
Платформа Вебзастосунок (браузер), серверна частина на Node.js, СКБД MySQL

1.4 Користувачі системи та розмежування прав доступу
Взаємодія із системою відбувається через рольову модель, що відображає реальний розподіл обов'язків у команді. Відповідно до фактичної реалізації, у системі передбачено п'ять ролей: системний адміністратор, менеджер проєкту, розробник, інженер з тестування (QA) та зовнішній замовник (клієнт). Обов'язковою стартовою точкою для будь-якого користувача є автентифікація; подальший доступ до операцій визначається роллю та перевіряється на стороні сервера.

Системний адміністратор відповідає за технічне обслуговування платформи та керування обліковими записами. Менеджер проєкту створює проєкти, формує спринти й беклог, призначає виконавців на завдання та ініціює формування звітності; саме менеджеру (разом з адміністратором) надано право створювати, змінювати й видаляти проєкти, спринти та завдання. Розробник та інженер з тестування є безпосередніми виконавцями: вони переглядають призначені завдання, змінюють їхній статус на Kanban-дошці шляхом перетягування та фіксують витрачений робочий час. Зовнішній замовник має обмежений доступ, орієнтований на спостереження за прогресом виконання робіт без можливості втручання у внутрішні процеси команди. Розподіл повноважень за ролями подано в таблиці 1.3.

Таблиця 1.3 — Ролі користувачів та їхні повноваження
Роль Зона відповідальності

Ключові повноваження	
Адміністратор (admin)	Технічне обслуговування, конфігурація
	Керування обліковими записами; повний доступ до управління проєктами, спринтами, завданнями
Менеджер проєкту (manager)	Планування та координація
	Створення/редагування/видалення проєктів, спринтів і завдань; генерація звітів; перегляд аналітики
Розробник (developer)	Виконання завдань
	Перегляд завдань, зміна статусу на дошці, облік часу, коментування
Інженер з тестування (qa)	Тестування завдань
	Перегляд завдань, зміна статусу на дошці, облік часу, коментування
Замовник (client)	Спостереження за прогресом
	Перегляд стану виконання проєкту (обмежений доступ)

1.5 Функціональні та нефункціональні вимоги

На основі аналізу предметної області, програмних аналогів і ролей користувачів сформовано перелік функціональних вимог до системи.

Розроблюване програмне забезпечення повинно забезпечувати автентифікацію користувачів та розмежування доступу відповідно до ролей, підтримувати управління проєктами, спринтами й завданнями, надавати можливість обліку витраченого часу та коментування виконаних робіт. Окрему увагу приділено візуалізації процесу виконання завдань за допомогою Kanban-дошки, автоматичному інформуванню користувачів про зміни в системі, формуванню аналітичних показників та генерації підсумкових звітів у форматі PDF. Деталізований перелік функціональних можливостей системи наведено в таблиці 1.4.

Таблиця 1.4 — Функціональні можливості системи

Модуль	Опис функціональності
Автентифікація та доступ	Вхід за email і паролем, видача токена доступу, перевірка ролі для захищених операцій
Проєкти	Створення, редагування, видалення проєктів; статуси проєкту та бюджет часу
Спринти	Формування спринтів у межах проєкту з визначенням часових рамок і статусу
Завдання	Ведення завдань: назва, опис, пріоритет, Story Points, виконавець, дедлайн, статус
Kanban-дошка	Подання завдань за стовпцями статусів; зміна статусу

перетягуванням карток
Облік часу Фіксація витрачених годин, дати та коментаря до виконаної роботи
Коментарі Додавання та видалення коментарів до завдань (із перевіркою прав)
Сповіщення Автоматичне інформування виконавця про зміну статусу його завдання
Аналітика KPI-показники, стовпчикова діаграма завантаженості, кругова діаграма статусів, індикатор прогресу
Звіти Генерація PDF-звіту з інформацією про проєкт, статистикою завдань, часом і спринтами

До нефункціональних вимог належать реалізація системи у вигляді вебзастосунку з односторінковим клієнтом, взаємодія клієнтської та серверної частин за архітектурним стилем REST із використанням формату JSON,

застосування реляційної СКБД MySQL для зберігання даних, а також забезпечення безпеки шляхом використання криптографічного хешування паролів, токенної автентифікації та налаштування безпекових механізмів сервера. Важливими вимогами також є супроводжуваність програмного коду завдяки використанню архітектурного підходу MVC, масштабованість системи та можливість подальшого розширення функціональності без істотної зміни її структури.
Для формалізації взаємодії користувачів із системою використано діаграму прецедентів (Use Case Diagram), яка відображає основні сценарії використання програмного продукту та їх розподіл між акторами. На діаграмі представлені прецеденти автентифікації користувачів, управління проєктами, спринтами та завданнями, обліку робочого часу, формування звітності та перегляду аналітичної інформації. Діаграму прецедентів наведено на рисунку 1.4.

Рисунок 1.4 — Діаграма прецедентів (Use Case) ІС «ProSync IT»

1.6 Постановка задачі

З урахуванням проведеного аналізу предметної області, програмних аналогів та сформованих вимог завдання роботи полягає у проєктуванні,

програмній реалізації та тестуванні вебзорієнтованої інформаційної системи управління проєктами «ProSync IT» із багаторівневою клієнт-серверною архітектурою.
У межах роботи необхідно: спроєктувати архітектуру системи з відокремленням клієнтської та серверної частин і взаємодією через REST-API; розробити структуру реляційної бази даних та побудувати UML-моделі (діаграми прецедентів, класів, компонентів, розгортання, послідовності та діяльності); реалізувати серверну частину на платформі Node.js із фреймворком Express, доступом до даних через ORM Sequelize поверх СКБД MySQL, рольовим розмежуванням доступу на основі токенів JWT та незворотним хешуванням паролів; реалізувати клієнтський односторінковий застосунок на бібліотеці React зі збирачем Vite, що включає Kanban-дошку з механізмом перетягування завдань, модуль обліку часу, систему сповіщень, аналітичну панель із графіками та формування PDF-звітів; виконати тестування системи із застосуванням автоматизованих засобів (Jest, Supertest) і ручних функціональних перевірок.
Обґрунтований у роботі технологічний стек охоплює: на боці клієнта — React, збирач Vite, бібліотеку маршрутизації, бібліотеку drag-and-drop для Kanban-дошки та бібліотеку побудови діаграм; на боці сервера — Node.js, Express, ORM Sequelize, драйвер MySQL, засоби автентифікації (JWT) і хешування (bcrypt), бібліотеку генерації PDF та безпекові посередники. Детальне обґрунтування вибору технологій і проєктні рішення наведено в

розділі 2.

1.7 Висновки до розділу 1

У розділі проведено аналіз предметної області управління IT-проєктами та визначено основні проблеми, що виникають під час організації командної роботи, зокрема недостатню прозорість розподілу навантаження, складність обліку робочого часу, трудомісткість формування звітності та фрагментованість даних. Виконано порівняльний аналіз сучасних програмних

рішень для управління проєктами, результати якого дали змогу визначити їх переваги та обмеження з погляду потреб невеликих команд розробників. На основі проведеного дослідження сформульовано призначення, цілі та межі інформаційної системи «ProSync IT», визначено ролі користувачів і розподіл прав доступу. Також сформовано перелік функціональних і нефункціональних вимог до програмного продукту, побудовано моделі взаємодії користувачів із системою та обґрунтовано вибір технологічного стеку для реалізації проєкту. Отримані результати є основою для подальшого проєктування архітектури та реалізації системи, що розглядаються в наступному розділі.

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА СТРУКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

На основі вимог, сформульованих у першому розділі, у цьому розділі виконано інженерне проєктування системи «ProSync IT»: обґрунтовано технологічний стек, побудовано функціональні та процесні моделі, спроєктовано програмну архітектуру, об'єктно-орієнтовані моделі та структуру реляційної бази даних. Наскрізне проєктування трансформує абстрактні бізнес-вимоги у чіткі технічні специфікації, придатні до безпосередньої програмної реалізації.

2.1 Обґрунтування вибору технологічного стеку

Вибір інструментальних засобів зумовлений вимогами до вебзорієнтованості системи, потребою у чіткому розділенні клієнтської та серверної частин, а також прагненням використати поширені, добре задокументовані технології з відкритим кодом. Систему побудовано за принципом єдиної мови програмування для всіх рівнів: і клієнт, і сервер реалізовано мовою JavaScript, що спрощує розроблення та супровід. Клієнтську частину реалізовано на бібліотеці React [7] — компонентній бібліотеці для побудови інтерфейсів користувача, що використовує віртуальну модель документа для ефективного оновлення подання та декларативний підхід до опису компонентів. Складання та запуск клієнта здійснюється засобом Vite [8], який забезпечує швидке інкрементне оновлення під час розроблення та оптимізовану збірку для продакшену. Навігацію між сторінками односторінкового застосунку реалізовано бібліотекою react-router-dom, а взаємодію з сервером — HTTP-клієнтом axios, що дозволяє централізовано додавати токен автентифікації до запитів. Для побудови Kanban-дошки обрано бібліотеку drag-and-drop @dnd-kit [9], яка є легкою, продуктивною та доступною (з підтримкою клавіатури й допоміжних

технологій), а для візуалізації аналітики — бібліотеку декларативних діаграм Recharts [10].

Серверну частину реалізовано на платформі Node.js [11] із вебфреймворком Express [12], що забезпечує лаконічну маршрутизацію запитів і конвеєр проміжного програмного забезпечення (middleware). Доступ

до бази даних організовано через об'єктно-реляційне відображення Sequelize [13], яке відображає таблиці на класи-моделі, підтримує міграції та зв'язки між сутностями, а також знижує ризик SQL-ін'єкцій завдяки параметризації запитів. Взаємодію з MySQL забезпечує драйвер mysql2. Підсистему безпеки побудовано на бібліотеці jsonwebtoken для роботи з токенами JWT [14], бібліотеці bcrypt для незворотного хешування паролів [15], захисних HTTP-заголовках helmet та контролі крос-доменних запитів cors; журналювання запитів виконує morgan, валідацію вхідних даних — express-validator. Формування PDF-звітів реалізовано бібліотекою PDFKit [16], а конфіденційні параметри конфігурації виносяться в середовище за допомогою dotenv.
Як систему керування базами даних обрано реляційну СКБД MySQL [17] із рушієм InnoDB, що забезпечує повну відповідність вимогам ACID, підтримку транзакцій, зовнішніх ключів та механізмів відновлення після збоїв. Реляційна модель є оптимальною для предметної області, у якій дані мають чітко визначену структуру та численні зв'язки між сутностями (проєкти, спринти, завдання, користувачі). Зведені відомості про технологічний стек наведено в таблиці 2.1. Таблиця 2.1 — Технології розроблення системи

Рівень	Технологія	Версія	Призначення
Клієнт	React, React DOM	19.2	Побудова компонентного інтерфейсу користувача
Клієнт	Vite	8.0	Складання та середовище розроблення клієнта
Клієнт	react-router-dom	7.17	Маршрутизація сторінок SPA
Клієнт	@dnd-kit/core, sortable	6.3 / 10.0	Механізм Drag-and-Drop для Kanban-дошки
Клієнт	Recharts	3.8	Побудова аналітичних діаграм
Клієнт	axios	1.17	HTTP-клієнт, передавання токена доступу
Сервер	mysql2	3.22	Драйвер з'єднання з MySQL
Сервер	jsonwebtoken	9.0	Автентифікація на основі токенів JWT
Сервер	bcrypt	6.0	Хешування паролів
Сервер	helmet, cors, morgan	8.2 / 2.8 / 1.11	Безпека, крос-доменні запити, журналювання
Сервер	express-validator	7.3	Валідація вхідних даних
Сервер	PDFKit	0.18	Генерація PDF-звітів
Дані	MySQL (InnoDB)	8.x	Реляційне сховище даних
Тестування	Jest, Supertest	30 / 7	Автоматизоване тестування серверної частини

2.2 Функціональне та процесне моделювання системи

Для формалізації процесів управління проєктами застосовано методологію функціонального моделювання IDEF0. На найвищому рівні абстракції систему подано контекстною діаграмою A-0 з єдиним функціональним блоком «Керувати проєктами компанії», взаємодію якого із зовнішнім середовищем описано за правилом ICOM (вхідні дані, керування, механізми, виходи). Вхідними даними є первинні вимоги замовника та доступні ресурси компанії; керуванням — методологія розробки, договірні зобов'язання та внутрішні стандарти якості; механізмами — менеджер проєкту, команда розробки й тестування та сама система «ProSync IT»; виходами — готовий програмний продукт, підсумкова звітність і статистика

закриття завдань. Контекстну діаграму наведено на рисунку 2.1.

Рисунок 2.1 — Контекстна діаграма IDEF0 (A-0)

Декомпозиція контекстного блоку дозволила виокремити чотири послідовні підпроцеси: ініціацію та планування проєкту, розподіл завдань і спринтів, моніторинг і виконання робіт, а також генерацію звітності й аналіз результатів. Діаграму декомпозиції наведено на рисунку 2.2.

Рисунок 2.2 — Діаграма декомпозиції IDEF0 (A0)

Для опису руху інформаційних масивів побудовано контекстну діаграму потоків даних (DFD) у нотації Гейна–Сарсона (рисунок 2.3), яка фіксує зовнішні сутності (менеджер, виконавець, замовник, адміністратор) та потоки даних між ними і центральним процесом. Динаміку технологічного процесу у часі описано діаграмою IDEF3 (рисунок 2.4): від авторизації та перевірки прав доступу через паралельні процедури фіксації трудовитрат і оновлення статусу завдання до генерації звіту й надсилання сповіщень.

Рисунок 2.3 — Контекстна діаграма потоків даних (DFD)

Рисунок 2.4 — Діаграма IDEF3 процесу роботи із системою

Для перевірки узгодженості динамічних процесів зі структурою бази даних сформовано матрицю трасування кроків IDEF3 до сутностей реляційної моделі (таблиця 2.2), яка демонструє, які сутності читаються та які створюються чи оновлюються на кожному кроці.

Таблиця 2.2 — Матриця трасування процесів IDEF3 до сутностей бази даних

Крок IDEF3	Сутність читається	Сутність створюється/оновлюється
Авторизувати користувача	users	—
Перевірити права доступу	roles	—
Зчитати дані беклогу	projects, tasks	—
Зафіксувати трудовитрати	tasks	time_logs
Оновити статус завдання	tasks	tasks
Згенерувати підсумковий звіт	projects, time_logs	reports
Надіслати сповіщення	users	notifications

Алгоритми виконання повсякденних завдань формалізовано засобами нотації BPMN 2.0. Розроблено три моделі різного рівня складності: базовий процес створення завдання в беклозі (рисунок 2.5), процес планування спринту з розподілом ролей між менеджером і виконавцем (рисунок 2.6) та міжорганізаційний процес перевірки й погодження результатів ітерації за участю замовника (рисунок 2.7).

Рисунок 2.5 — Діаграма бізнес-процесу створення завдання в беклозі

Рисунок 2.6 — Діаграма бізнес-процесу планування спринту

Рисунок 2.7 — Діаграма бізнес-процесу погодження результатів спринту

2.3 Проєктування програмної архітектури

Систему «ProSync IT» побудовано за багаторівневою клієнт-серверною архітектурою з чітким логічним і фізичним розділенням на клієнтську частину (Frontend), серверну частину (Backend) та рівень збереження даних. Таке розділення дозволяє незалежно розвивати інтерфейс і бізнес-логіку, спрощує

тестування та відкриває можливість у майбутньому під'єднати до наявного серверного ядра інші клієнти (наприклад, мобільний застосунок) без зміни логіки оброблення даних.

Клієнтська частина реалізована як односторінковий застосунок (SPA): браузер завантажує зібраний код один раз, після чого взаємодіє із сервером, обмінюючись винятково структурованими даними у форматі JSON. Сервер не генерує HTML-сторінки, а виступає постачальником даних через REST-API, доступне за протоколом HTTP(S). Такий підхід зменшує навантаження на мережу та забезпечує плавну взаємодію користувача з інтерфейсом.

Внутрішню структуру серверної частини організовано за патерном MVC із додатковим сервісним шаром. Маршрути (routes) визначають точки входу REST-API та зіставляють їх з обробниками; контролери (controllers) приймають запити, видобувають параметри й формують відповіді; сервіси (services) інкапсулюють бізнес-логіку (зокрема перевірку прав, формування сповіщень, агрегацію статистики); моделі (models) відповідають за відображення на таблиці бази даних через ORM Sequelize. Запит послідовно проходить конвеєр проміжного програмного забезпечення: встановлення захисних HTTP-заголовків (helmet), дозвіл крос-доменних запитів (cors), журналювання (morgan), розбір тіла запиту у форматі JSON, після чого передається відповідному маршрутизатору; захищені маршрути додатково проходять посередник автентифікації (перевірка токена JWT) та, за потреби, посередник авторизації (перевірка ролі). Помилки обробляються централізованим посередником, який формує уніфіковану відповідь із відповідним кодом стану.

Свідомо обрано архітектуру модульного моноліту, а не мікросервісів: для поточних меж проєкту та розміру команди мікросервіси створили б невиправдане навантаження на інфраструктуру й ускладнили б підтримку цілісності транзакцій. Структурні елементи програмного забезпечення та статичні зв'язки між ними відображено на діаграмі компонентів (рисунок 2.8),

а топологію апаратних засобів і середовищ виконання — на діаграмі розгортання (рисунок 2.9).

Рисунок 2.8 — Діаграма компонентів системи «ProSync IT»

Рисунок 2.9 — Діаграма розгортання системи «ProSync IT»

Згідно з діаграмою розгортання, систему побудовано на трьох вузлах: робоча станція користувача (браузер із завантаженим клієнтським застосунком), сервер застосунків (середовище Node.js із серверним кодом) та сервер бази даних (СКБД MySQL). Робоча станція взаємодіє із сервером

застосунків через мережу за захищеним протоколом HTTP(S), тоді як сервер застосунків звертається до бази даних у межах внутрішньої підмережі за протоколом TCP/IP, що захищає сховище від прямого доступу ззовні.

2.4 Об’єктно-орієнтоване моделювання

Статичну структуру системи на рівні програмного коду відображено діаграмою класів (рисунок 2.10), яка ідентифікує ключові сутності предметної області (User, Role, Project, Sprint, Task, TimeLog, Comment, Notification, Report), їхні атрибути, методи та типи зв’язків. Діаграма класів узгоджена зі структурою реляційної бази даних і слугує містком між об’єктною моделлю програмного коду та фізичним сховищем даних.

Рисунок 2.10 — Діаграма класів системи «ProSync IT»

Динаміку взаємодії об’єктів у часі описано діаграмою послідовності (рисунок 2.11) для одного з ключових сценаріїв — фіксації робочого часу та зміни статусу завдання. У сценарії беруть участь актор (виконавець), інтерфейс користувача (Kanban-дошка), контролер завдань та база даних: виконавець переміщує картку завдання, інтерфейс надсилає запит до сервера, контролер перевіряє дані та оновлює стан завдання, після чого база даних повертає підтвердження, а інтерфейс відображає оновлений стан.

Рисунок 2.11 — Діаграма послідовності сценарію «Зміна статусу завдання»

Для опису потоку керування під час однієї з найхарактерніших операцій системи додатково побудовано діаграму діяльності (Activity Diagram), що відображає фактичну логіку обробки зміни статусу завдання на сервері (рисунок 2.12). Процес розпочинається з перетягування картки користувачем, після чого клієнт надсилає запит на зміну статусу. Сервер перевіряє токен доступу: за невалідного токена повертається відмова з кодом 401; за валідного — здійснюється пошук завдання. Якщо завдання не знайдено, повертається код 404; інакше статус оновлюється, і за наявності призначеного виконавця автоматично створюється сповіщення. Завершальним кроком є повернення оновленого завдання клієнту та оновлення Kanban-дошки.

Рисунок 2.12 — Діаграма діяльності процесу зміни статусу завдання

2.5 Проєктування реляційної бази даних

2.5.1 Концептуальна модель

На концептуальному рівні предметну область подано у вигляді сукупності сутностей та зв’язків між ними, незалежно від конкретної СКБД. Порівняно з первинним проєктом, реалізована база даних розширена та містить дев’ять сутностей: ролі користувачів (roles), облікові записи (users), проєкти (projects), спринти (sprints), завдання (tasks), журнал обліку часу (time_logs), коментарі до завдань (comments), сповіщення (notifications) та звіти (reports).

Логіка взаємодії сутностей підпорядкована бізнес-правилам предметної області. Кожному користувачеві призначається одна роль, тоді як одну роль може мати багато користувачів (зв’язок «один-до-багатьох»). Проєкт має

менеджера-власника (зв’язок з користувачем) і композиційно об’єднує спринти: кожен спринт існує лише в межах батьківського проєкту. Спринт, своєю чергою, містить множину завдань. Кожне завдання може мати призначеного виконавця (зв’язок із користувачем, що допускає порожнє значення, якщо задача перебуває в загальному беклозі без призначення). Журнал обліку часу пов’язує користувача та завдання, фіксуючи витрачені години; коментар також належить одночасно завданню та автору. Сповіщення адресується конкретному користувачеві, а звіт пов’язаний із проєктом та користувачем, який його сформував. Концептуальну ER-діаграму бази даних наведено на рисунку 2.13.

Рисунок 2.13 — Концептуальна ER-діаграма бази даних «ProSync IT»

2.5.2 Логічна та фізична модель

На логічному рівні концептуальні зв’язки трансформовано у табличну структуру з визначенням назв полів, типів даних, обмежень на порожні

значення та ключів. Для кожної таблиці використано сурогатний первинний ключ — ціле число з автоінкрементом, що спрощує індексацію та пришвидшує пошук. Усі таблиці мають службові поля створення та оновлення запису (created_at, updated_at). Базу даних та всі таблиці налаштовано на набір символів utf8mb4 з порівнянням utf8mb4_unicode_ci, що гарантує коректне зберігання української та іншомовної текстової інформації. Узагальнену структуру таблиць наведено в таблиці 2.3, а перелічувані типи окремих полів — у таблиці 2.4.

Таблиця 2.3 — Структура таблиць бази даних

Таблиця	Призначення	Основні поля та ключі
roles	Довідник ролей доступу	id (PK), role_name (unique)
users	Облікові записи користувачів	id (PK), name, email (unique), password_hash, is_active, role_id (FK→roles)
projects	Проєкти компанії	id (PK), title, description, time_budget, start_date, end_date, status, manager_id (FK→users)
sprints	Спринти (ітерації) проєкту	id (PK), sprint_name, start_date, end_date, status, project_id (FK→projects)
tasks	Завдання спринту	id (PK), title, description, priority, story_points, status, deadline, sprint_id (FK→sprints), executor_id (FK→users, NULL)
time_logs	Журнал обліку робочого часу	id (PK), hours, log_date, comment, task_id (FK→tasks), user_id (FK→users)
comments	Коментарі до завдань	id (PK), content, task_id (FK→tasks), user_id (FK→users)
notifications	Сповіщення користувачів	id (PK), title, message, is_read, user_id (FK→users)
reports	Згенеровані звіти	id (PK), report_type, file_path, project_id (FK→projects), generated_by (FK→users)

Таблиця 2.4 — Перелічувані типи (ENUM) полів

Таблиця.поле	Допустимі значення	Значення за замовчуванням
tasks.priority	low, medium, high, critical	medium
tasks.status	backlog, todo, in_progress, testing, done	backlog
projects.status	planning, active, completed, archived	planning
sprints.status	planned, active, completed	planned

Особливу увагу приділено забезпеченню посилальної цілісності даних під час видалення та оновлення записів. Для залежних сутностей (спринти, завдання, журнали часу, коментарі, сповіщення та звіти) використано правило ON DELETE CASCADE, яке автоматично видаляє пов'язані записи разом із батьківським об'єктом. Для зв'язків, що забезпечують коректність облікових даних, застосовано правило ON DELETE RESTRICT, яке забороняє видалення користувачів або ролей за наявності залежних записів. Для поля виконавця завдання використано правило ON DELETE SET NULL, завдяки чому при видаленні користувача завдання зберігається, але втрачає прив'язку до виконавця. Логічну схему бази даних наведено на рисунку 2.14.

Рисунок 2.14 — Логічна схема реляційної бази даних «ProSync IT»

2.5.3 Аналіз на відповідність нормальним формам

Розроблену послідовно перевірено на відповідність першим трьом нормальним формам. Перша нормальна форма виконується, оскільки всі таблиці двовимірні, а кожен атрибут містить атомарне скалярне значення без повторюваних груп чи списків у межах одного поля (наприклад, завдання та сповіщення винесені в окремі пов'язані рядки, а не зберігаються переліком). Крім того, для кожної сутності визначено унікальний первинний ключ, що забезпечує однозначну ідентифікацію записів та виключає дублювання даних. Друга нормальна форма виконується автоматично: усі таблиці мають прості сурогатні первинні ключі, тож часткова залежність неключових атрибутів від частини ключа неможлива за визначенням. Усі описові атрибути залежать від первинного ключа відповідної таблиці та характеризують лише конкретний екземпляр сутності. Третя нормальна форма забезпечується відсутністю транзитивних залежностей — суміжні дані винесено в окремі довідкові таблиці. Зокрема, у таблиці користувачів зберігається лише ідентифікатор ролі, а її назва — в окремій сутності roles; це усуває аномалії оновлення, адже зміна назви ролі не потребує модифікації записів користувачів. Аналогічно, атрибути завдань залежать суто від ідентифікатора завдання, а дані про спринти та проєкти зчитуються опосередковано через зовнішні ключі. Таким чином, структура бази даних відповідає вимогам третьої нормальної форми, що гарантує її несуперечливість, зменшує надлишковість даних та забезпечує ефективне використання дискового простору.

2.6 Висновки до розділу 2

У розділі обґрунтовано технологічний стек системи «ProSync IT» та подано його у зведеній таблиці. Засобами IDEF0, DFD, IDEF3 і BPMN 2.0 виконано функціональне та процесне моделювання, а матриця трасування підтвердила узгодженість процесів зі структурою даних. Спроєктовано

багаторівневу клієнт-серверну архітектуру з патерном MVC і сервісним шаром, конвеєром проміжного програмного забезпечення та REST-

взаємодією, що відображено діаграмами компонентів і розгортання. Побудовано об’єктно-орієнтовані моделі (діаграми класів, послідовності та діяльності), які деталізують взаємодію користувачів і програмних компонентів системи. Окрему увагу приділено моделюванню процесу зміни статусу завдання та механізму автоматичного формування сповіщень. Спроєктовано реляційну базу даних із дев’яти таблиць, визначено типи полів, перелічувані значення та правила посилальної цілісності, а також підтверджено відповідність структури третій нормальній формі. Отримані результати формують завершену проєктну модель інформаційної системи та є підґрунтям для програмної реалізації й тестування системи, описаних у наступному розділі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ProSync IT»

У третьому розділі наведено результати програмної реалізації інформаційної системи «ProSync IT», спроєктованої в попередньому розділі. Розглянуто структуру клієнтської та серверної частин застосунку, реалізацію основних функціональних модулів, механізмів взаємодії через REST API та засобів роботи з базою даних. Завершальним етапом розділу є перевірка працездатності системи за допомогою автоматизованого й ручного тестування.

3.1 Структура та організація програмного коду

Програмний код системи організовано у вигляді двох незалежних частин, розміщених у каталогах client та server. Клієнтська частина побудована на React зі складанням засобом Vite; серверна — на Node.js і фреймворку Express. Така структура відповідає обраній клієнт-серверній архітектурі та дозволяє розгортати й розвивати обидві частини окремо.

Серверну частину організовано за рівнями відповідно до патерну MVC із сервісним шаром. Каталог server/src містить підкаталоги: config (налаштування з’єднання з базою даних), models (моделі Sequelize та їхні зв’язки), controllers (обробники запитів), services (бізнес-логіка), routes (маршрути REST-API) та middleware (проміжне програмне забезпечення). Окремо розміщено каталоги migrations і seeders (керування схемою та демонстраційними даними через Sequelize CLI), reports (згенеровані PDF-файли) та fonts (шрифт DejaVuSans для коректного відображення кирилиці у звітах). Клієнтську частину в каталозі client/src структуровано за призначенням: api (налаштування HTTP-клієнта), context (контекст автентифікації), routes (захищений маршрут), pages (екранні сторінки) та components (повторно використовувані компоненти). Структуру каталогів проєкту наведено на рисунку 3.1.

Рисунок 3.1 – Структура каталогів клієнтської та серверної частин ІС «ProSync IT»

Конфіденційні параметри конфігурації (реквізити з’єднання з базою даних DB_HOST, DB_NAME, DB_USER, DB_PASSWORD, порт сервера PORT та секретний ключ JWT_SECRET) винесено у файл середовища .env, який не потрапляє до системи контролю версій. З’єднання з базою даних встановлюється через Sequelize з увімкненим режимом underscored (поля у стилі snake_case), що відповідає іменуванню стовпців у базі.

3.2 Реалізація серверної частини

3.2.1 Рівень моделей даних

Кожну з дев’яти таблиць бази даних відображено окремою моделлю Sequelize, яка визначає атрибути, їхні типи та обмеження й автоматично супроводжується службовими полями created_at і updated_at. Поля priority та status моделі завдання реалізовано як перелічувані типи (ENUM), що змушує модель відхиляти некоректні значення ще до звернення до СКБД. Зв’язки між сутностями централізовано оголошено в модулі models/index.js за допомогою методів hasMany та belongsTo, зокрема з псевдонімами manager (менеджер

проєкту) та executor (виконавець завдання). Фрагмент моделі завдання з перелічуваними типами наведено нижче.

Лістинг 3.1 – Фрагмент моделі Task

```
const Task = sequelize.define('Task', {
  title:        { type: DataTypes.STRING(150), allowNull: false },
  priority:     { type: DataTypes.ENUM('low','medium','high','critical'),
                  allowNull: false, defaultValue: 'medium' },
  story_points: { type: DataTypes.INTEGER },
  status:       { type:
DataTypes.ENUM('backlog','todo','in_progress','testing','done'),
                  allowNull: false, defaultValue: 'backlog' },
  deadline:     { type: DataTypes.DATEONLY },
  sprint_id:    { type: DataTypes.INTEGER, allowNull: false },
  executor_id:  { type: DataTypes.INTEGER }
}, { tableName: 'tasks', timestamps: true,
     createdAt: 'created_at', updatedAt: 'updated_at' });
```

3.2.2 Проміжне програмне забезпечення

Кожен запит до сервера проходить конвеєр проміжного програмного забезпечення. Посередник автентифікації authenticate зчитує заголовок Authorization, перевіряє наявність токена у форматі Bearer, виконує його перевірку за допомогою секретного ключа й у разі успіху записує дані користувача (ідентифікатор і роль) до об’єкта запиту; за відсутності або недійсності токена повертається відповідь зі статусом 401. Посередник авторизації authorize приймає перелік дозволених ролей і блокує доступ зі статусом 403, якщо роль користувача до нього не належить. Валідацію вхідних даних завдання виконує посередник на основі бібліотеки express-validator: він перевіряє обов’язковість і довжину назви, належність пріоритету та статусу до допустимих значень, невід’ємність оцінки складності й обов’язковість спринту під час створення; за наявності помилок повертається статус 400 з їх переліком. Усі неперехоплені помилки обробляє централізований посередник, який формує уніфіковану відповідь із відповідним кодом стану. Реалізацію посередників автентифікації та авторизації наведено нижче.

Лістинг 3.2 – Реалізація автентифікації та рольового контролю доступу

```
const authenticate = (req, res, next) =&gt; {
  const authHeader = req.headers.authorization;
  if (!authHeader || !authHeader.startsWith('Bearer '))
    return res.status(401).json({ message: 'Потрібен токен' });
  try {
    req.user = jwt.verify(authHeader.split(' ')[1], process.env.JWT_SECRET);
    next();
  } catch { return res.status(401).json({ message: 'Недійсний токен' }); }
};

const authorize = (...roles) =&gt; (req, res, next) =&gt;
  roles.includes(req.user.role)
    ? next()
    : res.status(403).json({ message: 'Доступ заборонено' });
```

3.2.3 Рівень сервісів і контролерів

Контролери реалізовано «тонкими»: вони видобувають параметри запиту, викликають відповідний метод сервісу й формують відповідь, повертаючи коректний код стану в разі помилки. Уся змістовна логіка зосереджена в сервісах. Так, сервіс автентифікації відшукує користувача за електронною поштою разом із його роллю, звіряє пароль із збереженим хешем за допомогою bcrypt і, у разі успіху, формує токен JWT з ідентифікатором і роллю користувача та терміном дії вісім годин, повертаючи токен і дані профілю.

Лістинг 3.3 – Реалізація входу користувача та генерації JWT-токена

```
const login = async (email, password) =&gt; { const user = await User.findOne({ where: { email }, include: Role });
  if (!user) throw new Error('Невірний email або пароль');
```

```
  const ok = await bcrypt.compare(password, user.password_hash); if (!ok) throw new Error('Невірний email або пароль');
  const token = jwt.sign({ id: user.id, role: user.Role.role_name },
    process.env.JWT_SECRET, { expiresIn: '8h' });
  return { token, user: { id: user.id, name: user.name,
           email: user.email, role: user.Role.role_name } };
};
```


Прикладом перенесення бізнес-логіки на сервер є зміна статусу завдання. Відповідний сервіс не лише оновлює статус, а й — за наявності призначеного виконавця — автоматично створює запис-сповіщення, що гарантує надійне інформування незалежно від клієнта.

Лістинг 3.4 – Реалізація зміни статусу завдання та автоматичного формування сповіщення

```
const changeStatus = async (id, status) =&gt; {
  const task = await Task.findByPk(id);
  if (!task) throw new Error('Завдання не знайдено');
  await task.update({ status });
  if (task.executor_id) {
    await Notification.create({
      title: 'Зміна статусу завдання',
      message: `Статус завдання «${task.title}» змінено на «${status}»`,
      user_id: task.executor_id });
  }
  return task;
};
```

Окремий аналітичний сервіс обчислює завантаженість команди за проєктом параметризованим SQL-запитом, що підсумовує витрачені години за кожним користувачем, об’єднуючи таблиці журналу часу, користувачів, завдань і спринтів за умовою належності до проєкту. Приклад SQL-запиту, що використовується для обчислення завантаженості команди, наведено нижче.

```
SELECT u.name, SUM(tl.hours) AS total_hours
FROM time_logs tl
JOIN users u  ON u.id = tl.user_id
JOIN tasks t  ON t.id = tl.task_id
JOIN sprints s ON s.id = t.sprint_id
WHERE s.project_id = ?
GROUP BY u.id, u.name;
```

Сервіс формування звітів збирає дані про проєкт, його спринти, завдання та логи часу, обчислює статистику (кількість завдань за статусами, відсоток виконання, сумарні години, завантаженість команди) і генерує PDF-документ

засобами PDFKit, після чого зберігає файл у каталозі reports і фіксує запис про звіт у базі даних. Повні лістинги моделей, сервісів і контролерів наведено в додатку В.

3.3 Реалізація REST-API

Взаємодію клієнта із сервером реалізовано за архітектурним стилем REST: ресурси адресуються маршрутами під префіксом /api, а дані передаються у форматі JSON. Усі маршрути, окрім автентифікації, захищено посередником перевірки токена. Операції створення, редагування та видалення проєктів, спринтів і завдань додатково обмежено ролями менеджера та адміністратора, тоді як зміна статусу завдання (перетягування на Kanban-дошці), облік часу, перегляд і створення сповіщень, а також робота з коментарями доступні будь-якому автентифікованому користувачеві; видалення коментаря дозволено лише його автору або користувачеві з роллю менеджера чи адміністратора. Повний перелік точок доступу REST-API наведено в таблиці 3.1.

Таблиця 3.1 — Точки доступу REST-API системи «ProSync IT»

Метод Маршрут Доступ Призначення
POST /api/auth/login публічний Автентифікація, видача
токена JWT
GET /api/projects авторизований Перелік проєктів POST /api/projects manager, admin Створення проєкту PUT /api/projects/:id manager, admin Редагування проєкту DELETE /api/projects/:id manager, admin Видалення проєкту GET /api/ sprints авторизований Перелік спринтів
POST /api/sprints manager, admin Створення спринту
PUT /api/sprints/:id manager, admin Редагування спринту
DELETE /api/sprints/:id manager, admin Видалення спринту
GET /api/tasks авторизований Перелік завдань із
виконавцями
POST /api/tasks manager, admin Створення завдання (з
валідацією)

Продовження таблиці 3.1
Метод Маршрут Доступ Призначення
PATCH /api/tasks/:id/status авторизований Зміна статусу завдання +
сповіщення
GET /api/time-logs авторизований Перелік записів обліку
часу
POST /api/time-logs авторизований Створення запису обліку
часу
GET /api/comments/task
/:taskId
авторизований Коментарі конкретного
завдання
POST /api/comments авторизований Додавання коментаря
DELETE /api/comments/:id автор / manager
/ admin
Видалення коментаря
GET /api/notifications авторизований Перелік сповіщень
POST /api/notifications авторизований Створення сповіщення
GET /api/reports авторизований Перелік згенерованих
звітів
POST /api/reports/generate
/:projectId
авторизований Генерація PDF-звіту за
проєктом
GET /api/analytics
/workload
/:projectId
авторизований Завантаженість команди
проєкту

3.4 Реалізація клієнтської частини

Клієнтську частину реалізовано як односторінковий застосунок. Кореневий компонент огорнуто провайдером контексту автентифікації, а маршрутизацію між сторінками реалізовано бібліотекою react-router-dom. Передбачено сторінки входу, головної панелі (Dashboard), Kanban-дошки, аналітики, звітів і сповіщень. Доступ до сторінок обмежено компонентом захищеного маршруту, який перенаправляє неавтентифікованого користувача на сторінку входу, а сторінку аналітики додатково відкриває лише для ролей менеджера та адміністратора. Контекст автентифікації зберігає токен і дані користувача в локальному сховищі браузера та надає функції входу й виходу. HTTP-клієнт axios налаштовано перехоплювачем запитів, який автоматично

додає токен доступу до заголовка кожного звернення до сервера. Навігацію між модулями забезпечує бічна панель (Sidebar) з посиланнями на основні

розділи та кнопкою виходу.

3.5 Реалізація ключових функціональних модулів

3.5.1 Автентифікація та рольовий доступ

Сторінка входу надсилає введені облікові дані на сервер; у разі успіху отримані токен і профіль користувача зберігаються в контексті та локальному сховищі, після чого здійснюється перехід на головну панель. Подальший доступ до даних регулюється на двох рівнях: на клієнті — захищеним маршрутом, на сервері — посередниками автентифікації та авторизації. Екран входу наведено на рисунку 3.2.

Рисунок 3.2 — Екран автентифікації користувача

3.5.2 Kanban-дошка та механізм Drag-and-Drop

Центральним елементом інтерфейсу є Kanban-дошка, реалізована на бібліотеці @dnd-kit. Дошку обгорнуто контекстом перетягування DndContext; кожен стовпець-статус є зоною скидання (useDroppable) з ідентифікатором, що відповідає статусу завдання, а картка завдання — елементом перетягування (useDraggable) з рукояткою. Дошка відображає чотири стовпці статусів — «To

Do», «In Progress», «Testing» і «Done»; завдання у статусі беклогу зберігаються в базі, але на дошку не виводяться. Кожна картка показує назву, оцінку у Story Points, термін, призначеного виконавця та позначку пріоритету; подвійне клацання відкриває модуль обліку часу, а кнопка редагування — вікно редагування завдання. Після завершення перетягування обробник надсилає запит на зміну статусу завдання (метод PATCH), де нове значення статусу береться з ідентифікатора цільового стовпця, і оптимістично оновлює стан інтерфейсу.

Лістинг 3.5 – Реалізація зміни статусу завдання методом Drag-and-Drop

```
const handleDragEnd = async ({ active, over }) =&gt; {
  if (!over) return;
  const taskId = Number(active.id);
  const newStatus = over.id;
  await api.patch(`/tasks/${taskId}/status`, { status: newStatus });
  setTasks(prev =&gt; prev.map(t =&gt;
    t.id === taskId ? { ...t, status: newStatus } : t));
};
```

Загальний вигляд Kanban-дошки наведено на рисунку 3.3.

Рисунок 3.3 — Kanban-дошка завдань зі стовпцями статусів

3.5.3 Управління завданнями та спринтами

Створення завдання реалізовано модальним вікном із полями назви, опису, пріоритету, виконавця, оцінки складності, терміну та спринту (перелік

спринтів завантажується із сервера); новоствореному завданню присвоюється статус «todo». Вікно редагування дозволяє змінити атрибути завдання або видалити його з підтвердженням. Створення та редагування проєктів і спринтів виконується через відповідні захищені точки доступу, доступні менеджеру та адміністратору. Модальне вікно створення завдання наведено на рисунку 3.4.

Рисунок 3.4 — Модальне вікно створення завдання

3.5.4 Облік робочого часу

Модуль обліку часу реалізовано модальним вікном, що відкривається з картки завдання. Користувач вводить кількість витрачених годин (із кроком 0,5) та короткий коментар, після чого запис із поточною датою, ідентифікаторами завдання й користувача надсилається на сервер і зберігається в журналі обліку часу. Накопичені дані використовуються для обчислення завантаженості команди та формування звітності. Вікно обліку

часу наведено на рисунку 3.5.

Рисунок 3.5 — Модуль обліку робочого часу

3.5.5 Система сповіщень

Сповіщення формуються на стороні сервера автоматично — зокрема під час зміни статусу завдання його призначеному виконавцеві створюється відповідний запис. Сторінка сповіщень завантажує перелік повідомлень користувача й відображає їх списком, а за відсутності нових повідомлень показує відповідний порожній стан. Сторінку сповіщень наведено на рисунку 3.6.

Рисунок 3.6 — Сторінка сповіщень користувача

3.5.6 Аналітика та візуалізація даних

Аналітичні можливості реалізовано на двох сторінках. Головна панель відображає чотири ключові показники — кількість активних проєктів, завдань у роботі, прострочених завдань (термін яких минув, а статус не «done») і сповіщень, а також стовпчикову діаграму завантаженості команди, побудовану засобами Recharts на основі даних аналітичного сервісу. Сторінка аналітики, доступна менеджеру та адміністратору, містить показники загальної кількості завдань, виконаних і тих, що тестуються, сумарних витрачених годин, індикатор відсотка виконання та кругову діаграму розподілу завдань за статусами. Головну панель і сторінку аналітики наведено на рисунках 3.7 та 3.8.

Рисунок 3.7 — Головна панель із показниками та діаграмою завантаженості

Рисунок 3.8 — Сторінка аналітики розподілу завдань

3.5.7 Генерація PDF-звітів

Модуль звітності дозволяє сформувати підсумковий PDF-звіт за проєктом. На сторінці звітів користувач ініціює генерацію, після чого сервер створює документ і повертає шлях до файлу, який відкривається в новій вкладці; раніше сформовані звіти відображаються списком із можливістю повторного відкриття. Звіт формується засобами PDFKit із підключенням шрифту DejaVuSans для коректного відображення кирилиці й містить інформацію про проєкт, статистику завдань за статусами та відсоток виконання, сумарні витрачені години, завантаженість команди й відомості про спринти.

Лістинг 3.6 – Фрагмент реалізації генерації PDF-звіту

```
const doc = new PDFDocument();

doc.font(path.join(__dirname, '../../fonts/DejaVuSans.ttf'));
doc.pipe(fs.createWriteStream(filePath));

doc.fontSize(26)
   .fillColor('#2563eb')
   .text('ProSync IT', { align: 'center' });

doc.moveDown();
doc.fontSize(14)
   .text(`Проєкт: ${project.title}`);
```

doc.end();
Приклад згенерованого PDF-звіту наведено на рисунку 3.9.

Рисунок 3.9 — Приклад згенерованого PDF-звіту за проєктом

3.5.8 Коментування завдань

На рівні серверної частини реалізовано модуль коментарів: отримання коментарів завдання разом із даними автора, додавання коментаря (з автоматичним поверненням імені автора для відображення) та видалення з перевіркою прав — операція дозволена авторові коментаря або користувачеві з роллю менеджера чи адміністратора, інакше повертається відмова зі статусом 403. Відповідні точки доступу REST-API є повністю функціональними; інтеграція коментарів у графічний інтерфейс становить напрям подальшого розвитку системи.

3.6 Тестування системи

Перевірку працездатності системи виконано у два етапи — автоматизованим та ручним тестуванням. Автоматизоване тестування серверної частини реалізовано засобами Jest і Supertest, що дозволяє надсилати HTTP-запити до застосунку без запуску реального мережевого сервера. Розроблені тести перевіряють коректність роботи підсистеми безпеки: відмову

в автентифікації за невірних облікових даних (очікуваний статус 401) та блокування доступу до захищеного ресурсу без токена (очікуваний статус 401). Запуск тестів здійснюється командою npm test.

Для перевірки функціональних сценаріїв застосовано ручне тестування за методом «чорної скриньки». Сформовано набір тест-кейсів, що охоплюють ключові сценарії використання системи, з фіксацією очікуваного результату (таблиця 3.2).

Таблиця 3.2 — Функціональні тест-кейси системи

No	Сценарій	Вхідні дані	Очікуваний результат	Статус
1	Вхід із коректними даними	email, пароль	Статус 200, видано токен, перехід на Dashboard	✓
2	Вхід із невірним паролем	email, хибний пароль	Статус 401, повідомлення про помилку	✓
3	Запит даних без токена	GET /api/projects	Статус 401	✓
4	Створення завдання менеджером	дані завдання	Статус 201, завдання створено	✓
5	Створення			

	завдання розробником дані завдання		Статус 403, доступ заборонено
✓			
6	Створення завдання без назви	порожня назва	Статус 400, помилка валідації
✓			
7	Перетягування картки на дошці	нова колонка-статус	Статус оновлено, створено сповіщення
✓			
8	Внесення обліку часу	години, коментар	Статус 201, запис збережено
✓			
9	Генерація PDF-звіту	ідентифікатор проєкту	Статус 201, файл сформовано й відкрито
✓			
10	Відкриття аналітики роллю developer	перехід на /analytics	Перенаправлення на /dashboard
✓			

Для розгортання тестового середовища з демонстраційними даними використано засоби заповнення бази (seeders), які створюють п’ять ролей (admin, manager, developer, qa, client) та відповідні облікові записи користувачів. На рівні безпеки забезпечено зберігання паролів у вигляді криптографічного хешу, обмежений термін дії токена доступу, винесення секретів у середовище та встановлення захисних HTTP-заголовків.

3.7 Висновки до розділу 3

У розділі описано програмну реалізацію системи «ProSync IT» відповідно до спроєктованої архітектури. 23 Серверну частину реалізовано на Node.js та Express із рівнями моделей, сервісів і контролерів, доступом до даних через Sequelize, рольовим розмежуванням на основі токенів JWT і хешуванням паролів; побудовано REST-API з двадцяти чотирьох точок доступу. Клієнтську частину реалізовано як односторінковий застосунок на React зі складанням засобом Vite. Реалізовано ключові модулі: автентифікацію та рольовий доступ, Kanban-дошку з механізмом Drag-and-Drop, облік робочого часу, систему сповіщень, аналітику з діаграмами та генерацію PDF-звітів. Працездатність системи підтверджено автоматизованими тестами (Jest, Supertest) та набором ручних функціональних тест-кейсів, що засвідчили відповідність реалізації поставленим вимогам.

ВИСНОВКИ

У бакалаврській роботі розв'язано актуальну практичну задачу — спроєктовано та програмно реалізовано вебзорієнтовану інформаційну систему управління проєктами «ProSync IT», призначену для планування робіт, ведення завдань, обліку робочого часу та аналізу діяльності команди розробки.

На етапі аналізу предметної області досліджено процеси управління IT-проєктами за гнучкими методологіями та виявлено характерні проблеми: низьку прозорість розподілу навантаження, відсутність достовірного обліку часу, високу трудомісткість звітності й децентралізацію даних. Порівняльний аналіз програмних аналогів (Jira, Trello, Asana) показав, що наявні рішення є або надмірно складними й дорогими, або функціонально обмеженими, що обґрунтувало доцільність розроблення власної системи. На цій основі сформовано вимоги, визначено ролі користувачів і сценарії використання.

На етапі проєктування обґрунтовано технологічний стек, побудовано функціональні та процесні моделі (IDEF0, DFD, IDEF3, BPMN), спроєктовано багаторівневу клієнт-серверну архітектуру з патерном MVC і сервісним шаром та REST-взаємодією, розроблено об'єктно-орієнтовані моделі (діаграми класів, послідовності й діяльності) і спроєктовано нормалізовану реляційну базу даних із дев'яти таблиць із налаштованими правилами посилальної цілісності.

На етапі реалізації створено серверну частину на Node.js та Express із доступом до даних через Sequelize, рольовим розмежуванням на основі JWT і хешуванням паролів, а також REST-API з двадцяти чотирьох точок доступу. Клієнтську частину реалізовано як односторінковий застосунок на React зі складанням засобом Vite. Реалізовано ключові модулі системи: автентифікацію, Kanban-дошку з механізмом Drag-and-Drop, облік робочого часу, систему сповіщень, аналітику з візуалізацією та генерацію PDF-звітів. Працездатність підтверджено автоматизованим і ручним тестуванням.

Практична цінність роботи полягає у створенні працездатної інформаційної системи, готової до використання в реальному IT-середовищі. Спроєктована платформа формує єдиний цифровий простір управління проєктами, що сприяє досягненню закладених бізнес-метрик — скороченню витрат часу на звітність і зменшенню частки прострочених завдань. Подальший розвиток системи можливий у напрямах інтеграції коментарів у графічний інтерфейс, розширення аналітики (зокрема діаграми згоряння), додавання гнучкого вибору проєкту в модулях аналітики та звітності й під'єднання мобільного клієнта до наявного серверного ядра.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Швабер К., Сазерленд Дж. Керівництво зі Scrum: Вичерпний посібник зі Scrum: Правила гри / офіц. укр. переклад. 2020. URL: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Ukrainian.pdf (дата звернення: 09.06.2026).
2. Jira | Project management software : офіц. сайт. Atlassian. URL: https://www.atlassian.com/software/jira (дата звернення: 09.06.2026).
3. Trello : офіц. сайт. Atlassian. URL: https://trello.com (дата звернення: 09.06.2026).
4. Asana : офіц. сайт. Asana, Inc. URL: https://asana.com (дата звернення: 09.06.2026).

5. Сазерленд Дж. Scrum. Навчись робити вдвічі більше за менший час / пер. з англ. Харків : Клуб Сімейного Дозвілля, 2021. 288 с.
6. Manifesto for Agile Software Development / K. Beck et al. 2001. URL: https://agilemanifesto.org (дата звернення: 09.06.2026).
7. React : офіц. документація. Meta Open Source. URL: https://react.dev (дата звернення: 09.06.2026).
8. Vite : Next Generation Frontend Tooling : офіц. сайт. URL: https://vite.dev (дата звернення: 09.06.2026).
9. dnd kit — drag and drop toolkit for React : офіц. документація. URL: https://dndkit.com (дата звернення: 09.06.2026).
10. Recharts : Redefined chart library built with React and D3 : офіц. сайт. URL: https://recharts.org (дата звернення: 09.06.2026).
11. Node.js : офіц. документація. OpenJS Foundation. URL: https://nodejs.org/en/docs (дата звернення: 09.06.2026).

12. Express — Node.js web application framework : офіц. сайт. OpenJS Foundation. URL: https://expressjs.com (дата звернення: 09.06.2026).
13. Sequelize ORM : офіц. документація. URL: https://sequelize.org (дата звернення: 09.06.2026).
14. JSON Web Tokens — jwt.io : офіц. ресурс. URL: https://jwt.io (дата звернення: 09.06.2026).
15. bcrypt : A library to help you hash passwords : пакет npm. URL: https://www.npmjs.com/package/bcrypt (дата звернення: 09.06.2026).
16. PDFKit : A JavaScript PDF generation library for Node : офіц. сайт. URL: https://pdfkit.org (дата звернення: 09.06.2026).
17. MySQL 8.0 Reference Manual : офіц. документація. Oracle Corporation. URL: https://dev.mysql.com/doc (дата звернення: 09.06.2026).
18. Axios : Promise based HTTP client : офіц. документація. URL: https://axios-http.com (дата звернення: 09.06.2026).
19. Helmet.js : Secure Express apps with HTTP headers : офіц. сайт. URL: https://helmetjs.github.io (дата звернення: 09.06.2026).
20. React Router : офіц. документація. URL: https://reactrouter.com (дата звернення: 09.06.2026).
21. Трофименко О. Г., Прокоп Ю. В. Організація та проєктування баз даних : навч. посібник. 2-ге вид. Одеса : Фенікс, 2019. 266 с.
22. Грицюк Ю. І., Рак Т. Є. Архітектура та проєктування програмного забезпечення : навч. посібник. Львів : Вид-во ЛДУ БЖД, 2018. 320 с.

ДОДАТКИ

ДОДАТОК А
Лістинг SQL-скрипту створення бази даних
Таблиця «Довідник ролей»

```
CREATE TABLE `roles` (
 `id` int NOT NULL AUTO_INCREMENT,
 `role_name` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
 `created_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
 `updated_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (`id`),
 UNIQUE KEY `role_name` (`role_name`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Таблиця «Користувачі»

```
CREATE TABLE `users` (
 `id` int NOT NULL AUTO_INCREMENT,
 `name` varchar(100) COLLATE utf8mb4_unicode_ci NOT NULL,
 `email` varchar(100) COLLATE utf8mb4_unicode_ci NOT NULL, `password_hash` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
 `is_active` tinyint(1) NOT NULL DEFAULT '1',
 `role_id` int NOT NULL,
 `created_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
```

```
  `updated_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`),
  KEY `role_id` (`role_id`),
  CONSTRAINT `users_ibfk_1` FOREIGN KEY (`role_id`) REFERENCES `roles` (`id`) ON DELETE RESTRICT ON UPDATE CASCADE )
ENGINE=InnoDB AUTO_INCREMENT= 6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Таблиця «Проєкти»

```
CREATE TABLE `projects` (
  `id` int NOT NULL AUTO_INCREMENT,
  `title` varchar( 150) COLLATE utf8mb4_unicode_ci NOT NULL,
  `description` text COLLATE utf8mb4_unicode_ci,
  `time_budget` int DEFAULT NULL,
  `start_date` date NOT NULL,
  `end_date` date DEFAULT NULL,
  `status` enum('planning','active','completed','archived') COLLATE utf8mb4_unicode_ci
NOT NULL DEFAULT 'planning',
  `manager_id` int NOT NULL,
  `created_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (`id`),
  KEY `manager_id` (`manager_id`),
  CONSTRAINT `projects_ibfk_1` FOREIGN KEY (`manager_id`) REFERENCES `users` (`id`) ON
DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```


Таблиця «Спринти»

```
CREATE TABLE `sprints` (
  `id` int NOT NULL AUTO_INCREMENT,
  `sprint_name` varchar(100) COLLATE utf8mb4_unicode_ci NOT NULL,
  `start_date` date NOT NULL,
  `end_date` date NOT NULL,
  `status` enum('planned','active','completed') COLLATE utf8mb4_unicode_ci NOT NULL
DEFAULT 'planned',
  `project_id` int NOT NULL,
  `created_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (`id`),
  KEY `project_id` (`project_id`),
  CONSTRAINT `sprints_ibfk_1` FOREIGN KEY (`project_id`) REFERENCES `projects` (`id`) ON DELETE CASCADE ON UPDATE CASCADE )
ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Таблиця «Завдання»

```
CREATE TABLE `tasks` (
  `id` int NOT NULL AUTO_INCREMENT,
  `title` varchar( 150) COLLATE utf8mb4_unicode_ci NOT NULL,
  `description` text COLLATE utf8mb4_unicode_ci,
  `priority` enum('low','medium','high','critical') COLLATE utf8mb4_unicode_ci NOT NULL
DEFAULT 'medium',
  `story_points` int DEFAULT NULL,
  `status` enum('backlog','todo','in_progress','testing','done') COLLATE
utf8mb4_unicode_ci NOT NULL DEFAULT 'backlog',
  `deadline` date DEFAULT NULL,
  `sprint_id` int NOT NULL,
  `executor_id` int DEFAULT NULL,
  `created_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (`id`),
  KEY `sprint_id` (`sprint_id`),
  KEY `executor_id` (`executor_id`),
```



```
  CONSTRAINT `tasks_ibfk_1` FOREIGN KEY (`sprint_id`) REFERENCES `sprints` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `tasks_ibfk_2` FOREIGN KEY (`executor_id`) REFERENCES `users` (`id`) ON DELETE SET NULL ON UPDATE CASCADE )
ENGINE=InnoDB AUTO_INCREMENT= 11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Таблиця «Журнал обліку часу»

```
CREATE TABLE `time_logs` (
  `id` int NOT NULL AUTO_INCREMENT,
```

```
  `hours` decimal(5,2) NOT NULL,
  `log_date` date NOT NULL,
  `comment` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `task_id` int NOT NULL,
  `user_id` int NOT NULL,
  `created_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (`id`),
  KEY `task_id` (`task_id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `time_logs_ibfk_1` FOREIGN KEY (`task_id`) REFERENCES `tasks` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `time_logs_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE RESTRICT ON UPDATE CASCADE )
ENGINE=InnoDB AUTO_INCREMENT= 10 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
Таблиця «Коментарі»
CREATE TABLE `comments` (
  `id` int NOT NULL AUTO_INCREMENT,
  `content` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `task_id` int NOT NULL,
  `user_id` int NOT NULL,
  `created_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (`id`),
  KEY `task_id` (`task_id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `comments_ibfk_1` FOREIGN KEY (`task_id`) REFERENCES `tasks` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `comments_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE RESTRICT ON UPDATE CASCADE )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
Таблиця «Сповіщення»
CREATE TABLE `notifications` (
  `id` int NOT NULL AUTO_INCREMENT,
  `title` varchar(150) COLLATE utf8mb4_unicode_ci NOT NULL,
  `message` text COLLATE utf8mb4_unicode_ci NOT NULL,
```



```
  `is_read` tinyint(1) NOT NULL DEFAULT '0',
  `user_id` int NOT NULL,
  `created_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (`id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `notifications_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
ON DELETE CASCADE ON UPDATE CASCADE ) ENGINE=InnoDB AUTO_INCREMENT= 29 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
Таблиця «Звіти»
CREATE TABLE `reports` (
  `id` int NOT NULL AUTO_INCREMENT,
  `report_type` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `file_path` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `project_id` int NOT NULL,
  `generated_by` int NOT NULL,
  `created_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (`id`),
  KEY `project_id` (`project_id`),
  KEY `generated_by` (`generated_by`),
  CONSTRAINT `reports_ibfk_1` FOREIGN KEY (`project_id`) REFERENCES `projects` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `reports_ibfk_2` FOREIGN KEY (`generated_by`) REFERENCES `users` (`id`) ON DELETE RESTRICT ON UPDATE CASCADE )
ENGINE=InnoDB AUTO_INCREMENT= 2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
Таблиця «Службова таблиця міграцій Sequelize»
CREATE TABLE `sequelizemeta` (
  `name` varchar(255) COLLATE utf8mb3_unicode_ci NOT NULL,
  PRIMARY KEY (`name`),
  UNIQUE KEY `name` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_unicode_ci;
```

ДОДАТОК Б
Лістинги серверної частини
Лістинг Б.1 — Визначення моделей і зв'язків (Sequelize)

```
const Role = require('./Role');
const User = require('./User');
const Project = require('./Project');
const Sprint = require('./Sprint');
const Task = require('./Task');
const TimeLog = require('./TimeLog');
const Notification = require('./Notification');
const Report = require('./Report');
const Comment = require('./Comment');

Role.hasMany(User, {
  foreignKey: 'role_id'
});

User.belongsTo(Role, {
  foreignKey: 'role_id'
});

User.hasMany(Project, {
  foreignKey: 'manager_id'
});

Project.belongsTo(User, {
  as: 'manager',
  foreignKey: 'manager_id'
});

Project.hasMany(Sprint, {
  foreignKey: 'project_id'
});

Sprint.belongsTo(Project, {
  foreignKey: 'project_id'
});

Sprint.hasMany(Task, {
  foreignKey: 'sprint_id'
});

Task.belongsTo(Sprint, {
  foreignKey: 'sprint_id'
});

Task.hasMany(TimeLog, {
  foreignKey: 'task_id'
});
```


Продовження додатку Б

```
TimeLog.belongsTo(Task, {
  foreignKey: 'task_id'
});
User.hasMany(TimeLog, {
  foreignKey: 'user_id'
});

User.hasMany(Notification, {
  foreignKey: 'user_id'
});

Notification.belongsTo(User, {
  foreignKey: 'user_id'
});
```

```
TimeLog.belongsTo(User, {
  foreignKey: 'user_id'
});

User.hasMany(Task, {
  foreignKey: 'executor_id'
});

Task.belongsTo(User, {
  as: 'executor',
  foreignKey: 'executor_id'
});

User.hasMany(Report, {
  foreignKey: 'generated_by'
});

Report.belongsTo(User, {
  foreignKey: 'generated_by'
});

Project.hasMany(Report, {
  foreignKey: 'project_id'
});

Report.belongsTo(Project, {
  foreignKey: 'project_id'
});

Task.hasMany(Comment, {
  foreignKey: 'task_id'
});

Comment.belongsTo(Task, {
  foreignKey: 'task_id'
});
```


Продовження додатку Б

```
User.hasMany(Comment, {
  foreignKey: 'user_id'
});
Comment.belongsTo(User, {
  foreignKey: 'user_id'
});

module.exports = {
  Role,
  User,
  Project,
  Sprint,
  Task,
  TimeLog,
  Notification,
  Report,
  Comment
};
```

Лістинг Б.2 — Проміжне ПЗ автентифікації та авторизації

```
(auth. middleware.js)
 const jwt = require('jsonwebtoken'); const   authenticate = (req, res, next) =&gt; { const authHeader =  req.headers.authorization;

  if (
    ! authHeader ||
    !authHeader. startsWith(' Bearer  ')
  ) {
```

```
    return res.status( 401). json({
      message: 'Потрібен токен'
    });
  }

  try { const  token = authHeader.split(' ')[1];

    req.user  =  jwt.verify( token, process.env.JWT_SECRET
  );

    next(); } catch (error)  {
    return res.status(401).json({
      message:  'Недійсний токен'
    });
  }
};

const authorize = (...roles) =&gt; {
```


Продовження додатку Б

```
  return (req, res, next) =&gt; { if (!roles.includes(req.user. role)) {
    return res.status(403).json({ message:  'Доступ заборонено'
  });
  }

    next(); }; }; module.exports = { authenticate, authorize
};
```

Лістинг Б.3 — Сервіс автентифікації (auth.service.js)

```
const bcrypt = require( 'bcrypt');
const jwt = require('jsonwebtoken'); const  { User,  Role } =  require('../models');

const login = async (email, password) =&gt; {
  const user = await User.findOne({
    where: { email },
    include: Role
  });

  if (!user) {
    throw new Error(' Невірний email або  пароль');
  }

  const  isValidPassword =  await bcrypt.compare( password, user.password_  hash
  );

  if (!isValidPassword) {
    throw new Error('Невірний email або пароль');
  }

  const token = jwt.sign(
    {
      id: user.id,
      role: user.Role. role_name
    },
    process.env.JWT_SECRET,
    {
      expiresIn:  '8h'
    }
  );
```


Продовження додатку Б

```
  return {
    token,
    user: {
```

```
    id: user.id,
    name: user.name,
    email: user.email,
    role: user.Role.role_name
   }
  };
};

module.exports = {
  login
};
```

Лістинг Б.4 — Сервіс управління завданнями (task.service.js)

```
const { Task, User, Notification } = require('../models');

// Список задач разом із виконавцем (щоб дошка могла показувати, на кого призначено)
const getAllTasks = async () =&gt; {
  return await Task.findAll({
    include: [
     {
       model: User,
       as: 'executor',
       attributes: ['id', 'name']
     }
    ],
    order: [['created_at', 'DESC']]
  });
};

const createTask = async (taskData) =&gt; {
  return await Task.create(taskData);
};

const updateTask = async (id, taskData) =&gt; {
  const task = await Task.findByPk(id);

  if (!task) {
    throw new Error('Завдання не знайдено');
  }

  await task.update(taskData);

  return task;
};

const deleteTask = async (id) =&gt; {
  const task = await Task.findByPk(id);

  if (!task) {
```



```
    throw new Error('Завдання не знайдено');
  }

  await task.destroy();

  return {
    message: 'Завдання успішно видалено'
  };
};

const changeStatus = async (id, status) =&gt; {
  const task = await Task.findByPk(id);

  if (!task) {
    throw new Error('Завдання не знайдено');
```

```
  }

  await task.update({ status });

  // Бізнес-логіка тепер на сервері: сповіщення створюється надійно при кожній
  // зміні статусу, незалежно від клієнта. Сповіщаємо призначеного виконавця.
  if (task.executor_id) {
    await Notification.create({
      title: 'Зміна статусу завдання',
      message: `Статус завдання «${task.title}» змінено на «${status}»`,
      user_id: task.executor_id
    });
  }

  return task;
};

module.exports = {
  getAllTasks,
  createTask,
  updateTask,
  deleteTask,
  changeStatus
};
```

Лістинг Б.5 — Сервіс генерації PDF-звітів (report.service.js)

```
const PDFDocument = require('pdfkit');
32 const fs = require('fs'); const path = require('path'); const analyticsService = require('./ analytics.service');

 const {
  Report,
  Project,
  Sprint,
  Task,
  TimeLog
```



```
} = require('../models');

const generateReport = async (
  projectId,
  generatedBy
) =&gt; {
  const reportsDir = path.join(
    __dirname,
    '../../reports'
  );

  if (!fs.existsSync(reportsDir)) {
    fs.mkdirSync(reportsDir);
  }

  const now = new Date();

  const fileName =
    `Project_Report_${now.getFullYear()
    }-${String(
      now.getMonth() + 1
    ).padStart(2, '0')
    }-${String(
      now.getDate()
    ).padStart(2, '0')
    }_${String(
      now.getHours()
    ).padStart(2, '0')
    }-${String(
```

```
    now.getMinutes()
  ).padStart(2, '0')
  }-${String(
    now.getSeconds()
  ).padStart(2, '0')
  }.pdf`;

 const filePath =
  path.join(reportsDir, fileName);

 // Отримуємо проєкт
 const project =
  await Project.findByPk(projectId);

 if (!project) {
  throw new Error(
   'Проєкт не знайдено'
  );
 }

 // Отримуємо спринти проєкту
 const sprints =
  await Sprint.findAll({
```


Продовження додатку Б

```
    where: {
     project_id: projectId
    }
  });
 const sprintIds =
 sprints.map(
    sprint =&gt; sprint.id
  );

 // Отримуємо задачі проєкту
 const tasks =
  await Task.findAll({
    where: {
     sprint_id: sprintIds
    }
  });

 const taskIds =
  tasks.map(
    task =&gt; task.id
  );

 // Статистика задач
 const totalTasks =
  tasks.length;

 const todoTasks =
  tasks.filter(
    task =&gt; task.status === 'todo'
  ).length;

 const inProgressTasks =
  tasks.filter(
    task =&gt;
     task.status === 'in_progress'
  ).length;

 const testingTasks =
  tasks.filter(
    task =&gt; task.status === 'testing'
```

```
    ).length;

  const doneTasks =
    tasks.filter(
      task =&gt; task.status === 'done'
    ).length;

  const completionRate =
    totalTasks &gt; 0
      ? Math.round(
        (doneTasks / totalTasks) * 100
      )
```


Продовження додатку Б

```
      : 0;

  // Логи часу
  const timeLogs =
    await TimeLog.findAll({
      where: {
        task_id: taskIds
      }
    });

  const totalHours =
    timeLogs.reduce(
      (sum, log) =&gt;
        sum + Number(log.hours),
      0
    );

  const workload =
    await analyticsService
      .getWorkloadByProject(
        projectId
      );

  // PDF
  const doc = new PDFDocument();

  doc.font(
  path.join(
    __dirname,
    '../../fonts/DejaVuSans.ttf'
  )
);

  doc.pipe(
    fs.createWriteStream(filePath)
  );

  doc
  .fontSize(26)
  .fillColor('#2563eb')
  .text(
    'ProSync IT',
    {
      align: 'center'
    }
  );

doc
  .fontSize(18)
  .fillColor('black')
  .text(
```

```
    'Project Analytics Report',
```

74

Продовження додатку Б

```
    {
      align: 'center'
    }
  );
  doc.moveDown();

  doc.text(
    '================================',
    {
      align: 'center'
    }
  );

  doc.moveDown();

  doc
    .fontSize(18)
    .fillColor('#2563eb')
    .text('Project Information');

  doc.fillColor('black');

  doc.moveDown(0.5);

  doc.text(
  `Project: ${project.title}`
);

  doc.text(
    `Description: ${project.description}`
  );

  doc.text(
    `Status: ${project.status}`
  );

  doc.text(
    `Start Date: ${project.start_date}`
  );

  doc.text(
    `End Date: ${project.end_date}`
  );

  doc.moveDown();

  doc
    .fontSize(16)
    .fillColor('#2563eb')
    .text('Task Statistics');
```

75

Продовження додатку Б

```
  doc.fillColor('black');
    doc.moveDown(0.5);

  doc.text(
  `Total Tasks: ${totalTasks}`
);

doc.moveDown(0.3);
```

```
  doc.text(
    `Done ..... ${doneTasks}`
  );

  doc.text(
    `Testing ..... ${testingTasks}`
  );

  doc.text(
    `In Progress ..... ${inProgressTasks}`
  );

  doc.text(
    `To Do ..... ${todoTasks}`
  );

  doc.moveDown(0.3);

  doc.text(
    `Completion Rate .... ${completionRate}%`
  );

  doc.moveDown();

doc
  .fontSize(16)
  .fillColor('#2563eb')
  .text('Time Tracking');

doc.fillColor('black');

doc.moveDown(0.5);

doc.text(
  `Total Hours Logged: ${totalHours}`
);

doc.moveDown();

doc
  .fontSize(16)
  .fillColor('#2563eb')
  .text('Team Workload');
```


Продовження додатку Б

```
doc.fillColor('black');

doc.moveDown(0.5);
workload.forEach(item =&gt; {

  doc.text(
    `${item.name} ..... ${item.total_hours} h`
  );

});

doc.moveDown();
doc.moveDown();
doc.moveDown();

doc
  .fontSize(16)
  .fillColor('#2563eb')
  .text('Sprint Information');
```

```
doc.fillColor('black');

doc.moveDown(0.5);

sprints.forEach(
  sprint =&gt; {

    doc.text(
      `${sprint.sprint_name}`
    );

    doc.text(
      `Status: ${sprint.status}`
    );

    doc.text(
      `${sprint.start_date} - ${sprint.end_date}`
    );

    doc.moveDown(1);

  }
);

doc.moveDown();

  doc.moveDown();

  doc.text(
    '----------------------------------------'
  );
```


Продовження додатку Б

```
  doc.moveDown(0.5);

  doc.fontSize(10);
  doc.text(
    `Generated by ProSync IT`
  );

  doc.text(
    new Date().toLocaleString()
  );

  doc.end();

  const report =
    await Report.create({
      report_type: 'project',
      file_path: `/reports/${fileName}`,
      project_id: projectId,
      generated_by: generatedBy
    });

  return report;
};

const getAllReports = async () =&gt; {

  return await Report.findAll({
    order: [
      ['created_at', 'DESC']
    ]
  });
```

```
};

module.exports = {
  generateReport,
  getAllReports
};
```

Лістинг Б.6 — Сервіс аналітики (analytics.service.js)

```
const sequelize = require('../config/database');

const getWorkloadByProject = async (projectId) =&gt; {
  const [results] = await sequelize.query(`
    SELECT
      u.name,
      SUM(tl.hours) AS total_hours
    FROM time_logs tl
    JOIN users u ON u.id = tl.user_id
    JOIN tasks t ON t.id = tl.task_id
    JOIN sprints s ON s.id = t.sprint_id
    WHERE s.project_id = ?
```



```
    GROUP BY u.id, u.name
  `, {
    replacements: [projectId]
  });
  return results;
};

module.exports = {
  getWorkloadByProject
};
```

Лістинг Б.7 — Маршрути REST-API (routes/*.js)

```
// ----- analytics.routes.js ----- const express = require('express'); const router = express.Router(); const analyticsController = require(
  '../controllers/analytics.controller'
);

const {
  authenticate
} = require('../middleware/auth.middleware');

router.get(
  '/workload/:projectId',
  authenticate,
  analyticsController.workload
);

module.exports = router;

// ----- auth.routes.js ----- const express = require('express'); const router = express.Router(); const authController = require(
  '../controllers/auth.controller'
);

router.post(
  '/login',
  authController.login
);

module.exports = router;

// ----- comment.routes.js -----
```

```
const express = require('express'); const router = express.Router(); const commentController = require(
  '../controllers/comment.controller'
);

const {
  authenticate
} = require('../middleware/auth.middleware');

// Коментарі конкретної задачі
router.get(
  '/task/:taskId',
  authenticate,
  commentController.getByTask
);

// Додати коментар (task_id і content у тілі; автор — з токена)
router.post(
  '/',
  authenticate,
  commentController.create
);

// Видалити коментар (автор або manager/admin)
router.delete(
  '/:id',
  authenticate,
  commentController.remove
);

module.exports = router;

// ----- notification.routes.js ----- const express = require('express'); const router = express.Router(); const notificationController = require(
  '../controllers/notification.controller'
);

const {
  authenticate
} = require('../middleware/ auth.middleware');

router.get(
  '/',
  authenticate,

80

Продовження додатку Б
  notificationController.getAll
);

router.post(
  '/',
  authenticate,
  notificationController.create
);

module.exports = router;

// ----- project.routes.js -----
const express = require('express'); const router = express.Router(); const projectController = require(
  '../controllers/project.controller'
);

const {
  authenticate,
  authorize
} = require('../middleware/auth.middleware');
```

```
router.get(
  '/',
  authenticate,
  projectController.getAll
);

router.post(
  '/',
  authenticate,
  authorize('manager', 'admin'),
  projectController.create
);

router.put(
  '/:id',
  authenticate,
  authorize('manager', 'admin'),
  projectController.update
);

router.delete(
  '/:id',
  authenticate,
  authorize('manager', 'admin'),
  projectController.remove
);
```



```
module.exports = router;

// ----- report.routes.js -----
const express = require('express'); const router = express.Router(); const reportController =
  require('../controllers/report.controller');

const {
  authenticate
} = require('../middleware/auth.middleware');

router.get(
  '/',
  authenticate,
  reportController.getAll
);

router.post(
  '/generate/:projectId',
  authenticate,
  reportController.generate
);

module.exports = router;

// ----- sprint.routes.js -----
const express = require('express'); const router = express.Router(); const sprintController = require(
  '../controllers/sprint.controller'
);

const {
  authenticate,
  authorize
} = require('../middleware/auth.middleware');

router.get(
  '/',
  authenticate,
```

```
  sprintController.getAll
);

router.post(
  '/',
```


Продовження додатку Б

```
  authenticate,
  authorize('manager', 'admin'),
  sprintController.create
);

router.put(
  '/:id',
  authenticate,
  authorize('manager', 'admin'),
  sprintController.update
);

router.delete(
  '/:id',
  authenticate,
  authorize('manager', 'admin'),
  sprintController.remove
);

module.exports = router;

// ----- task.routes.js -----
const express = require('express'); const router = express.Router(); const taskController = require(
  '../controllers/task.controller'
);

const {
  authenticate,
  authorize
} = require('../middleware/auth.middleware');

const validateTask = require('../middleware/task.validator');

router.get(
  '/',
  authenticate,
  taskController.getAll
);

router.post(
  '/',
  authenticate,
  authorize('manager', 'admin'),
  validateTask,
  taskController.create
);
```


Продовження додатку Б

```
router.put(
  '/:id',
  authenticate,
  authorize('manager', 'admin'),
  validateTask,
  taskController.update
);
```

```
router.delete(
  '/:id',
  authenticate,
  authorize('manager', 'admin'),
  taskController.remove
);

router.patch(
  '/:id/status',
  authenticate,
  taskController.changeStatus
);

module.exports = router;

// ----- timeLog. routes. js -----
 const express = require('express'); const router = express.Router();  const  timeLogController =  require(
  '../controllers/timeLog.controller'
);

const {
  authenticate
} = require('../middleware/auth.middleware');

router.get(
  '/',
  authenticate,
  timeLogController.getAll
);

router.post(
  '/',
  authenticate,
  timeLogController.create
);

module.exports = router;
```


ДОДАТОК В

Лістинги клієнтської частини

Лістинг В.1 — Маршрутизація застосунку (App.jsx)

```
import {
  BrowserRouter,
  Routes,
  Route,
  Navigate
} from 'react-router-dom';

import  Login from './pages/Login';  import  Dashboard  from './pages/ Dashboard';
 import  ProtectedRoute from './routes/ProtectedRoute';
 import  Analytics  from './pages/ Analytics';
 import  ProjectBoard  from './pages/  ProjectBoard';
 import Sidebar from  './components/Sidebar';
 import  Reports from './pages/Reports';
import Notifications from './pages/Notifications';

 function App() {

  return (

    &lt;BrowserRouter&gt;

      &lt;Routes&gt;

        &lt;Route
```

```
        path="/login"
        element={&lt;Login /&gt;}
      /&gt;

      &lt;Route
        path="/dashboard"
        element={
          &lt;&gt;
            &lt;Sidebar /&gt;

            &lt;ProtectedRoute&gt;
              &lt;Dashboard /&gt;
            &lt;/ProtectedRoute&gt;
          &lt;/&gt;
        }
      /&gt;

      &lt;Route
        path="/analytics"
        element={
          &lt;&gt;
            &lt;Sidebar /&gt;
```


Продовження додатку В

```
            &lt;ProtectedRoute
              roles={[
                'admin',
                'manager'
              ]}
            &gt;
              &lt;Analytics /&gt;
            &lt;/ProtectedRoute&gt;
          &lt;/&gt;
        }
      /&gt;

      &lt;Route
        path="/board"
        element={
          &lt;&gt;
            &lt;Sidebar /&gt;

            &lt;ProtectedRoute&gt;
              &lt;ProjectBoard /&gt;
            &lt;/ProtectedRoute&gt;
          &lt;/&gt;
        }
      /&gt;

      &lt;Route
        path="/reports"
        element={
          &lt;&gt;
            &lt;Sidebar /&gt;

            &lt;ProtectedRoute&gt;
              &lt;Reports /&gt;
            &lt;/ProtectedRoute&gt;
          &lt;/&gt;
        }
      /&gt;

      &lt;Route
        path="/notifications"
        element={
```

```
            &lt;&gt;
              &lt;Sidebar /&gt;

              &lt;ProtectedRoute&gt;
                &lt;Notifications /&gt;
              &lt;/ProtectedRoute&gt;
            &lt;/&gt;
          }
        /&gt;

        &lt;Route
```

86

Продовження додатку В

```
          path="*"
          element={
            &lt;Navigate
              to="/dashboard"
              replace
            /&gt;
          }
        /&gt;

      &lt;/Routes&gt;

    &lt;/BrowserRouter&gt;

  );

}

export default App;
```

Лістинг В.2 — Контекст автентифікації (AuthContext.jsx)

```
import { createContext, useState } from 'react';

export const AuthContext = createContext(); export const AuthProvider = ({ children }) =&gt; { const [user, setUser] =
  useState(
    JSON.parse(
      localStorage.getItem('user')
    )
  );

  const [token, setToken] =
    useState(
      localStorage.getItem('token')
    );

  const login = (
    token,
    user
  ) =&gt; {

    localStorage.setItem(
  'token',
  token
);
```

87

Продовження додатку В

```
localStorage.setItem(
  'user',
  JSON.stringify(user)
);

    setToken(token);
```

```
    setUser(user);

  };

  const logout = () =&gt; {

    localStorage.removeItem(
      'token'
    );

    localStorage.removeItem(
      'user'
    );

    setToken(null);
    setUser(null);

  };

  return ( &lt;AuthContext.Provider value={{ user, token, login, logout
  }}
  &gt;

    { children} &lt;/AuthContext.Provider&gt;

  );

};
```

Лістинг В.3 — Налаштування HTTP-клієнта (axios) (api.js)

```
import axios from 'axios';

const api = axios.create({ baseURL: import.meta. env. VITE_ API_URL
});

api. interceptors.request.use(
```



```
  (config) =&gt; {

    const token =
      localStorage.getItem('token');
    if (token) {
      config.headers.Authorization =
        `Bearer ${token}`;
    }

    return config;

  }
);

export default api;
```

Лістинг В.4 — Kanban-дошка з механізмом Drag-and-Drop (ProjectBoard.jsx)

```
import { useEffect, useState } from 'react'; import TaskCard from '../components/ TaskCard';
import DroppableColumn from '../components/ DroppableColumn';
import TimeLogModal from '../components/TimeLogModal';
import CreateTaskModal from '../components/CreateTaskModal';
import EditTaskModal from '../components/EditTaskModal';

import {
  DndContext
} from '@dnd-kit/core';

import api from '../api/api';
```

```
import './ProjectBoard.css';

const ProjectBoard = () =&gt; {

  const [tasks, setTasks] =
    useState([]);

  const [selectedTask, setSelectedTask] =
    useState( null); const [showCreateModal, setShowCreateModal] = useState(false);  const  [editingTask, setEditingTask] =
    useState( null);

  const handleEditTask = task =&gt; {

    setEditingTask(task);

  };
```


Продовження додатку В

```
  useEffect(() =&gt; {
    loadTasks();

  }, []);
  const loadTasks = async () =&gt; {

    try {

      const response =
        await api.get('/ tasks');

      setTasks(
        response.data
      );

    } catch (error) {

      console.error( error);

    }

  };

  const handleDragEnd =
  async ({ active, over }) =&gt; {

    if (!over) return;

    const taskId =
      Number(active.id);

    const newStatus =
      over.id;

    console.log(
      'taskId:',
      taskId
    );

    console.log(
      'newStatus:',
      newStatus
    );

    try {
```

```
      const response =
        await api.patch(
          `/tasks/${taskId}/status`,
          {
            status:
```


Продовження додатку В

```
              newStatus
            }
          );

        console.log(
       'PATCH response:',
          response.data
        );

        setTasks(prev =&gt;
          prev.map(task =&gt;
            task.id === taskId
              ? {
                  ...task,
                  status:
                    newStatus
                }
              : task
          )
        );

      } catch (error) {

        console.error(
          'PATCH ERROR:',
          error.response?.data
        );

        console.error(error);

      }

    };

    const renderColumn =
(
    title,
    status
) =&gt; (

    &lt;DroppableColumn
    id={status}
    title={`${title} (${
      tasks.filter(
        task =&gt;
          task.status === status
      ).length
    })`}
    className={`column ${status}`}
&gt;

      {tasks
```


Продовження додатку В

```
        .filter(
          task =&gt;
```

```
          task.status === status
        )
        .map(task =&gt; (
          &lt;TaskCard
            key={task.id}
            task={task}
            onClick={setSelectedTask}
            onEdit={handleEditTask}
          /&gt;

        ))}

    &lt;/DroppableColumn&gt;

  );

    return (

      &lt;DndContext
        onDragEnd={
          handleDragEnd
        }
      &gt;

        &lt;div className="board"&gt;

          &lt;div className="board-header"&gt;

            &lt;h1&gt;
              Project Board
            &lt;/h1&gt;

            &lt;button
              className="new-task-btn"
              onClick={() =&gt;
                setShowCreateModal(true)
              }
            &gt;
              + New Task
            &lt;/button&gt;

          &lt;/div&gt;

          &lt;div className="columns"&gt;

            {renderColumn(
              'To Do',
              'todo'
            )}
```



```
            {renderColumn(
              'In Progress',
              'in_progress'
            )}

            {renderColumn(
              'Testing',
              'testing'
            )}

            {renderColumn(
              'Done',
              'done'
            )}
```

</div>

</div>

{selectedTask && (

<TimeLogModal
task={selectedTask}
onClose={() => setSelectedTask(null)}
/>

)}

{showCreateModal && (

<CreateTaskModal
onClose={() =>
setShowCreateModal(false)
}
onCreated={loadTasks}
/>

)}

{editingTask && (

<EditTaskModal
task={editingTask}
onClose={() =>
setEditingTask(null)
}
onUpdated={loadTasks}
/>

)}

</DndContext>

Продовження додатку В

);

};

export default ProjectBoard;

Лістинг В.5 — Стовпець-зона скидання (DroppableColumn.jsx)

```
import {
  useDroppable
} from '@dnd-kit/core';

const DroppableColumn = ({
  id,
  title,
  children,
  className
}) =&gt; {

  const { setNodeRef
  } = useDroppable({
    id
  });

  return ( &lt;div ref={setNodeRef} className={ className}
    &gt;
```

```
      &lt;h2&gt;
        {title}
      &lt;/h2&gt;

      {children}

    &lt;/div&gt;

  );

};

export default DroppableColumn;
```

Лістинг В.6 — Картка завдання (перетягування) (TaskCard.jsx)

```
35 import { useDraggable } from '@dnd-kit/core'; import './TaskCard.css';

const TaskCard = ({
  task,
  onClick,
```



```
  onEdit
}) =&gt; {
35 const { attributes, listeners, setNodeRef, transform } = useDraggable({ id: task.id }); const style = transform
  ? {
      transform: `translate3d(
        ${transform.x}px,
        ${transform.y}px,
        0
      )`
    }
  : undefined;

  return (

  &lt;div
    ref={setNodeRef}
    style={style}
    className="task-card"
    onDoubleClick={() =&gt;
      onClick(task)
    }
  &gt;

    &lt;div className="task-header"&gt;

      &lt;div className="task-header-left"&gt;

        &lt;div
          className="drag-handle"
          {...listeners}
          {...attributes}
        &gt;
          ⋮⋮
        &lt;/div&gt;

        &lt;h4&gt;
          {task.title}
        &lt;/h4&gt;

      &lt;/div&gt;

      &lt;button
        className="edit-task-btn"
```

Продовження додатку В

```
onClick={(e) =&gt; {

            e.preventDefault();
            e.stopPropagation();
            if (onEdit) {
              onEdit(task);
            }

          }}
        &gt;

        &lt;/button&gt;

      &lt;/div&gt;

      &lt;p className="task-sp"&gt;
        SP: {task.story_points}
      &lt;/p&gt;

      &lt;p className="task-deadline"&gt;
        Deadline: {task.deadline}
      &lt;/p&gt;

        {task.executor &amp;&amp; (

          &lt;div className="task-assignee"&gt;
              {task.executor.name}
          &lt;/div&gt;

        )}

      &lt;small&gt;
        Double click for details
      &lt;/small&gt;

      &lt;div&gt;

        &lt;span
          className={`priority ${task.priority}`}
        &gt;
          {task.priority}
        &lt;/span&gt;

      &lt;/div&gt;

    &lt;/div&gt;

  );

};
```


Продовження додатку В

export default TaskCard;

Лістинг В.7 — Модуль обліку робочого часу (TimeLogModal.jsx)

```
import { useState } from 'react';
import api from '../api/api';
import './TimeLogModal.css';
const TimeLogModal = ({
  task,
  onClose
}) =&gt; {
```

```
  const [hours, setHours] =
    useState('');

  const [comment, setComment] =
    useState('');

  const saveLog =
    async () =&gt; {

      try {

        const user =
          JSON.parse(
            localStorage.getItem(
              'user'
            )
          );

        await api.post(
          '/time-logs',
          {
            hours,
            comment,

            log_date:
              new Date()
                .toISOString()
                .split('T')[0],

            task_id:
              task.id,

            user_id:
              user.id
          }
        );

        alert(
          'Time log saved'
        );
```


Продовження додатку В

```
        onClose();

      } catch (error) {

        console.error(error);

      }

    };

  return (

    &lt;div className="modal-overlay"&gt;
      &lt;div className="modal"&gt;

        &lt;h2&gt;
          Time Tracking
        &lt;/h2&gt;

        &lt;div className="task-info"&gt;

          &lt;h3&gt;
```

```
          {task.title}
        &lt;/h3&gt;

        &lt;div className="task-meta"&gt;

          &lt;p&gt;
            Priority: {task.priority}
          &lt;/p&gt;

          &lt;p&gt;
            Story Points: {task.story_points}
          &lt;/p&gt;

        &lt;/div&gt;

      &lt;/div&gt;

      &lt;input
        className="time-input"
        type="number"
        step="0.5"
        placeholder="Hours Worked"
        value={hours} onChange={e =&gt;
          setHours(
            e.target.value
          )
        }
      /&gt;
```


Продовження додатку В

```
    &lt;br /&gt;
    &lt;br /&gt;

      &lt;textarea
        className="comment-input"
        placeholder="Describe what was completed..."
        value={comment}
        onChange={e =&gt;
          setComment(
            e.target.value
          )
        }
      /&gt;

    &lt;br /&gt;
    &lt;br /&gt;

      &lt;div className="modal-buttons"&gt;

        &lt;button
          className="save-btn"
          onClick={saveLog}
        &gt;
          Save Log
        &lt;/button&gt;

        &lt;button
          className="close-btn"
          onClick={onClose}
        &gt;
          Cancel
        &lt;/button&gt;

      &lt;/div&gt;
```

```
      &lt;/div&gt;
    &lt;/div&gt;

  );

};

export default TimeLogModal;
```

Лістинг В.8 — Діаграма завантаженості (Recharts) (WorkloadChart.jsx)

```
import { BarChart, Bar, XAxis, YAxis,
  Tooltip,
  CartesianGrid,
```



```
  ResponsiveContainer
} from 'recharts';

const WorkloadChart = ({ data }) =&gt; {

return (

    &lt;ResponsiveContainer width="95%"
      height={ 350}
    &gt;

      &lt; BarChart
        data={ data} &gt; &lt;CartesianGrid strokeDasharray= "3 3"
        /&gt;

        &lt;XAxis
          dataKey="name"
        /&gt;

        &lt;YAxis /&gt;

        &lt;Tooltip
          cursor={false}
          contentStyle={{
            background: '#1e293b',
            border: 'none',
            borderRadius: '10px',
            color: '#fff'
          }}
        /&gt;

        &lt;Bar
          dataKey="total_hours"
          fill="#0274f7"
        /&gt;

      &lt;/BarChart&gt;

    &lt;/ResponsiveContainer&gt;

  );

};

export default WorkloadChart;
```


Лістинг В.9 — Кругова діаграма статусів (Recharts) (TaskStatusChart.jsx)

```
import {
```

```
  PieChart,
  Pie,
  Cell,
  Tooltip,
  ResponsiveContainer,
  Legend
} from 'recharts';
const COLORS = [
  '#3b82f6',
  '#f59e0b',
  '#8b5cf6',
  '#22c55e'
];

const TaskStatusChart = ({
  data
}) =&gt; {

  return (

    &lt;ResponsiveContainer
      width="100%"
      height={350}
    &gt;

      &lt;PieChart&gt;

        &lt;Pie
          data={data}
          dataKey="value"
          nameKey="name" outerRadius={ 120}
          label
        &gt;

          {data.map(
            (entry, index) =&gt; ( &lt;Cell key={index} fill={ COLORS[ index % COLORS.length
                ]
              }
            /&gt;

          )
```



```
        )}

        &lt;/Pie&gt;

        &lt;Tooltip /&gt;

        &lt;Legend /&gt;

      &lt;/PieChart&gt;

    &lt;/ResponsiveContainer&gt;

  );

};

export default TaskStatusChart;
```

Тема дипломної роботи:
«Розробка інформаційної системи для управління проєктами компанії»

Обсяг пояснювальної записки 59 аркушів.

Перелік графічного матеріалу:
1. КБР.ІСТ – 13.00.00.000 С1. Варіанти використання інформаційної системи управління проєктами.
2. КБР.ІСТ – 13.00.00.000 С1. Діаграма компонентів інформаційної системи. управління проєктами
3. КБР.ІСТ – 13.00.00.000 С1. Концептуальна ER-діаграма бази даних.
4. КБР.ІСТ – 13.00.00.001. Екранні форми інтерфейсу інформаційної системи управління проєктами

Дата завершення роботи: червня 2026 р.
Підпис студента-дипломника _____ Осадчук В. Ю.

КРБ.ІСТ-13.00.00.000С1
С1

КРБ.ІСТ-13.00.00.000С1
С1

КРБ.ІСТ-13.00.00.000С1
С1

КРБ.ІСТ-13.00.00.001