Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

ІНТЕГРОВАНА КОМП'ЮТЕРНА СИСТЕМА ДЛЯ ОПЕРАТИВНОЇ ВЗАЄМОДІЇ ПЕРСОНАЛУ ВИРОБНИЧОГО ПІДПРИЄМСТВА З ВИГОТОВЛЕННЯ МЕБЛІВ

Виконав: студент 2 курсу, групи 2КІ-23м
спеціальності 123 – «Комп'ютерна інженерія»
_____ Палько А. С.
Керівник: к.т.н., доц. каф. ОТ
_____ Савицька Л. А.
«___» _____ 2024 р.
Опонент: к.ф-м.н., доц. каф. МІБС
_____ Шиян А.А.
«___» _____ 2024 р.

Допущено до захисту
Завідувач кафедри ОТ
_____ д.т.н., проф. Азаров О. Д.
«___» _____ 2024 р.

ВНТУ 2024

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Галузь знань — Інформаційні технології

Освітній рівень — магістр

Спеціальність — 123 Комп'ютерна інженерія

Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

_____ О. Д. Азаров

« 18 » вересня 2024 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Пальку Артуру Сергійовичу

1 Тема роботи «Інтегрована комп'ютерна система для оперативної взаємодії персоналу виробничого підприємства з виготовлення меблів» керівник роботи Азаров О. Д., д.т.н., проф. каф. ОТ, затверджено наказом вищого навчального закладу № 310 від 17.09.2024 року.

2 Строк подання студентом роботи 16.12.2024 р.

3 Вихідні дані до роботи: призначення системи — оперативна взаємодія персоналу меблевого виробничого підприємства для підвищення ефективності комунікації та координації робочих процесів, тип взаємодії — текстовий обмін повідомленнями, створення завдань та призначення їх виконавців, організація системи — клієнт-серверна архітектура з веб-інтерфейсом.

4 Зміст текстової частини (перелік питань, які потрібно розробити): вступ, вплив корпоративної етики підприємства на вимоги щодо функціоналу

розроблюваного додатку, опис інструментів розробки, архітектура та програмна реалізація застосунку, тестування застосунку, економічна частина.

5 Перелік графічного матеріалу: використано 10 таблиць та 24 рисунки.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		видав	прийняв
1-4	Савицька Людмила Анатоліївна, к.т.н, доц.		
5	Небава Микола Іванович, к.е.н., проф.		

7 Дата видачі завдання 18.09.2024 р. Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Примітка
1	Постановка задачі	20.09.2024	Виконано
2	Огляд існуючих рішень	24.09.2024	Виконано
3	Теоретичні дослідження	29.09.2024	Виконано
4	Розробка структурної схеми	03.10.2024	Виконано
5	Розрахунок аналогової частини	07.10.2024	Виконано
6	Розробка принципової схеми	13.10.2024	Виконано
7	Моделювання роботи системи	20.10.2024	Виконано
8	Розрахунок економічної частини	27.10.2024	Виконано
9	Оформлення ПЗ та ілюстративного матеріалу	03.11.2024	Виконано
10	Виконання магістерської кваліфікаційної роботи	10.11.2024	Виконано
11	Перевірка якості виконання МКР та усунення недоліків	14.12.2024	Виконано
12	Підписи у керівника, опонента, нормоконтролера	15.12.2024	Виконано
13	Перевірка «антиплагіат»	17.11.2024	Виконано
14	Попередній захист	18.11.2024	Виконано

Студент Палько А. С.

Керівник к.т.н., доц. каф. ОТ Савицька Л. А.

АНОТАЦІЯ

УДК 004

Палько А.С Інтегрована комп’ютерна система для оперативної взаємодії пересоналу виробничого підрпиємства з виготовлення меблів.

Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2024, 120 с.

На укр. мові. Бібліогр.: 31 назв; рис.: 24; табл. 10.

У магістерській роботі здійснено огляд сучасних підходів до організації внутрішньої комунікації працівників на підприємствах з метою впровадження власного програмного рішення.

Проведено детальний аналіз та опис інструментів, що сприяють вирішенню поставленого завдання.

Розроблено технічну архітектуру та користувацький інтерфейс інтегрованої комп'ютерної системи.

Ключові слова: інтенгрована комп’ютерна система, C#, ASP .NET Core, JavaScript, React, Entity Framework, MS Sql Server.

ABSTRACT

УДК 004

Palko A.S. Integrated computer system for operational interaction of personnel of a furniture manufacturing enterprise. Master's thesis on specialty 123 – Computer Engineering, Vinnytsia: VNTU, 2024, 120 p.

In Ukrainian speech Bibliography: 31 titles; Fig.: 24; tables 10.

This master's thesis provides an overview of modern approaches to organizing internal communication among employees in enterprises, with the aim of implementing a custom software solution.

A detailed analysis and description of tools that facilitate the resolution of the defined task have been conducted.

The technical architecture and user interface of the integrated computer system have been developed.

Keywords: integrated computer system, C#, ASP.NET Core, JavaScript, React, Entity Framework, MS SQL Server.

Зміст

ВСТУП

Меблеві підприємства займаються виготовленням меблів та супутніх товарів із дерева, які задовольняють потреби населення і бізнесу. Вони виробляють широкий асортимент продукції, що включає як домашні, так і офісні меблі, забезпечуючи затишок і функціональність у житлових та робочих просторах.

Підприємства із виготовлення меблів відносяться до категорії промислових підприємств, а конкретніше — до обробної промисловості. Що ще більше підкреслює їх значимість для економіки, оскільки виготовлення готової продукції майже завжди є вигіднішим, аніж продаж самої сировини. Під час розширення циклу виробництва до випуску кінцевих товарів збільшується кількість робочих місць, у значної частини працівників з'являється можливість для підвищення їх кваліфікації, щоб перейти на більш технологічний виробничий процес, що врешті має позитивно вплинути на рівень їх заробітної плати. Виграш отримує також і саме підприємство, оскільки зростає його впізнаваність серед покупців, адже здебільшого саме товари, а не сировина лежать на полицях магазинів, постачаючи якісні товари можна затаврувати власний бренд на ринку, що буде викликати довіру та привертати все більше уваги. Як для працівників, так і для самого підприємства та його власників, ми без зайвих вагань прийшли до очевидного висновку: матеріальна вигода — це ключова мета. І це відповідно логічно, оскільки головним завданням будь-якого підприємства, незалежно від галузі, є отримання прибутку.

Але якщо поглянути глибше, то отриманий прибуток буде також витрачений на покращення умов праці на підприємстві, його благоустрій, що має в хорошому ключі вплинути на задоволеність людей своєю роботою, що, у свій час, створює їх мотивацію та відданість справі, забезпечуючи ще більшу продуктивність і ефективність роботи. Відійшовши на мить від теми саме меблевих підприємств, варто згадати що існує низка галузей що неминуче завдають шкоду навколишньому середовищу під час своєї діяльності, яку не можна припинити, бо суспільство не може відмовитись від продуктів їх

діяльності, крім того вони не порушують норми щодо здійснення господарської діяльності будь-якого виду що не заборонена законодавством. Наприклад, у процесі виробництва хімічних речовин, пластмас, фарб, добрив і пестицидів відбуваються викиди токсичних сполук, які можуть забруднювати повітря, воду й ґрунт. Водночас під час виготовлення цементу утворюється велика кількість пилу та парникових газів, які шкодять атмосфері й можуть викликати респіраторні захворювання. Також лісозаготівельна галузь, яка може або бути у прямому підпорядкуванні місцевим меблевим підприємствам, що маючи зароблену роками репутацію, будуть контролювати цей процес, виступаючи їх постачальниками, або ж бездумно вирубувати ліси для необмеженого продажу за кордон іншим підприємствам, незважаючи на екологічні ризики, що можуть виникнути внаслідок таких дій, як вплив на ріст дерев і доступність лісових ресурсів. Це негативно вплине на клімат і ускладнить у майбутньому сам процес заготівлі, спричинить конфлікти з місцевими громадами через землекористування або вплив на довкілля.

Тобто чим більший дохід підприємства, тим більша ймовірність, що воно буде приділяти увагу охороні навколишнього середовища. Це пов’язано з тим, що зростання прибутків дозволяє інвестувати в екологічно чисті технології, покращення виробничих процесів та заходи зі зменшенням викидів. Крім того, фінансові можливості підприємства можуть забезпечити реалізацію ініціатив, спрямованих на збереження природних ресурсів та покращення екологічної ситуації в регіоні. Таким чином, підприємства з високим доходом мають більшу відповідальність і можливості для позитивного впливу на довкілля.

Меблеві підприємства, подібно до інших виробників готової продукції, створюють значний внесок у науку та технологічний прогрес. Розширення технологічних процесів, які вони впроваджують, має далекосяжні дослідження для багатьох аспектів. Крім того, меблеві компанії активно співпрацюють з науковими установами, що стимулює обмін знаннями та досвідом. Такі партнерства сприяють розвитку нових технологій та досліджень, які можуть бути

адаптовані лише в меблевій галузі, а й у суміжних секторах, таких як будівництво, дизайн інтер’єру.

У своїй магістерській роботі я хочу покращити продуктивність саме цього типу підприємства створивши веб-додаток що представлятиме собою інтегровану комп’ютерну систему для взаємодії його працівників, яка поєднає різні відділи, такі як: дослідницький, господарсько-експлуатаційний, кадровий, юридичний відділ, бухгалтерію, відділ інформаційних технологій. Та дозволить комунікацію між працівниками підприємства, розсилання завдань по відділам та контроль за їх виконанням, проведення опитувань на різні теми для отримання зворотної відповіді від колег щодо тих чи інших управлінських рішень.

Метою дослідження є автоматизація процесів взаємодії працівників меблевого підприємства шляхом впровадження власного механізму та аналізу відповідних алгоритмів.

Задачі дослідження:

— проаналізувати існуючі аналоги корпоративних платформа;

— створити власний додаток для взаємодії працівників підприємства з виготовлення меблів;

— здійснити обґрунтування доцільності здійснення нового наукового рішення, виконати розрахунок економічних затрат для створення програмного додатку, його впровадження та обслуговування.

Об’єкт дослідження — сукупність процесів, систем та структур, що забезпечують взаємодію працівників у контексті виробництва меблів.

Методи дослідження: аналіз вимог до розроблюваної інтегрованої комп’ютерної системи для взаємодії користувачів підприємства із виготовлення меблів, використання мов програмування та фреймворків для створення веб-додатку.

Новизна одержаних результатів полягає в розробці веб-додатку, що об'єднує ключові підрозділи підприємства для управління внутрішніми комунікаціями, впровадженні механізмів опитувань для покращення процесу прийняття управлінських рішень та створенні власної системи комунікації в

межах локальної мережі для підвищення інформаційної безпеки та збереження конфіденційності корпоративних даних.

Практичне значення одержаних результатів:

— отримані результати забезпечують значне підвищення ефективності та оптимізацію управлінських процесів на меблевому підприємстві і запропонований веб-додаток сприятиме покращенню комунікації між відділами, забезпечить централізоване управління завданнями та контроль за їх виконанням, що дозволить мінімізувати адміністративні витрати часу та ресурсів;

— інструменти для збирання зворотного зв’язку через опитування дозволять керівництву оперативно отримувати інформацію про настрій працівників та ефективність прийнятих управлінських рішень.

Апробація матеріалів роботи виконана під час LIV конференції.

Публікація за темою роботи: А. С. Палько, Л. А. Савицька. ІНТЕГРОВАНА КОМП’ЮТЕРНА СИСТЕМА ДЛЯ ОПЕРАТИВНОЇ ВЗАЄМОДІЇ ПЕРСОНАЛУ ВИРОБНИЧОГО ПІДПРИЄМСТВА З ВИГОТОВЛЕННЯ МЕБЛІВ. ВНТКП ВНТУ. Факультет інформаційних технологій та комп'ютерної інженерії. Матеріали LIV Всеукраїнської науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2025)[1].

1 ВПЛИВ КОРПОРАТИВНОЇ ЕТИКИ ПІДПРИЄМСТВА НА ВИМОГИ ЩОДО ФУНКЦІОНАЛУ РОЗРОБЛЮВАНОГО ДОДАТКУ

1.1 Поняття корпоративної етики

Люди, будучи за своєю природою соціальними істотами, автоматично прагнуть до інтеграції в колектив, що змушує їх адаптуватися до встановлених норм і правил поведінки. Під впливом спільних цілей, взаємодій та комунікації з колегами вони поступово приймають корпоративну етику, яка визначає не лише професійні, але й соціальні стандарти поведінки. Цей процес відбувається природно, адже, щоб залишатися ефективним членом команди, людина повинна враховувати не лише особисті інтереси, але й колективні, а також відповідати очікуванням і цінностям організації.

Корпоративна етика — це система цінностей, норм та правил, які визначають поведінку співробітників на підприємстві. Корпоративний етикет може бути частково закріплений в офіційних документах, але часто існує й у неформальному вигляді як сукупність негласних правил та норм поведінки [2]. Ці правила можуть охоплювати не лише поведінку співробітників на робочому місці, але й їхній зовнішній вигляд, а в окремих випадках — навіть вимагати дотримання єдиної позиції з певних питань. Оскільки кожен працівник є частиною загального іміджу компанії, його слова або дії можуть опинитися під увагою громадськості чи ЗМІ. Якщо він опиниться в центрі подій або дасть коментар журналісту, його висловлювання можуть набрати широкого розголосу та автоматично асоціюватимуться з компанією, у якій він працює, що може вплинути на її репутацію.

Умовними правилами корпоративної взаємодії можуть бути такі елементи, як перехід від офіційного звертання "Ви" на більш дружнє "ти", що часто відбувається після певного часу спільної роботи, проте деякі компанії відразу приймають неформальний стиль спілкування, переходячи на "ти" з усіма співробітниками. Це допомагає створити атмосферу рівності, зменшити напругу та полегшити адаптацію новачків у колективі. Також важливу роль у зміцненні

командного духу відіграють традиції святкування днів народжень колег, які допомагають створити атмосферу підтримки та взаєморозуміння. Крім того, організація корпоративних вечірок або заходів з певних нагод, таких як досягнення важливих результатів чи відзначення свят, сприяє налагодженню неформальних стосунків, що в свою чергу добре впливає на взаємодію в робочих процесах і загальну продуктивність команди. До правил хорошого тону також можна віднести не порушення особистого простору колеги, коли ми не намагаємось вирішити робочі питання у його позаробочий час. Можливо, нам дійсно з радістю допоможуть цього разу, чи просто хочуть справити хороше враження повної відданості роботі, але це не зможе продовжуватись вічно. Робота займає значну частину життя більшості людей, і важливо мати можливість відпочивати, щоб зберігати ефективність. Тому в певний момент ми ризикуємо зіткнутися з непорозуміннями щодо такої поведінки, що можуть призвести до конфліктів. У більшості колективів вітається взаємодопомога, але в цьому питанні важливо зберігати баланс. Допомога має бути незначною, щоб не займати багато часу колеги, не відволікати його від виконання власних завдань і не створювати відчуття перекладання обов'язків на нього з боку інших. Також не прийнято обговорювати заробітну плату з колегами, оскільки це може призвести до непорозумінь, конфліктів чи навіть зниження мотивації в колективі. Відповідно до корпоративних правил, такі питання зазвичай розглядаються під час особистих зустрічей із керівництвом, що дозволяє зберігати конфіденційність та уникати зайвих обговорень у робочому середовищі. Обговорення зарплат може викликати порівняння, які не завжди є конструктивними, адже кожен працівник має свої унікальні навички, досвід і внесок у компанію. Це може призвести до незадоволення або навіть конфліктів між співробітниками, особливо якщо хтось відчуває, що отримує менше, ніж заслуговує. Тому збереження цієї інформації в межах особистих бесід допомагає підтримувати позитивну атмосферу в колективі, а також заохочує відкритий діалог між працівниками та керівництвом стосовно розвитку кар’єри та підвищення кваліфікації. Таким чином, корпоративна культура, яка забороняє

обговорення заробітної плати, може стати важливим елементом для створення здорового робочого середовища та підтримки колективної згуртованості.

Від працівників на роботі очікується охайний зовнішній вигляд, але в сучасному світі все частіше відмовляються від традиційної практики носити ділові костюми, особливо в компаніях з неформальною атмосферою. Це свідчить про зміну підходів до професійного іміджу, де комфорт та індивідуальність починають відігравати важливу роль, не зменшуючи при цьому стандартів професіоналізму. Проте, якщо мова йде про працівника на конвеєрі, носіння спеціальної форми вимагається не лише для дотримання виробничого процесу, а й для забезпечення максимального захисту його здоров'я. Крім того, споживачі хочуть бачити, що технології підприємства дотримуються на всіх етапах виробництва, а не лише на фотографіях, де всі заздалегідь підготувалися до зйомки, адже це допомагає їм довіряти компанії. Багато клієнтів банків справді вважають, що працівники повинні носити ділові костюми, оскільки це підкреслює професіоналізм і серйозність установи. Офіційний дрес-код може створити враження надійності та довіри, що особливо важливо в фінансовому секторі, де клієнти часто прагнуть впевненості в тому, що їхні гроші в безпеці. Вплив татуювань та пірсингу на корпоративний етикет, може бути досить суттєвим. У багатьох організаціях, традиційні стандарти зовнішнього вигляду можуть вимагати від співробітників дотримання консервативного стилю. Татуювання та пірсинг можуть сприйматися як відображення неформального підходу, що не завжди відповідає вимогам офіційного дрес-коду. В державній службі, де важливими є довіра та професійність, наявність видимих татуювань або пірсингу може викликати неоднозначну реакцію у громадськості та колег. Це може вплинути на сприйняття компетентності співробітника, що, в свою чергу, може позначитися на загальному іміджі організації. Тому багато державних установ впроваджують певні правила щодо зовнішнього вигляду, які обмежують або забороняють носіння яскравих татуювань і пірсингу, щоб підтримувати професійний стандарт. Проте, з часом суспільство стає більш відкритим до різноманітності в самовираженні, і деякі організації можуть

адаптувати свої підходи, дозволяючи співробітникам демонструвати індивідуальність, за умови, що це не порушує загальноприйняті норми поведінки та не заважає виконанню службових обов'язків.

Продовжуючи тему державної служби, зручно продемонструвати правило єдності думок у корпоративній поведінці щодо певних питань. Чиновники мають виступати з однаковою позицією щодо політичних чи державних рішень, незалежно від власної думки, щоб не порушувати єдність державної політики. Державні управлінці зобов'язані не обговорювати внутрішні питання або стратегії державних органів з представниками преси чи громадськості без відповідного дозволу. При спілкуванні з громадськістю чи пресою чиновники повинні дотримуватися єдиної лінії, використовуючи узгоджені формулювання щодо державної політики та рішень. Усі висловлювання держслужбовців повинні відповідати офіційній позиції організації чи міністерства, в якому вони працюють. Особиста думка у публічному просторі має бути або узгоджена, або повністю виключена. Правило єдності думок у корпоративній етиці, яке вимагає від працівників дотримання спільної позиції з певних питань, звісно, може бути порушене за умов, коли особиста думка або дії співробітника можуть бути обґрунтовані необхідністю відстоювати інтереси громадськості чи служити вищим цілям, які виходять за межі звичайних корпоративних норм. Державні службовці, перш за все, несуть відповідальність перед громадянами, адже їх головне завдання — відстоювати інтереси суспільства. Тому порушення певних правил корпоративної поведінки, якщо воно не суперечить фактам і спрямоване на благо народу, може мати навіть позитивні наслідки. Проте, якщо такі дії виявляться помилкою, це може призвести до критики, і тоді можуть бути згадані й інші недоліки цього працівника, що негативно позначиться на його репутації. Якщо ж негативні наслідки виявляться серйозними, це може загрожувати навіть звільненням або притягненням до відповідальності.

Меблеві підприємства, за своєю специфікою, зосереджені на виробничих процесах, дизайні, логістиці та співпраці з іншими компаніями, що потребує дотримання чіткого й професійного підходу в комунікаціях. Це також стосується

майбутнього додатку, де варто використовувати діловий стиль мовлення в інтерфейсі. Крім того, користувачам необхідно забезпечити можливість встановлення лише офіційної фотографій на аватарі, підкреслюючи важливість професійного підходу у внутрішніх комунікаціях. Такий підхід сприятиме створенню атмосфери відповідальності та підвищенню рівня довіри між працівниками.

У колективі можуть виникнути суперечки щодо розподілу завдань або підходів до виконання роботи. Якщо ці питання залишаються невирішеними, емоції можуть загострюватися, і розбіжності починають викликати напруження серед колег. Непорозуміння може перетворитися на критику, а критика — на образи. У такій атмосфері довіра зменшується, а спілкування ускладнюється. Люди можуть почати відстоювати свої позиції агресивніше, що призводить до ще більшого загострення ситуації. Врешті-решт, ці накопичені розбіжності і напруга можуть викликати конфлікт — зіткнення протилежних інтересів і поглядів, що супроводжується активними діями і складними колізіями.

Причини конфліктів:

— господарсько-організаційні — неправильна організація праці, неправильна організація заробітної плати;

— соціально-професійні — недосконаліть системи добору і розстановки кадрів; обмеження просування працівника на вищі посади; правова закріпленість працівника за посадою при низькому рівні його ділових якостей;

— соціально-демографічні — порушення вікової структури колективу; порушення структури за ознакою статі;

— соціально-психологічні — психотипологічна несумісність; морально-духовна несумісність [3].

Досвід свідчить, що легше уникнути конфлікту, ніж його вирішити. Тому профілактика конфліктів є такою ж важливою, як і здатність конструктивно їх розв’язувати. Основною метою є недопущення дій, які можуть призвести до виникнення суперечностей у різних сферах взаємодії. У таких ситуаціях одна

сторона може бути активною, а інша — пасивною, причому дії кожної з них можуть суттєво впливати на розвиток конфлікту.

Стратегії вирішення конфліктів можлива декількома шляхами.

Стиль «Конкуренція» можна описати як «Для моєї перемоги тобі доведеться зазнати поразки». У цьому випадку особа або сторона орієнтована на задоволення власних інтересів, нехтуючи потребами та інтересами інших учасників. Така стратегія може включати нав'язування своєї думки на шкоду іншим або активний опір діям супротивника Зазвичай, той, хто обирає цей стиль, використовує позиції сили, що випливають з його статусу, професійного досвіду або вміння переконувати.

Стиль «Пристосування» можна охарактеризувати як «Щоб ти здобув перемогу, я повинен поступитися». Ця стратегія, відома як «згладжування», передбачає, що інтереси та потреби іншої сторони мають пріоритет, навіть якщо це відбувається за рахунок власних інтересів.

Стиль «Компроміс» можна описати як «Щоб кожен з нас отримав певну вигоду, кожен мусить чимось поступитися». Ця стратегія полягає в пошуку рішень, які є прийнятними для обох сторін, але водночас жодна з них не отримує повного задоволення своїх потреб.

Стиль «Співпраця» можна сформулювати як «Щоб ти виграв, я також повинен виграти». Ця стратегія, відома також як підхід до вирішення проблем, полягає у пошуку взаємоприйнятних рішень, що передбачає прагнення зрозуміти потреби та занепокоєння іншої сторони.

Стиль «Уникання» можна описати як «Мені байдуже, виграєш ти чи програєш, я просто не беру участі в цьому». Ця стратегія, відома як «утримання від дій» або «уникання», полягає в тому, що особа не відстоює свої інтереси та потреби; конфлікт залишається без обговорення, а його вирішення відтерміновується за допомогою обхідних маневрів [4].

Тож, ефективне управління конфліктами передбачає врахування контексту, можливостей та ризиків. На вибір стратегії врегулювання конфліктів впливає чимало факторів, у тому числі час, довіра сторін, важливість питання.

Важливо свідомо підходити до вибору стратегії і не забувати про довгострокову перспективу. Зважаючи на вище викладене, доцільно впровадити механізм для обговорення поставлених завдань між користувачами. Це дозволить кожному користувачу висловити свою думку, поділитися ідеями або зауваженнями щодо завдань, що стоять перед ними. Така можливість сприятиме відкритості в комунікації та забезпечить простір для взаємодії, де кожен голос може бути почутий. Це також допоможе уникнути непорозумінь і створити атмосферу довіри та взаємоповаги між учасниками обговорення.

1.2 Порівняльний огляд існуючих рішень для здійснення комунікації — Jira, Teams та АСКОД

Цифрові інструменти зв’язку щільно увійшли до нашого повсякденного життя, перетворивши спосіб, яким ми спілкуємось, співпрацюємо та обмінюємось інформацією. Від моменту, коли ці технології почали розвиватися, вони змінили наше сприйняття комунікації, зробивши її швидшою, зручнішою та доступнішою. Сучасні засоби цифрового спілкування, які ми знаємо сьогодні, почали набувати широкого поширення з початку 2000-х років і продовжують еволюціонувати, відповідаючи потребам користувачів у швидкості, зручності та ефективності.

Першим значним кроком у цьому напрямку стала електронна пошта, яка, незважаючи на розвиток нових технологій, все ще залишається актуальною. Вона не тільки використовує для ділового спілкування, а й інтегрується з іншими сервісами, що дозволяє, наприклад, швидко скинути пароль або обмінюватися важливими відомостями. Проте електронна пошта часто не підходить для динамічного спілкування. Користувачі не завжди знають про активність співрозмовника, і лише після отримання відповіді можна дізнатися, чи було повідомлення прочитано. Крім того, електронна пошта не має функцій збереження контактів, як у телефоні, і щоразу потрібно вводити громіздкі адреси поштових скриньок, що можна уповільнювати комунікацію.

Уже із середин першого десятиліття з’явилися соціальні мережі. Які виправили всі вищеописані недоліки поштових сервісів. Проте вони більшою мірою позиціонували себе як засоби для спілкування з друзями та близькими, уже згодом розробники цих сервісів проклали ідею про те що у соціальних мережах можна вести офіційні сторінки різних установ. Окрім цього у соцмережах можна ділитися моментами із власного життя, публікуючи на своїх сторінці різні дописи у вигляді текстових повідомлень, фото чи відео. Перед початком наступного десятиліття відбувся неймовірний розвиток мобільних телефоні, коли вони перетворилися у новий пристрій — смартфон. Смартфони стали виконувати деякі функції комп’ютера, але уже достатні для того щоб здійснювати цифрове спілкування через них, оскільки вони є компактнішими. Хоча через смартфон можна були зайти на свою сторінку у соціальній мережі, проте ці пристрої перенесли спілкування у цифровому просторі на зовсім невимушені теми, крім того на початку потужності смартфонів не були такими великими. Тому можна сказати що ці фактори стали стимулом для розвитку полегшених версій соціальних мереж — месенджерів у яких було менше, аудіо, відео, зображень основний фокус робився саме на текстових повідомленнях. Середина 2010-2020 років ознаменувалася підвищення доступності смартфонів із якісними камерами. Це дозволило користувачам публікувати знімки в соціальних мережах безпосередньо зі своїх пристроїв. Завдяки цьому виник новий тип соціальної мережі, в якому основний акцент був зроблений на публікаціях візуальних дописів. Хоча текстове спілкування та дописи також залишалися можливими, нові платформи значно змінили формат взаємодії. Згодом ці візуально орієнтовані соціальні мережі навіть трохи перевищили популярність своїх попередників.

І, нарешті, на початку третього десятиліття масово почали використовуватися платформи для перенесення роботи та навчання в цифровий простір. Поштовхом для швидкого розвитку цих сервісів стала пандемія коронавірусу, яка вимагала збереження дистанції між людьми. Нові сервіси представляли собою щось на кшталт соціальної мережі, але з акцентом на роботу

та навчання. В них було мінімум особистих фотографій, а більше робочих завдань та інструментів для їх вирішення та обговорення.

Перш ніж розпочинати розробку власного програмного продукту, варто уважно розглянути вже існуючі рішення на ринку. Важливо переконатися, що розробка власного рішення є доцільною, і воно не втратить своєї актуальності в майбутньому. Це можна зробити шляхом аналізу тенденцій на ринку цифрових інструментів комунікації, потреб користувачів та специфіки галузі. Дослідження дозволить визначити, чи існує попит на нові функції чи послуги, яких ще немає у вже наявних рішеннях.

Одним із таких успішних рішень, яке є заточеним для галузі інформаційних технологій і широко використовується у сфері розробки програмного забезпечення, є платформа Jira [5]. Ця платформа (рис.1.1) дозволяє команді ефективно співпрацювати, відстежувати прогрес і швидко реагувати на зміни.

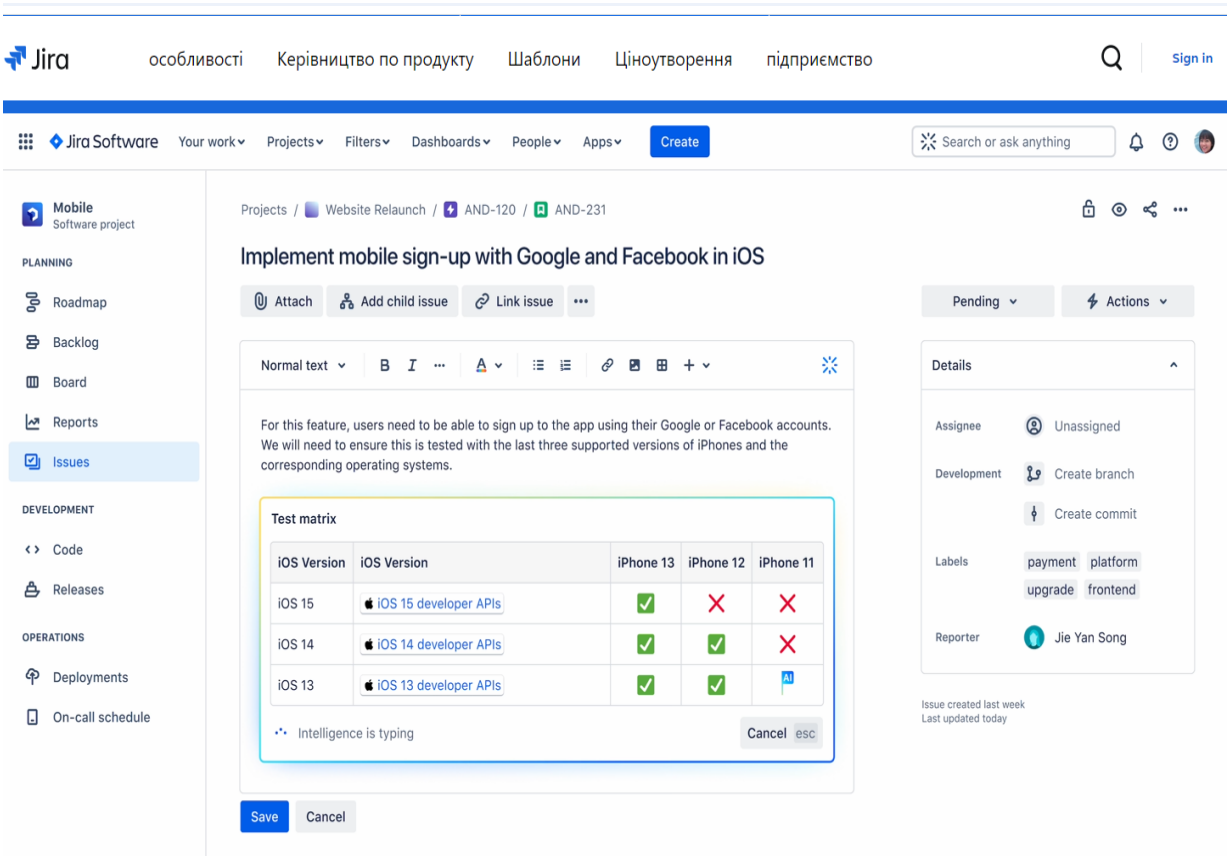

Рис 1.1 — Інтерфейс Jira

Для організації роботи необхідно створити проект, який включає всі завдання, які необхідно виконати, ролі для членів команди та запросити їх приєднатися до спільної роботи. Завдання може призначатися окремим учасникам або групам учасників відповідно до проекту.

Крім вкладки «Проекти», інтерфейс Jira включає й інші ключові розділи, які забезпечують повноцінне управління робочими процесами. Наприклад, вкладка «Завдання» надає доступ до всіх завдань, що знаходяться в межах проекту. Тут можна фільтрувати завдання за всіма критеріями, сортувати їх за пріоритетністю або статусом, а також переглядати детальну інформацію про кожне завдання.

Інструмент «Дошка» надає користувачам можливість налаштувати власний робочий простір, відображаючи ті дані, які є кінцевими для конкретного користувача чи команди. Це можуть бути графіки, діаграми, віджети, відомості про поточні завдання або прогрес у досягненнях цілей.

У вкладці «Релізи» можна відстежити етапи підготовки до випуску нової версії продукту, побачити які завдання вже завершені, а які потребують доопрацювання. Цей розділ дозволяє не лише планувати релізи, але й ефективно контролювати виконання завдань, пов’язаних із тестуванням, документуванням, а також забезпеченням готовності.

Ще одним компонентом є розділ «Звіти», де можна формувати різні типи звітів для аналізу ефективності роботи команди, відстеження часу, витраченого на завдання та оцінки продуктивності.

Ще одним рішенням цифрової корпоративної платформи, яка не є орієнтованою суто на якусь галузь і використовується як для роботи так і для навчання є Teams (див. рис.1.2) [6].

Робота над поставленими задачами тут організована у вигляді створення окремих чат-груп, які називаються командами. Власник команди може додавати до неї учасників, публікувати завдання із вказанням часу виконання, а потім переглядати надані учасниками відповіді у вигляді текстових повідомлень або вкладених матеріалів. У чаті можна публікувати теми для обговорення, також

можна обговорювати і самі завдання у коментарях до відповідного посту. Teams дозволяє проводити відео-конференції із подальшим її записом, який буде доступний у файлах каналу. У Microsoft Teams підтримується повна функціональність месенджера, включаючи приватні та групові чати, можливість надсилання мультимедійних повідомлень, реакції на повідомлення, цитування, а також функцію згадування.

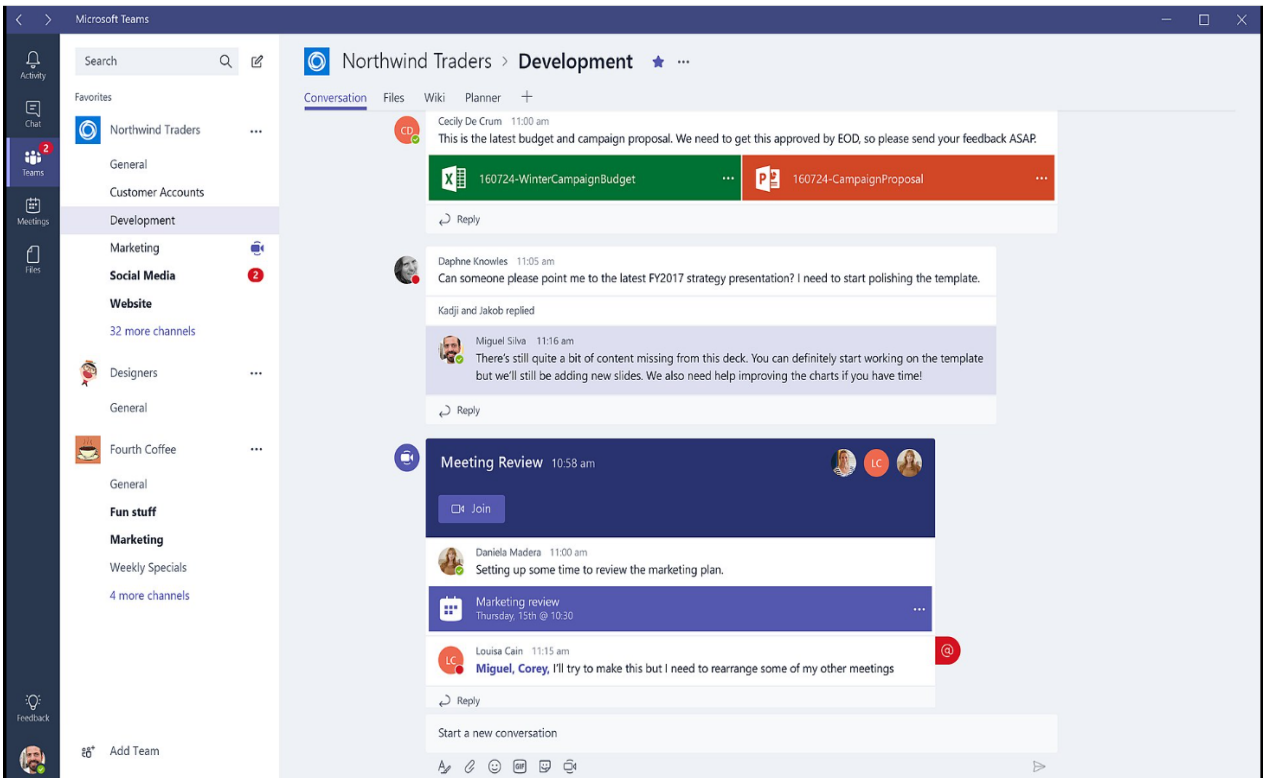

Рис 1.2 — Інтерфейс Teams

Основним фокусом платформи АСКОД показаної на рисунку 1.3 є впровадження електронного документообігу на підприємстві. Ця платформа забезпечує централізоване зберігання документів, контроль за їхнім рухом і виконанням завдань, а також спрощує пошук та обробку інформації [7].

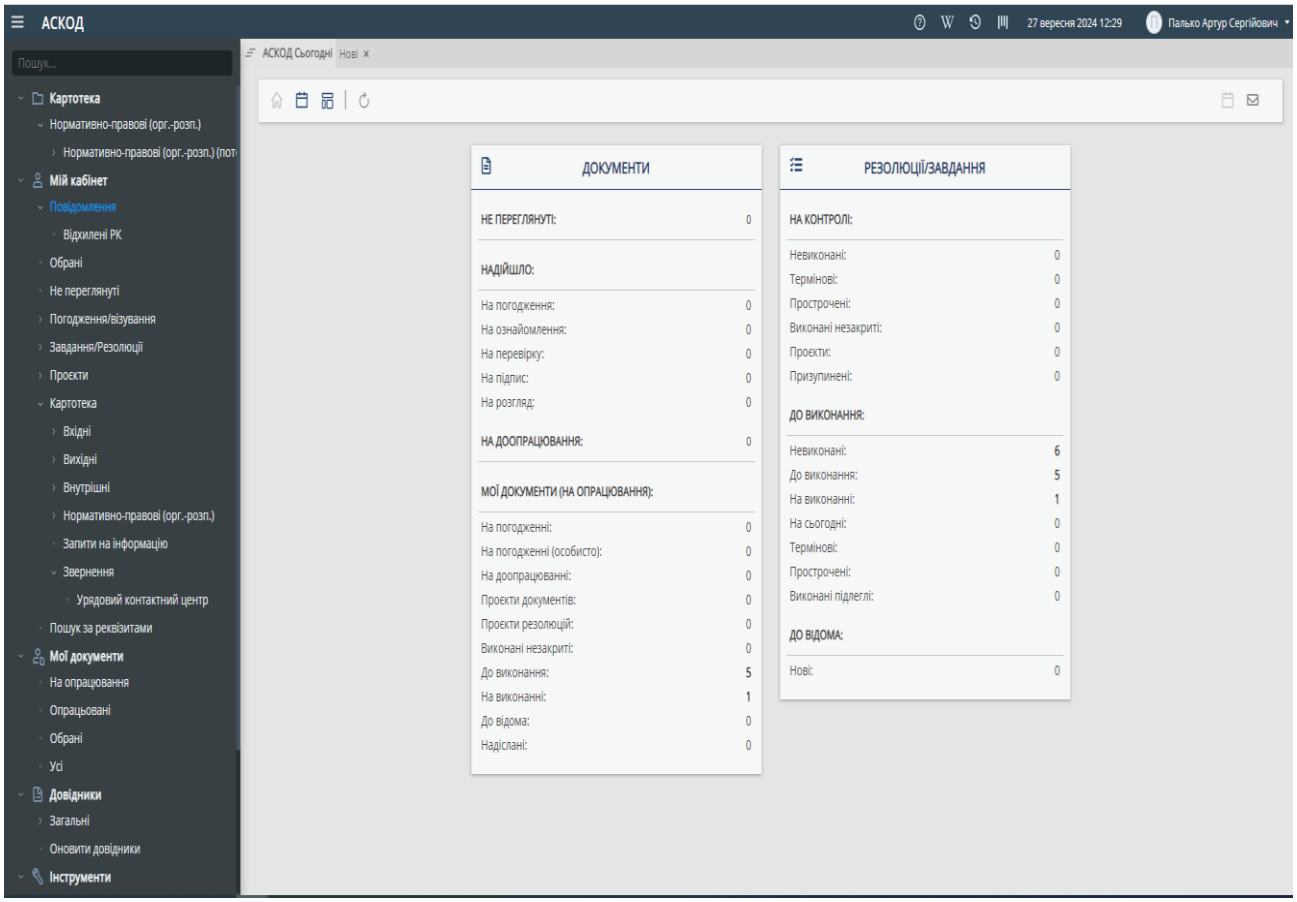

Рис 1.3 – Інтерфейс АСКОД

У самому додатку немає можливості для прямого пересилання повідомлень між користувачами, проте можна:

— обговорювати документи, створювати черги погоджень;

— завдання можна призначати без вкладень документу;

— міститься зручний механізм звітувань по виконаному завданню;

— окрім пріоритизації можна вказати один із стандартних типів завдання (до відома, до виконання, для врахування в роботі) або ж зробити власний заголовок;

— використовується електронно-цифровий підпис документів;

— можна складати резолюції документів.

Інтерфейс додатку складається із бічної панелі, яка містить декілька розділів. У розділі «Мій кабінет» містяться підрозділи що дозволяють перегляну документи які надійшли користувачу за їх категорією. Розділ «Мої документи» дозволяє перегляну опрацьовані документи незалежно від їх категорії,

документи що тільки перебувають на опрацюванні у даного користувача, обрані документи або ж одразу всі. Знайти допоміжну інформацію щодо користування програмою можна у розділі «Довідник». Якщо виникає питання щодо часу відклику додатку, можна скористатися розділом «Інструменти», який містить вбудовані засоби для перевірки швидкості роботи системи.

Робоча область вікна містить дві дошки «Документи» та «Завдання». Обидві дошки містять відповідну інформацію щодо кількості отриманих документів у першому випадку та — завдань у другому. Їх основна перевага порівняно із бічною панеллю те що вони дозволяють оперативно відслідковувати кількість отриманих завдань та ще й за категоріями, що дозволить правильно розставити пріоритети щодо порядку їх виконання.

Підходи до комунікації, втілені у вищезгаданих рішеннях, однозначно стануть корисними при створенні власного додатку. У Microsoft Teams можна запозичити структурування обговорень, що дозволяє вести цілеспрямовані та організовані дискусії. Платформа АСКОД демонструє ефективні методи організації обліку вхідних завдань, що сприяє покращенню управління проектами. А JIRA, у свою чергу, пропонує потужні інструменти для трекінгу завдань та їхнього статусу, що допомагає командам ефективно планувати й контролювати виконання робіт. Комбінування цих елементів може забезпечити створення інтегрованого та продуктивного додатку для управління комунікацією та завданнями в команді.

У підсумку цього розділу порівняльні аналізи, представлені нижче, ілюструють сильні сторони кожного з рішень.

Jira, рік випуску 2002, розроблений ф-ма Atlassian з архітектурою client-server, мова написання— Java.

Функції Jira:

— управління проєктами та завданнями;

— інтеграція з іншими інструментами розробки;

— відстеження помилок та проблем.

Переваги Jira:

— гнучка система налаштувань для різних типів проєктів;

— широкі можливості для звітності та аналітики;

— підтримка Agile-методологій.

Недоліки Jira:

— складний інтерфейс для нових користувачів;

— висока вартість для малих команд;

— обмежені можливості комунікації всередині команди.

Microsoft Teams рік випуску 2017, розроблений корпорацією Microsoft з архітектурою cloud-based, мови написання— C#, TypeScript.

Функції Microsoft Teams:

— групові чати та відеоконференції;

— спільне редагування документів в реальному часі;

— інтеграція з Office 365 та іншими сервісами Microsoft.

Переваги Microsoft Teams:

— зручність в організації командної роботи;

— інтеграція з популярними бізнес-інструментами.

Недоліки Microsoft Teams:

— обмежена функціональність без платної підписки;

— інтерфейс може бути перевантажений для початківців;

— обмеження на роботу в офлайн-режимі.

АСКОД рік випуску 2010, розроблений ТОВ «ІнфоПлюс» з архітектурою client-server, мова написання — C++.

Функції АСКОД:

— електронний документообіг;

— управління діловими процесами;

— зберігання та архівація документів.

Переваги АСКОД:

— високий рівень безпеки для державних установ;

— повноцінна інтеграція з українським законодавством;

— гнучкість налаштувань для різних видів документообігу.

Недоліки АСКОД:

— обмежений набір функцій порівняно з міжнародними аналогами;

— складний інтерфейс для нових користувачів;

— висока вартість для невеликих організацій.

1.3 Забезпечення конфіденційності внутрішньої комунікації

Збереження конфіденційності та забезпечення безпеки корпоративних даних є надзвичайно важливими аспектами для будь-якого підприємства. Це питання не тільки технічної, але й корпоративної відповідальності, що безпосередньо впливає на довгострокову стратегію компанії та її майбутнє.

Корпоративні дані охоплюють важливі аспекти діяльності підприємства, включаючи інформацію про персонал, інтелектуальну власність і фінансові показники, кожен із яких є критично важливим для стабільного функціонування та розвитку компанії. Інформація про працівників містить не тільки заяви на працевлаштування та контракти, але й ключові відомості, що впливають на стратегічні кадрові рішення, які визначають продуктивність та ефективність роботи всього колективу. Інтелектуальна власність, яка охоплює патенти, торгові марки та інноваційні стратегії, є основою для утримання лідерських позицій на ринку. Її витік може призвести до значних фінансових втрат та втрати репутації. Компанії, які не вживають заходів для захисту цієї інформації, наражають себе на серйозні ризики. Фінансові дані, такі як звіти про доходи та баланси, відображають реальний стан компанії та дозволяють оцінити її перспективи на майбутнє. Витік або неправильне управління цими даними може не лише нашкодити репутації компанії, а й створити загрозу для її фінансової стабільності. Тому захист корпоративних даних є важливим не тільки з точки зору безпеки, але й з позиції стратегічного управління підприємством.

Кібербезпека — це постійний процес захисту інформаційних систем і даних від несанкціонованого доступу, модифікації або знищення, що спрямований на забезпечення ключових характеристик даних: їхньої цілісності,

конфіденційності та доступності для авторизованих користувачів [8]. Властивість цілісності передбачає точність, надійність та послідовність даних на всіх етапах їх обробки та зберігання. Вона забезпечує захист від несанкціонованих змін під час передачі або зберігання, гарантує, що дані залишаються незмінними та достовірними до моменту їх використання. Доступність означає можливість для користувачів своєчасно отримувати дані відповідно до їхніх ролей та повноважень у системі. Вона передбачає безперервну роботу системи та захист від збоїв, які можуть перешкоджати доступу до інформації, гарантуючи, що дані завжди будуть доступні для уповноважених осіб. Конфіденційність даних, або їх захищеність, забезпечує доступ до певної інформації лише обмеженому колу користувачів. Це критично важливо для запобігання несанкціонованому доступу та захисту чутливої інформації, такої як персональні дані, фінансові записи чи комерційні таємниці.

До вразливостей системи безпеки відноситься переповнення буфера. Ця вразливість не тільки виявляє недоліки в коді, а й вказує на проблеми в архітектурі програмного забезпечення загалом. Оскільки буфер є ключовим сховищем під час виконання коду, він використовується для зберігання різних проміжних значень, що виникають під час обробки даних. Коли програма не контролює обсяги даних, які записуються в буфер, це може призвести до переповнення, яке, в свою чергу, призведе до краху програми. Ситуація, коли програма не провела належну перевірку вхідних даних, може призвести до знищення або розповсюдження конфіденційної інформації. Однією з найпоширеніших практик є впровадження зловмисниками різного роду SQL запитів [9] у поля введення. Це техніка, відома як SQL-ін'єкція, яка дозволяє зловмиснику маніпулювати запитами до бази даних, обходити механізми безпеки та отримувати доступ до чутливих даних, таких як паролі, номери кредитних карток або особисті дані користувачів. Яскравим прикладом такого випадку є введення на формі входу власного SQL виразу, який модифікує SQL запит, що здійснюється після підтвердження входу, для перевірки облікових даних як вірних при будь-яких параметрах. Це досягається за рахунок вставлення

зловмисником спеціально crafted SQL-коду у поле введення. Для незаконного доступу до конфіденційних даних зловмисник може також скористатися методом грубої сили, використовуючи програми для підбору паролів. Стабільність роботи будь-якого сайту, що перебуває в мережі Інтернет, можна порушити навіть без застосування складних хакерських прийомів, здійснивши DDoS-атаку. Це відбувається, коли нетипово велика кількість комп'ютерів заплановано та одночасно намагається отримати доступ до сайту, що призводить до його перевантаження, внаслідок чого він не може продовжувати функціонування і надавати відповіді клієнтам.

Незалежно від виду зловмисного ПЗ, що атакувало систему, можна виділити основні ознаки зараження пристрою:

— підвищене навантаження на процесор;

— помітне уповільнення роботи комп'ютера;

— часті зависання або збої в роботі;

— зниження швидкості завантаження веб-сторінок;

— виникнення незрозумілих проблем з інтернет-з'єднанням;

— зміни або видалення файлів без відома користувача;

— поява невідомих файлів, програм або ярликів на робочому столі;

— запуск невідомих процесів;

— самовільне завершення роботи програм або зміна їх налаштувань;

— автоматична розсилка електронної пошти без відома користувача.

Шпигунське програмне забезпечення — це тип шкідливого ПЗ, який приховано відстежує дії користувача на пристрої з метою збору інформації. Воно може перехоплювати дані з клавіатури, відстежувати інтернет-активність і навіть передавати особисті відомості без відома користувача. Шпигунське ПЗ часто вносить зміни в налаштування системи для уникнення виявлення та обходу засобів безпеки, що ускладнює його видалення.

Вірус — це шкідлива програма дії якої спрямовані переважно на поширення та пошкодження фалів або системи. Вірус прикріплюється до файлів і активується при їх відкритті, після чого заражає інші частини системи. Зазвичай

має характерний життєвий цикл, що включає етапи інфекції, розмноження та поширення. Для захисту від вірусів важливо використовувати антивірусне програмне забезпечення, регулярно оновлювати його, а також дотримуватися обережності при відкритті файлів з неперевірених джерел [10].

Рекламне програмне забезпечення — це тип програм, які автоматично відображають або завантажують рекламний контент на пристрій користувача, зазвичай без його згоди. Воно часто вбудовується у безкоштовні програми і може мати різний рівень серйозності. Це можуть бути як декілька спонсорських програм, що легко піддаються видаленню, так і безкінечне скачування непотрібних програм, які генерують велику кількість оголошень або банерних вікон, що закривають робочу область монітора. Це може призвести до перевантаження комп'ютера різними процесами. Користувачам завжди неприємно стикатися з програмами такого роду. Крім того, у них можуть виникнути проблеми з видаленням рекламного програмного забезпечення, оскільки вони можуть не знати, яка програма або процес є першоджерелом проблеми. Деякі встановлені програми можуть також не піддаватися видаленню. Часто можна уникнути встановлення рекламного програмного забезпечення, оскільки його блокують антивіруси. Якщо користувач уважно читає пропозиції щодо встановлення додаткових програм під час установки, він може

Програми-вимагачі — це тип шкідливого програмного забезпечення, яке блокує доступ до комп'ютерної системи або шифрує файли на ній, вимагаючи від жертви сплати викупу за відновлення доступу. Найчастіше такі програми заважають користувачеві взаємодіяти з власним комп'ютером, відображаючи лише банерне повідомлення з реквізитами для сплати, які жертва має заповнити, і тільки в такому разі зможе повернути контроль над ПК. Однак немає жодної гарантії, що зловмисники насправді повернуть дані або доступ до системи. Крім того, знаючи вразливості ПК, вони можуть вдатися до повторного шантажу, а гроші, отримані від жертв, слугують стимулом для продовження подібних атак на інших користувачів. Тому в такій ситуації не варто піддаватися залякуванням

щодо розповсюдження персональних даних у мережі чи їх повного знищення. Краще звернутися за допомогою до спеціаліста з кібербезпеки.

Троянський кінь — це тип шкідливого програмного забезпечення, що в основному являє собою невелику виконувану програму, яка не запускається та не поширюється автоматично [11]. Зловмиснику необхідно, щоб користувач самовільно запустив цю програму, яка, скориставшись його повноваженнями у системі, зможе перехопити якусь інформацію чи надати зловмиснику доступ до певної інформації від імені користувача. Основною хитрістю є те, що користувач абсолютно не розуміє, як виконується маніпуляція з його комп'ютером, на яку він, нічого не підозрюючи, дав дозвіл. Однією із схем, якою користуються зловмисники для зараження комп'ютера трояном, є надсилання підроблених листів на електронну пошту, у яких під різними приводами зловмисник вмовляє відкрити файл трояну. Наприклад, він може сказати, що цей файл є зображенням або відеофайлом, і користувач, не підозрюючи різниці, запустить файл, що не має розширення фото чи відео, а є виконуваною програмою. Троянські коні часто не можуть сховатися від вбудованого в систему антивірусу, тому зловмисники мають ще один дуже небезпечний варіант. Іншою небезпечною ситуацією, якою часто користуються зловмисники, є намір користувача встановити неліцензійне програмне забезпечення. Складовою цього процесу є потреба у відключенні антивірусу, на яку користувач змушений піти. Після цього він розв'язує руки троянському коню, якому тепер нічого не загрожує, і система не зможе сповістити про це користувача або ж заблокувати загрозу, як це було раніше.

З огляду на описані в цьому розділі небезпеки для розроблюваного додатку, більшість з яких виникає внаслідок вірусів, що поширюються через Інтернет, було прийнято рішення про доцільність його відокремлення від глобальної мережі шляхом впровадження в локальну мережу. Детальний опис цієї локальної мережі буде надано в наступному підрозділі.

1.4 Дослідження локальних мереж

Основною властивістю локальної мережі порівняно з глобальною є вища ступінь захищеності. Локальна мережа (LAN) функціонує в межах обмеженого географічного простору, тобто вона не має виходу в Інтернет, що дозволяє краще контролювати доступ до ресурсів і застосовувати більш ефективні заходи безпеки [12]. Це робить локальні мережі менш вразливими до зовнішніх атак у порівнянні з глобальними мережами (WAN), які охоплюють великі відстані і піддаються більшому ризику перехоплення або вторгнень через публічні канали зв'язку. Власна мережа підприємства дозволяє зручне використання мережевих принтерів і телефонії, безпечне підключення до інших комп'ютерів через віддалений робочий стіл, а також забезпечує централізоване сховище для даних на сервері, який фізично знаходиться на підприємстві. Така мережа дозволяє ефективно керувати ресурсами, забезпечувати швидкий обмін інформацією між відділами, а також підвищує безпеку даних завдяки контролю доступу та внутрішнім політикам безпеки.

Фактично ЛОМ складається з персональних комп’ютерів підприємства кожний комп’ютер підключається до мережевої розетки у кімнаті, а та у свою чергу до комутаційної панелі яка з’єднана із серверами. Для маскування та організації кабелів у приміщеннях використовуються лотки та короби, які зазвичай розташовуються під стелею, уздовж стін або під підлогою. Це дозволяє акуратно прокладати кабелі, запобігаючи їхньому переплутуванню, а також зменшує ризик механічних пошкоджень і захищає від зовнішніх впливів.

Вибір правильної топології мережі є критично важливим рішенням для будь-якої організації, адже саме від неї залежить ефективність, надійність та безпека передачі даних. Існує кілька типів топології, кожна з яких має свої унікальні переваги та обмеження, які слід враховувати залежно від вимог до мережі.

Однією з найбільш поширених топологій є «зірка», де всі пристрої підключені до центрального вузла, такого як комутатор або маршрутизатор. Ця

топологія забезпечує легкість у розширенні мережі та простоту управління, адже кожен пристрій незалежно підключений до центрального елементу. Якщо один пристрій виходить з ладу, це не впливає на працездатність інших, що робить мережу дуже надійною. Однак, головний недолік полягає в залежності від центрального вузла — його вихід з ладу призведе до відключення всієї мережі.

У противагу «зірці» існує топологія «шина», де всі пристрої підключені до одного загального кабелю. Ця топологія є досить простою і економічною, оскільки використовує мінімум кабелів. Проте обрив головного кабелю призведе до зупинки всієї мережі, що робить її менш надійною у порівнянні з іншими топологіями.

Іншим підходом є топологія «кільце», де кожен пристрій з'єднаний з двома сусідніми пристроями, утворюючи замкнуте коло. Дані передаються від одного вузла до іншого по колу, що дозволяє рівномірно розподілити навантаження на мережу. Однак, будь-який збій в мережі може порушити весь трафік, хоча це можна частково компенсувати механізмами дублювання даних.

Для забезпечення максимальної надійності та стійкості до збоїв часто використовують топологію «сітка» (Mesh), де кожен пристрій з'єднаний з кількома іншими, створюючи множинні шляхи для передачі даних. Якщо один з'єднувальний шлях виходить з ладу, мережа може перенаправити трафік іншими маршрутами, що робить 'сітку' ідеальним вибором для мереж, де критично важлива стабільність. Однак, така топологія є складною в налаштуванні і потребує значних фінансових витрат через велику кількість кабелів та обладнання.

Для великих ієрархічних мереж часто обирають топологію «дерево», яка поєднує кілька топологій «зірка» у багаторівневій структурі. Це дозволяє легко розширювати мережу, додаючи нові пристрої на різних рівнях. Однак вихід з ладу вузла вищого рівня може вплинути на всю підмережу, що потребує ретельного планування.

У великих територіях, де важливо забезпечити мобільність та покриття, застосовується топологія «комірка» (Cellular). Вона дозволяє розділити мережу

на комірки, кожна з яких обслуговується власною базовою станцією, що дозволяє ефективно розподілити навантаження на мережу. Проте така структура може бути складною в управлінні, особливо при великій кількості комірок.

Нарешті, існує «гібридна» топологія, що поєднує в собі кілька різних типів топологій, наприклад, зірка і кільце. Це дозволяє створити мережу, що максимально відповідає потребам організації, комбінуючи переваги різних підходів. Така топологія надає високу гнучкість, але водночас ускладнює проєктування та обслуговування.

Таким чином, кожен тип топології має свої сильні та слабкі сторони, і вибір оптимального варіанту залежить від специфічних вимог до мережі: її масштабу, стабільності, потреб у безпеці та фінансових обмежень. Раціональне поєднання різних топологій може забезпечити найкраще рішення для будь-якої організації.

Комутаційні панелі та сервери знаходяться у спеціальному серверному приміщенню до якого є низка вимог:

— забезпечення належної вентиляції та кондиціонування для підтримки; оптимальної температури обладнання;

— система резервного живлення для безперебійної роботи серверів;

— контроль доступу для захисту від несанкціонованого проникнення;

— пожежна безпека, зокрема системи пожежогасіння;

— захист від електромагнітних перешкод та інших факторів, які можуть вплинути на роботу обладнання.

Відсутність вікон забезпечує підвищену безпеку, зменшує ризик фізичного проникнення та впливу погодних умов, таких як підвищена температура через сонячне світло. Також серверне приміщення рекомендується розташовувати на середніх поверхах будівлі для зниження ризиків затоплення на нижніх поверхах або пошкоджень від падіння конструкцій на верхніх поверхах. Важливо також, щоб приміщення мало металеві двері для захисту від несанкціонованого доступу та забезпечення додаткової безпеки.

Обслуговуванням мережі підприємства займаються системні адміністратори, на яких, зокрема, покладаються функції фахівця з кібербезпеки,

що виконують низку важливих завдань. До їхніх обов'язків входить прокладання кабелів у приміщеннях, монтаж мережевих розеток, а також налаштування серверів баз даних, де зберігається інформація підприємства, зокрема система пропуску через турнікети. Крім того, вони займаються налаштуванням комп'ютерів співробітників і мережевого обладнання, такого як принтери, камери відеоспостереження та телефонія. Системні адміністратори також відповідають за впровадження і підтримку політик безпеки, розподіл прав доступу до різних ресурсів і розділів системи для користувачів, ведення обліку мережевого обладнання, а також складання планів його модернізації і закупівель. Крім того, вони забезпечують безперебійну роботу всієї інфраструктури, швидке вирішення технічних проблем, регулярне оновлення програмного забезпечення і резервне копіювання даних для підтримки належного рівня захисту та ефективності мережі.

Відокремлення мережі підприємства від загроз з боку глобальної мережі, безсумнівно, є важливим кроком у забезпеченні безпеки. Проте воно не виключає внутрішніх загроз, які можуть виникати з боку користувачів, адже вони мають фізичний доступ до комп'ютерів мережі. Для мінімізації цих ризиків на кожному комп'ютері існує обліковий запис адміністратора, а звичайні користувачі не мають прав адміністратора на своїх пристроях і не можуть встановлювати власне програмне забезпечення. Крім того, важливо дотримуватися принципів корпоративної етики в аспекті безпеки. Це передбачає не лише технічні заходи, але й свідоме ставлення до власної безпеки та безпеки колег. Користувачам слід уникати залишати сторонніх осіб у кабінеті, завжди закривати двері, якщо в кімнаті немає інших співробітників. Також, коли працівник відходить від свого робочого місця, необхідно блокувати робочий стіл комп'ютера, щоб уникнути несанкціонованого доступу до конфіденційної інформації. Важливо проводити регулярні тренінги для співробітників, щоб підвищити їх обізнаність щодо загроз безпеці, методів захисту і правильного реагування на можливі інциденти. Забезпечення безпеки вимагає спільних зусиль усіх працівників підприємства.

У розділі було проведено аналіз сфери комунікації працівників підприємства через огляд існуючих на ринку аналогів : Jira, Teams, АСКОД . Визначено які ідеї доречно впровадити у своєму додатку. Розглянуто поняття корпоративної етики та визначено вимоги, які необхідно врахувати при розробці майбутнього додатку. Проведено розгляд причин виникнення конфліктів та методів їх вирішення, з чого зроблено висновок про необхідність впровадження у розроблюваному додатку механізму щодо обговорення поставлених на виконання завдань.

Крім того, у розділі була розкрита важливість захисту інформації, що є ключовим аспектом для будь-якого підприємства в умовах сучасного інформаційного середовища. Розглянуті загрози, що виникають у мережі Інтернет, включаючи можливість кібератак, витоку конфіденційних даних та інших небезпек, які можуть поставити під загрозу безпеку корпоративної інформації. Із проведених міркувань виплив висновок про розгортання розроблюваного додатку у локальній мережі підприємства.

2 ОПИС ІНСТРУМЕНТІВ РОЗРОБКИ

Розділ містить детальний опис інструментів, використаних під час розробки системи, включаючи технології для створення серверної та клієнтської частин, а також засоби для проектування бази даних і управління комунікацією користувачів.

2.1 Ключові особливості розробки веб-платформ

Розробка веб-платформ є однією з найважливіших складових сучасної цифрової трансформації, яка забезпечує ефективність і розвиток як підприємств, так і освітніх установ чи державних організацій. У наш час веб-сайт виступає не тільки як інструмент для представлення бренду або послуг, а й як потужний механізм для автоматизації різних бізнес-процесів, покращення комунікації між співробітниками, клієнтами та партнерами, а також для інтеграції з іншими цифровими платформами [13]. Тому розробка веб-сайту вимагає уваги до кожного етапу процесу і глибокого професіоналізму.

Перший етап створення веб-сайту полягає в аналізі вимог клієнта, що часто фіксуються у вигляді брифу. Це документ, який надає замовник і в якому викладає свої побажання щодо дизайну, функціональності, специфікацій та особливостей проєкту. Бриф дозволяє розробникам краще зрозуміти потреби замовника та правильно спланувати наступні етапи роботи. На основі цього документа створюється технічне завдання, яке описує не лише цілі сайту, а й конкретні етапи розробки, необхідні для досягнення цих цілей, а також терміни виконання робіт.

Наступним етапом є прототипування, де розробники створюють попередній план сайту, що включає структуру його сторінок, елементи навігації, інтерфейс та основні функціональні блоки. Прототип може бути представленим у вигляді ескізів або інтерактивних моделей, що дають чітке уявлення про розміщення елементів на сторінці, а також дозволяють тестувати користувацький досвід до створення фінальної версії сайту. Це дозволяє виявити потенційні проблеми на ранньому етапі та оптимізувати взаємодію з користувачем.

Після прототипування розпочинається робота над дизайном. Дизайн сайту повинен бути не лише естетично привабливим, а й функціональним, зручним для користувачів та адаптованим під різні типи пристроїв. Адаптивний дизайн став стандартом сучасного веб-дизайну, оскільки він забезпечує однакову зручність перегляду сайту як на настільних комп'ютерах, так і на мобільних телефонах чи планшетах. Важливим є також врахування сучасних тенденцій дизайну, таких як мінімалізм, зручність навігації та швидкість завантаження сторінок.

Після створення дизайну переходять до етапу реалізації. Front-end розробники відповідають за інтерактивність сайту, використовуючи HTML, CSS, JavaScript та інші технології для створення ефективного та зручного інтерфейсу. Це включає створення якості відображення сайту на різних екранах, інтерактивні елементи та анімацію, а також забезпечення доступності для користувачів з різними можливостями. Back-end розробники працюють над серверною частиною сайту, налаштовують бази даних, сервери, забезпечують функціональність, що лежить в основі сайту, а також реалізують інтеграцію з іншими системами, такими як платіжні системи або CRM.

Після того, як основні функціональні блоки реалізовані, наступає етап тестування. Тестування — це одна з найбільш критичних стадій процесу розробки, оскільки на ньому виявляються помилки та недоліки, що можуть вплинути на подальшу роботу сайту. Тестування передбачає перевірку всіх аспектів веб-сайту: від коректної роботи функціональних кнопок і форм до тестування дизайну на різних пристроях і браузерах. Крім того, на цьому етапі перевіряється безпека сайту, щоб запобігти можливим зламам та витоку даних користувачів.

Паралельно з тестуванням або після його завершення йде етап наповнення сайту контентом. Наповнення сайту — це не лише додавання текстів, зображень чи відео, а й створення унікального контенту, що відповідає вимогам SEO (пошукової оптимізації). Ключові слова, метатеги, структура контенту — все це має бути ретельно продумано для того, щоб сайт був зручний для пошукових

систем і користувачів. Від якості контенту залежить не тільки привабливість сайту, а й його позиції в пошукових результатах.

Останні етапи включають хостинг і доменне ім’я, які необхідні для публікації сайту в Інтернеті. Вибір хостингу, налаштування серверів та забезпечення швидкого доступу до сайту є ключовими для його ефективної роботи [14]. Після цього сайт стає доступним для широкої аудиторії. І, нарешті, просування сайту в Інтернеті. Це включає в себе SEO-оптимізацію, контекстну рекламу, соціальні мережі, email-маркетинг та інші стратегії, що сприяють залученню відвідувачів і підвищенню популярності сайту.

Таким чином, створення веб-сайту — це складний і багатоступеневий процес, який вимагає поєднання технічних знань, креативності та розуміння потреб користувачів. Для того, щоб сайт був успішним, важливо правильно спланувати кожен етап розробки, від першого аналізу вимог до фінального просування. Веб-сайт, створений за цими принципами, здатен стати потужним інструментом для розвитку бізнесу, підвищення ефективності роботи та комунікації в межах організації або з зовнішніми клієнтами.

Авторизація є ключовим етапом при створенні будь-якої веб-системи, забезпечуючи доступ користувачів до ресурсів у відповідності з їх правами та ролями.

Існує кілька підходів до реалізації авторизації та аутентифікації, серед яких стандартна перевірка паролю та логіну, авторизація через куки, використання JWT-токенів, а також авторизація через сторонні сервіси.

Стандартна перевірка паролю та логіну — підхід, що є одним з найбільш поширених і простих у реалізації. Під час авторизації користувач вводить свій логін та пароль, які перевіряються в базі даних. Однак, цей метод має недоліки, такі як можливість вразливості до атак типу "brute force", а також потребує додаткових заходів для захисту паролів (наприклад, хешування).

Куки — це невеликі файли, що зберігаються в браузері користувача, і використовуються для збереження інформації про сесію. Авторизація через куки дозволяє автоматично ідентифікувати користувача під час наступних запитів до

сервера. Проте використання куків може бути небезпечним через можливість їх викрадення через атаки на сесії, якщо не застосовувати додаткові заходи безпеки, такі як шифрування і налаштування прапорців безпеки (Secure, HttpOnly) [15].

Авторизація через JWT- токени (JSON Web Token) є сучасним методом аутентифікації та авторизації. JWT складається з трьох частин: Header, Payload і Signature. У цьому підході сервер генерує токен, який містить інформацію про користувача, і надсилає його клієнту. Кожного разу, коли клієнт робить запит, він відправляє цей токен на сервер для перевірки. Це дозволяє реалізувати безпечну авторизацію без необхідності зберігати стан сесії на сервері, що робить систему масштабованою. Одним із основних переваг є те, що JWT-токени можуть бути використані для авторизації через різні сервіси та мобільні додатки [16].

OAuth 2.0 є популярним протоколом для авторизації через сторонні сервіси, такі як соціальні мережі (Facebook, Google, і т.д.). Він дає змогу користувачам входити в систему без необхідності створення нових облікових записів. Система працює через надання дозволу користувача для доступу до його даних, після чого створюється доступний токен для подальшої аутентифікації. Це дозволяє скоротити час на реєстрацію та підвищує зручність користування.

Для веб-додатку, який працюватиме в локальній мережі підприємства, оптимальним вибором є авторизація через JWT-токени. Цей метод дозволяє здійснювати швидку, безпечну та масштабовану аутентифікацію без необхідності зберігати сесійні дані на сервері. JWT також забезпечує підтримку різних типів додатків і інтеграцій, що важливо для корпоративних платформ, таких як Teams. Крім того, для додаткової безпеки можна поєднати цей метод з двофакторною аутентифікацією або використанням SSL/TLS для захисту передачі даних.

2.2 Обґрунтований вибір середовища, мови програмування та опис використаних програмних пакетів

Для розробки веб-додатку системи було обрано Microsoft Visual Studio 2022 [17] та мову програмування C# [18] для серверної частини, а для клієнтської

частини — JavaScript [19] у поєднанні з Visual Studio Code [20]. Це рішення дозволяє оптимально використовувати потужні інструменти для кожної частини додатка.

Visual Studio 2022 є популярним середовищем для розробки, яке надає розширені можливості для командної роботи завдяки інтеграції з Team Foundation Server та GitHub. Це дозволяє команді ефективно керувати версіями коду та організовувати робочі процеси. Крім того, Visual Studio забезпечує розширені можливості для тестування: можна писати юніт-тести, проводити автоматизоване тестування, використовувати зручні інструменти налагодження.

Це дозволяє зосередитись на якості програмного забезпечення на кожному етапі розробки. Особливу цінність має інтеграція Visual Studio з хмарними сервісами Azure, яка дозволяє легко публікувати додатки в хмару, масштабувати їх, використовувати сервіси баз даних та хостинг веб-додатків. Це значно полегшує розгортання проєкту та його підтримку. Ще однією перевагою Visual Studio є можливість автоматично генерувати діаграми UML на основі коду, що спрощує моделювання архітектури додатка та допомагає відслідковувати зв'язки між його компонентами.

Попри значні переваги, Visual Studio має і деякі недоліки. Основним з них є обмежена кросплатформність. Хоча існує версія Visual Studio для macOS, вона має дещо урізаний функціонал порівняно з версією для Windows, що може бути проблемою для команд, які використовують різні операційні системи. Також корпоративне використання Visual Studio вимагає придбання ліцензії. Проте Microsoft пропонує безкоштовні версії для студентів, а також для невеликих команд розробників через програму Visual Studio Community.

Принцип серверної розробки в ASP.NET Core [21] полягає в тому, що кожен запит до сервера спочатку проходить через middleware — це проміжні функції, які можна налаштувати за допомогою об'єкта app у методах Run, Use, або Map. Middleware обробляють запит послідовно, виконуючи певні операції, такі як аутентифікація, логування чи обробка помилок. Зрештою, запит доходить до кінцевих точок, які визначені за допомогою методів Map або MapControllers, що

обробляють запити з конкретним маршрутом. Особливим випадком кінцевих точок є контролери — це класи, які дозволяють згрупувати декілька кінцевих точок за загальним маршрутом або за типом HTTP-запиту.

Більшість веб-сайтів, які надають статичну або довідкову інформацію, зазвичай використовують підхід, при якому на кожен запит клієнта сервер відправляє нову HTML-сторінку. Цей підхід є оптимальним для простих сайтів, де інформація змінюється не часто, а також коли немає необхідності в інтерактивних елементах або динамічному оновленні контенту. Сайти такого типу часто мають низькі вимоги до швидкості завантаження та взаємодії з користувачем, тому традиційний підхід з перезавантаженням сторінок залишається ефективним і зрозумілим. Однак, коли веб-сайт має складний інтерфейс з багатьма формами, користувацькими взаємодіями або вимогами до динамічного оновлення контенту, більш доцільно використовувати принцип Single Page Application. Такий підхід дозволяє значно покращити взаємодію з користувачем, оскільки всі зміни в інтерфейсі виконуються без перезавантаження сторінки, що забезпечує швидкість і зручність.

Для реалізації клієнтської частини цього додатку було обрано JavaScript та фреймворк React [22], що дозволяє створювати динамічні компоненти та ефективно управляти станом додатку. Використання Visual Studio Code як основного середовища розробки забезпечує зручну інтеграцію з інструментами для написання та тестування коду, а також оптимізує процес розробки завдяки широкому вибору плагінів і налаштувань.

Розробник отримує повний контроль над налаштуваннями, які спрощують інтеграцію з Git, автоматичне форматування коду та використання сучасних JavaScript-інструментів, таких як ESLint і Webpack. Крім того, для розробки клієнтської частини в Visual Studio Code необхідно встановити Node.js, оскільки він надає серверне середовище для запуску JavaScript-коду та необхідний для використання інструментів збірки, таких як npm або yarn. Це дозволяє налаштувати всі потрібні залежності, управляти пакетами та запускати локальний сервер для розробки React-додатка.

На відміну від використання шаблону ASP.NET Core + React у Visual Studio, Visual Studio Code дозволяє чітко розділити серверну та клієнтську частини додатка [23]. Це спрощує тестування, розгортання та оптимізацію кожної частини окремо. Крім того, Visual Studio Code значно легший і швидший, що дозволяє працювати комфортно навіть з великими проєктами, і має відмінну інтеграцію з інструментами для розробки JavaScript та React.

Фреймворк React використовує принцип односторінкового додатку, де сторінка завантажується лише один раз при першому зверненні до сервера. Після цього React самостійно керує інтерфейсом сторінки, здійснюючи перерендеринг лише тих компонентів, які потребують оновлення, використовуючи власну пам'ять та логіку рендерингу. Компоненти в React — це функції або класи, які повертають JSX — комбінацію JavaScript із HTML та CSS, що дозволяє створювати динамічний інтерфейс користувача.

Однією з основних переваг React є система віртуального DOM, яка дозволяє ефективно оновлювати користувацький інтерфейс. Коли змінюється стан компонента, React створює віртуальне представлення DOM, яке потім порівнюється з реальним DOM, і тільки відмінності застосовуються до реального DOM, що значно покращує продуктивність порівняно з традиційними підходами.

Таким чином, вибір Visual Studio 2022 для серверної частини та JavaScript з Visual Studio Code для клієнтської дозволяє ефективно поєднувати надійність, масштабованість та гнучкість, що відповідає сучасним стандартам розробки веб-додатків.

Під час розробки серверної частини було обрано шаблон ASP.NET Core у Visual Studio, який автоматично створив основну структуру проекту. Для створення клієнтської частини було використано команду create-react-app у Visual Studio Code, що сформувало структуру для роботи з React та JavaScript, включаючи всі необхідні залежності.

Клієнтська частина використовує наступні пакети:

— react — це основний пакет для розробки на бібліотеці React, який надає основні функції для створення компонентів, управління станом і рендерингу інтерфейсів;

— react-rom — пакет, що забезпечує функції для рендерингу React-компонентів на веб-сторінці через методи DOM (Document Object Model). Це дозволяє інтегрувати React у веб-додатки, зв'язуючи його з браузерним DOM;

— react-router-dom — містить усі необхідні компоненти та утиліти для побудови маршрутизації у веб-додатках React, включаючи BrowserRouter, компоненти для управління маршрутизацією, такі як Route, Link, Switch, а також хуки для доступу до параметрів маршруту.

Серверна частина використовує пакети, опис яких надано нижче.

Microsoft.AspNetCore — це основний пакет для розробки веб-додатків на платформі ASP.NET Core. Він містить усі необхідні компоненти для створення та запуску сучасних веб-застосунків, зокрема веб-API, MVC-додатки, сервіси для обробки запитів, маршрутизацію, автентифікацію, безпеку та інші важливі аспекти. Цей пакет є обов'язковим для будь-якого додатку, створеного на ASP.NET Core, оскільки він надає основний функціонал для роботи з веб-запитами та управління життєвим циклом додатку;

EntityFramework — пакет, який забезпечує функціонал для взаємодії з базами даних через Entity Framework, дозволяючи зручно працювати з реляційними базами даних, створювати моделі даних та виконувати запити до них [24]. Цей пакет полегшує роботу з даними, абстрагуючи деталі взаємодії з базою даних і надаючи інструменти для реалізації об'єктно-реляційного відображення (ORM). Для додаткових можливостей, зокрема роботи з міграціями, необхідно встановити додаткові пакети, опис яких наведено нижче;

Microsoft.EntityFrameworkCore.Tools — забезпечує інструменти для створення, управління та застосування міграцій бази даних. Цей пакет включає командний інтерфейс (CLI) та інтеграцію з Visual Studio, що дозволяє розробникам створювати і застосовувати міграції безпосередньо з IDE. Завдяки

цим інструментам можна легко підтримувати актуальність структури бази даних, синхронізуючи її з оновленнями в моделі даних проєкту;

Microsoft.EntityFrameworkCore.Design — містить інструменти для розробки інфраструктури Entity Framework, особливо для роботи з міграціями. Цей пакет забезпечує можливість автоматично генерувати код для міграцій на основі внесених змін у модель даних, спрощуючи процес оновлення структури бази даних. Завдяки цьому інструменту можна легко керувати змінами в схемі даних під час розвитку проєкту;

Microsoft.EntityFrameworkCore.SqlServer — цей пакет надає драйвер для підключення до MS SQL Server і забезпечує можливість використовувати функціональність цієї СУБД. Для інших типів баз даних, таких як MySQL, PostgreSQL чи Oracle, необхідно встановити відповідні драйвери, щоб забезпечити сумісність і доступ до всіх функцій через Entity Framework Core.

2.3 Технології баз даних: сучасний аналіз і тренди

Із розвитком технологій не лише змінилися способи зберігання даних, але й трансформувалися підходи до їх аналізу та використання. Сьогодні інформація виступає стратегічним ресурсом, який може визначати конкурентоспроможність компаній, успішність проектів або навіть розвиток цілих галузей. Великі обсяги даних (Big Data), які раніше було складно уявити, тепер обробляються за лічені секунди завдяки хмарним обчисленням, потужним базам даних і алгоритмам машинного навчання.

Такий стрімкий прогрес ставить перед суспільством нові завдання, серед яких — забезпечення інформаційної безпеки, захист конфіденційних даних і мінімізація ризиків втрати важливої інформації. Крім того, постійне накопичення даних потребує ефективних систем їхньої класифікації та зберігання, що сприяє розвитку автоматизованих та автономних рішень, які можуть самостійно оптимізувати роботу з інформацією. Розвиток сучасних технологій та зростання обсягів даних значно змінюють підходи до зберігання й обробки інформації. Технологічний прогрес, зокрема, розвиток Інтернету речей (IoT), штучного

інтелекту, хмарних обчислень, призводить до того, що обсяг інформації, яка генерується щодня, росте експоненційно. За оцінками експертів, з кожним роком кількість даних в інтернеті подвоюється, що ставить серйозні виклики перед існуючими системами зберігання та обробки інформації.

Однією з найбільших проблем сучасних інформаційних систем є здатність швидко отримувати необхідні дані серед величезних обсягів інформації. Існуючі бази даних і методи обробки даних повинні бути здатними ефективно працювати з великими масивами інформації, забезпечувати оперативний доступ до необхідних відомостей та виконувати складні запити в реальному часі. Забезпечення швидкого доступу до інформації є критичним для багатьох сфер: від фінансових установ до медичних закладів і навіть соціальних мереж. В умовах великої конкуренції та стрімко змінюваного ринку інформація повинна бути доступною не тільки в момент її потреби, а й із можливістю аналізу, прогнозування та прийняття рішень у найкоротші терміни. Це вимагає від сучасних інформаційних систем високої швидкодії, масштабованості та інтеграції з іншими платформами.

Щоб відповідати на ці виклики, виникла потреба в нових архітектурах і технологіях для зберігання і обробки даних. Зокрема, з'явилися нові моделі баз даних, такі як графові бази даних, документоорієнтовані бази даних та нереляційні бази даних, що дозволяють зберігати дані в гнучкій формі, адаптуючись до специфіки даних і запитів користувачів.

Також, важливу роль у вирішенні цих проблем відіграють хмарні технології, які забезпечують не тільки масштабованість, а й економічну ефективність обробки великих обсягів даних. Хмарні платформи дають змогу зберігати дані на віддалених серверах і отримувати до них доступ з будь-якої точки світу, що особливо важливо для глобальних організацій, які працюють із великими масивами даних. Такі сервіси, як Amazon Web Services, Microsoft Azure та Google Cloud, мають у своєму розпорядженні потужні інструменти для обробки даних у реальному часі, що дозволяє значно підвищити швидкість аналізу й доступу до інформації.

Бази даних (БД) продовжують відігравати центральну роль у вирішенні задач зберігання, структурування та доступу до інформації. Сучасні системи керування базами даних (СКБД) забезпечують широкий спектр можливостей, включаючи підтримку масштабованості, швидкодії та інтеграції даних із різних джерел.

Історично бази даних еволюціонували від простих структур, таких як плоскі файли, до складніших реляційних і навіть нереляційних моделей.

СКБД надає користувачам можливість ефективно взаємодіяти з даними, зберігати їх, виконувати різноманітні операції (вставка, оновлення, видалення) та забезпечувати цілісність і безпеку інформації. Важливою особливістю СКБД є те, що вона дозволяє організувати дані у структурованому вигляді, що значно спрощує їх обробку і доступ.

Основними компонентами СКБД є:

— дані — це інформація, яка зберігається в базі даних, наприклад, записи про клієнтів, продукти, транзакції;

— користувачі — особи або програми, які використовують систему для введення, оновлення або отримання даних;

— запити — мови програмування та інтерфейси, за допомогою яких користувачі взаємодіють з даними в базі даних (наприклад, SQL);

— менеджер бази даних — компонент СКБД, який керує доступом до даних, їх збереженням, забезпеченням безпеки та цілісності.

СКБД забезпечує такі ключові функції:

— структурування даних — створення таблиць, зв'язків між ними та визначення типів даних;

— обробка запитів — можливість виконувати складні запити для вибору, оновлення та агрегації даних;

— забезпечення цілісності — гарантія того, що дані будуть коректними і узгодженими, навіть при паралельному доступі кількох користувачів;

— безпека та захист — контроль доступу до даних, використання паролів та шифрування для запобігання несанкціонованому доступу.

Типи баз даних можна розглядати з різних точок зору в залежності від їх структури, способу зберігання та організації даних. Опис двох основних типів баз даних приведено нижче.

Бази даних із плоскими файлами — це найпростіший тип баз даних, який використовує файли для зберігання даних у вигляді однорідних записів без будь-якої внутрішньої структури, що зв'язує їх між собою. Кожен запис є самодостатнім і містить інформацію в необробленому вигляді, без використання індексів або складних структур. У таких базах даних вся інформація зберігається у вигляді одного великого файлу, де записи йдуть один за одним, часто дублюючись і не маючи чіткої ієрархії. Для доступу до даних в таких системах зазвичай використовуються методи послідовного сканування файлів від початку до кінця, що може бути неефективно для великих обсягів інформації. Хоча це дуже проста модель, вона підходить для невеликих баз даних, де немає потреби у складних зв'язках між даними або частих запитах.

Реляційні бази даних — є одними з найпоширеніших і потужніших систем для зберігання інформації. Вони організовують дані у вигляді таблиць, що складаються з рядків (записів) і стовпців (полів). Різні таблиці можуть бути зв'язані між собою за допомогою ключів — спеціальних полів, які містять унікальні ідентифікатори. Це дозволяє створювати складні структури, в яких дані не дублюються і можуть бути ефективно оновлені або видалені. Основною перевагою реляційних баз є використання індексів, які дозволяють значно пришвидшити процес пошуку і доступу до даних, навіть якщо вони розподілені по багатьох таблицях. Завдяки використанню структурованих запитів (наприклад, SQL) користувач може отримувати необхідну інформацію із різних таблиць за допомогою об'єднання їх через спільні поля, що дає змогу гнучко маніпулювати даними. Реляційні бази даних є більш масштабованими та простими в підтримці порівняно з базами даних із плоскими файлами, особливо коли мова йде про великі обсяги даних або складні зв'язки між ними.

У порівнянні з базами даних із плоскими файлами, реляційні бази мають значні переваги у швидкодії, надійності та зручності в управлінні даними. Вони

здатні ефективно обробляти великий обсяг інформації, а їх структура дозволяє гнучко адаптуватися до змін, що виникають в процесі розширення або модифікації даних.

MongoDB, MS SQL, Redis, Oracle і PostgreSQL — це сучасні системи управління базами даних (СКБД), кожна з яких має свої особливості, архітектуру та призначення. Короткий опис кожної із них приведено нижче.

MongoDB [25] є NoSQL документно орієнтованою базою даних.

Призначення MongoDB:

— використовується для зберігання даних у вигляді JSON-подібних документів;

— підходить для масштабованих додатків, що працюють з великими обсягами даних, де структура даних не є фіксованою.

Особливості MongoDB:

— гнучка схема, підтримка горизонтального масштабування через шардинг;

— висока продуктивність для операцій з великими даними.

MS SQL Server (Microsoft SQL Server) [26] є реляційною базою даних (RDBMS).

Призначення MS SQL Server:

— використовується для обробки структурованих даних в таблицях;

— підтримує транзакції та SQL запити;

— зазвичай використовується в екосистемі Microsoft.

Особливості MS SQL Server:

— підтримка ACID-транзакцій

— зручна інтеграція з іншими продуктами Microsoft

— вбудовані інструменти для безпеки, моніторингу та адміністрування.

Redis [27] є NoSQL in-memory базою даних та системою кешування.

Призначення Redis:

— використовується для зберігання даних у пам'яті (ключ-значення) з високою швидкістю доступу;

— підходить для реалізації кешування, черг, лічильників, публікацій/підписок тощо;

Особливості Redis:

— дуже швидка обробка даних завдяки зберіганню їх в оперативній пам'яті; підтримка складних структур даних (рядки, списки, множини);

— можливість персистентного зберігання.

Oracle Database [28] є реляційною базою даних (RDBMS).

Призначення Oracle Database:

— використовується для обробки великих обсягів структурованих даних, з високою надійністю та масштабованістю;

— підходить для підприємств з високими вимогами до безпеки та стабільності.

Особливості MS SQL Server:

— підтримка складних запитів, ACID-транзакцій;

— потужні механізми безпеки, реплікація;

— високі вимоги до адміністрування.

PostgreSQL [29] є реляційною базою даних (RDBMS) та об'єктно-реляційною базою даних.

Призначення PostgreSQL:

— використовується для зберігання структурованих даних з можливістю роботи з об'єктними типами даних;

— підтримує складні запити, транзакції та велику кількість розширень;

Особливості PostgreSQL:

— відкрите програмне забезпечення (open source);

— підтримка розширених SQL-запитів;

— зручна масштабованість та висока продуктивність для великих даних.

Отже, на основі проведених порівнянь, MS SQL Server виявляється найоптимальнійшим вибором для створення інтегрованої комп’ютерної системи для оперативної взаємодії працівників виробничого підприємства із виготовлення меблів. Завдяки своїй реляційній структурі, високій надійності,

інтеграції з іншими продуктами Microsoft і широким аналітичним можливостям, ця база даних забезпечує ефективне управління великими обсягами даних. MS SQL дозволяє забезпечити безпеку інформації, зручність адміністрування, гнучкість у масштабуванні та відповідність вимогам корпоративного середовища, що робить її ідеальним інструментом для підтримки бізнес-процесів і комунікації між співробітниками.

У цьому розділі проведено огляд основних характеристик веб-платформ, які включали вибір технологій для розробки, зокрема фреймворків React та ASP.NET Core, а також опис необхідних програмних пакетів.

Розглянувши реляційні бази даних (PostgreSQL, MS SQL Server, Oracle) та нереляційні (MongoDB, Redis), було зроблено висновок, що MS SQL Server є найоптимальнішим варіантом для впровадження.

3 АРХІТЕКТУРА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

У цьому розділі було спроектовано базу даних застосунку, що включає в себе визначення основних таблиць, їхніх зв'язків та структури для зберігання даних. Крім того, була представлена постановка задачі щодо функціоналу додатку, що охоплює основні вимоги до системи, включаючи реалізацію ключових операцій, таких як створення та редагування завдань.

3.1 Проектування бази даних

Для проектування бази даних комп'ютерної системи для взаємодії працівників підприємства з виготовлення меблів, що включає створення та надсилання завдань, а також обмін повідомленнями, було використано реляційну модель. Це рішення забезпечило структурування даних відповідно до потреб підприємства. Результатом стало створення бази даних, схема якої наведена на рисунку 3.1.

У таблиці 3.1 описану структуру сутності TaskExecutors, яка містить інформацію про виконавців завдань.

Таблиця 3.1 — TaskExecutors

Назва	Тип	Опис
Id	Int	Унікальний ідентифікатор запису, який однозначно визначає кожен рядок, автоінкрементований
ExecutorId	Int	Номер виконавця завдання із таблиці Users
TaskId	Int	Номер завдання із таблиці Tasks

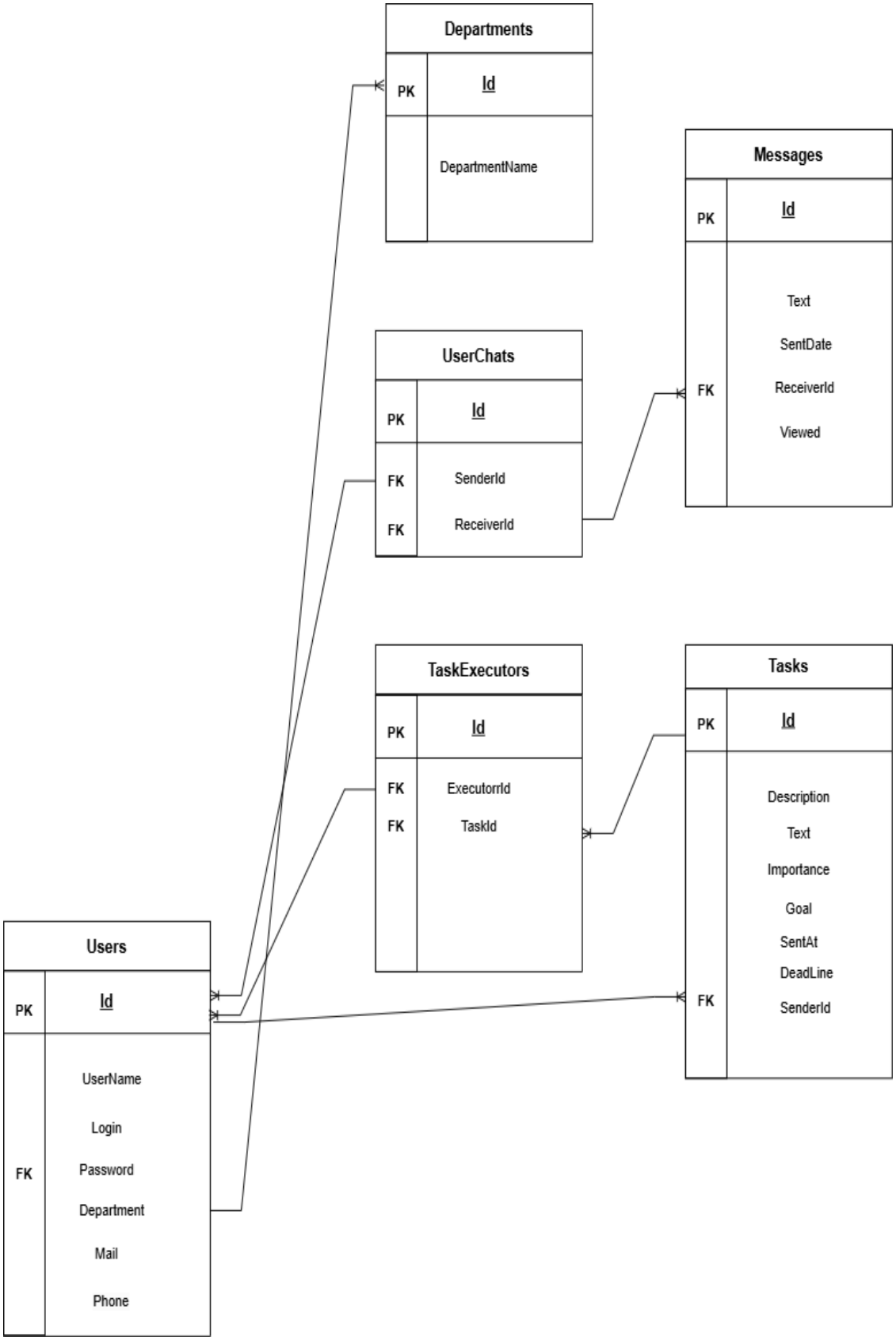

Рисунок 3.1 — Схема бази даних

У таблиці 3.2 наведено структуру сутності Departments, яка представляє відділи у яких працюють користувачі.

Таблиця 3.2 – Departments

Назва	Тип	Опис
Id	Int	Номер департаменту що однозначно визначає його, автоіндексований
DepartmentName	Nvarchar(max)	Назва департаменту працівника

У таблиці 3.3 описану структуру сутності Tasks, яка містить інформацію про кожне завдання.

Таблиця 3.3 — Tasks

Назва	Тип	Опис
Id	Int	Номер завдання, що однозначно визначає його, автоіндексований
Description	Nvarchar(50)	Короткий опис завдання
Text	Nvarchar(max)	Основний текст завдання
Importance	Int	Ступінь важливості завдання

Продовження таблиці 3.3

Goal	Int	Тип завдання (До відома, Для врахування в роботі, До виконання)
SentAt	datetime	Час створення завдання
DeadLine	datetime	Термін виконання завдання
SenderId	Int	Власник завдання із таблиці Users

У таблиці 3.4 описано структуру сутності UserChats, яка є проміжною таблицею для встановлення зв’язку багато до багатьох між користувачами та повідомленнями.

Таблиця 3.4 — UserChats

Назва	Тип	Опис
Id	Int	Унікальний ідентифікатор бесіди між користувачами
SenderId	Int	Ідентифікатор користувача що надіслав повідомлення
ReceiverId	Int	Ідентифікатор користувача кому призначено повідомлення

У таблиці 3.5 наведено детальну структуру сутності Users, яка містить інформацію про користувачів системи та їхні дані.

Таблиця 3.5 — Users

Назва	Типа	Опис
Id	Int	Номер користувача, що однозначно визначає його, автоіндексований
UserName	Nvarchar(max)	Повне ім’я користувача
Login	Nvarchar(30)	Логін користувача для входу
Password	Nvarchar(10)	Пароль користувача для входу
Department	Nvarchar(30)	Назва відділу користувача
Mail	Nvarchar(30)	Пошта корисувача
Phone	Nvarchar(10)	Мобільний телефон користувача

У даній схемі бази даних є наступні зв'язки між таблицями:

1) таблиця "Departments" (Департаменти) має зв'язок "один до багатьох" (one-to-many) з таблицею "Users" через стовпець "DepartmentName", один департамент може мати кількох користувачів, які працюють в ньому;

2) таблиця "Users" (Користувачі);

— зв'язок "багато до одного" (many-to-one) з таблицею "Departments" через стовпець "Department", кожен користувач належить до одного департаменту;

— зв'язок "один до багатьох" (one-to-many) з таблицею "Tasks" через стовпець "SenderId", один користувач може створювати кілька завдань;

— зв'язок "один до багатьох" (one-to-many) з таблицею "TaskExecutors" через стовпець "ExecutorId", один користувач може бути виконавцем декількох завдань;

— зв'язок "один до багатьох" (one-to-many) з таблицею "UserChats" через стовпець "SenderId", один користувач може брати участь у кількох чатах;

3) таблиця "Tasks" (Завдання) має зв'язок "багато до одного" (many-to-one) з таблицею "Users" через стовпець "SenderId". Кожне завдання має одного автора (відправника), який є користувачем;

4) таблиця "TaskExecutors" (Виконавці завдань);

— зв'язок "багато до одного" (many-to-one) з таблицею "Users" через стовпці "SenderId" та "ReceiverId", тотбо один виконавець або відправник завдання завжди є користувачем із таблиці "Users";

— зв'язок "багато до одного" (many-to-one) з таблицею "Tasks" через стовпець "Id" (ідентифікатор завдання), тобто одне завдання може мати декількох виконавців;

5) таблиця "UserChats" (Чати між користувачами);

— зв'язок "багато до одного" (many-to-one) з таблицею "Users" через стовпець "ExecutorId";

— зв'язок "багато до одного" (one-to-many) з таблицею "Messages" через стовпець "ReceiverId", тобто кожен чат пов'язаний з багатьма повідомленнями.

3.2 Постановка задачі

Необхідно створити інтегровану комп'ютерну систему, що забезпечить ефективну взаємодію працівників виробничого підприємства з виготовлення меблів. Ця система буде реалізована як веб-додаток для роботи в локальній мережі підприємства. Вона повинна дозволяти користувачам виконувати персоналізований вхід, переглядати стислий звіт про кількість отриманих завдань, а також сортувати ці завдання за типом, рівнем важливості та відділом, який надіслав завдання. Крім того, користувачі матимуть змогу бачити зворотній зв'язок від виконавців.

Система повинна забезпечувати функціонал для створення завдань та заповнення картки завдання, яка міститиме детальну інформацію про цілі, автора, відповідальних осіб та дедлайни. Важливим елементом буде вбудований механізм комунікації між працівниками, що дозволить обмінюватися текстовими повідомленнями для швидкого вирішення робочих питань.

Система складається з таких основних компонентів:

— сторінка логіну, де користувач здійснює персоналізований вхід у комп’ютерну систему;

— головне меню звідки доступний перехід до всіх інших розіділів сайту;

— модальне вікно для створення власного завдання з головної сторінки;

— сторінка завдань, де відображаються всі завдання користувача;

— вікно карти завдання для надання зворотного зв’язку по завданню зі сторінки завдань;

— сторінка діалогів звідки доступний перехід до чату з конкретним користувачем.

Усі наведенні вище міркування графічно відображені у вигляді діаграми станів проєкту та схеми маршрутизації сайту, що представлені на наступних сторінках.

На рисунку 2.1 представлено діаграму станів проєекту [30], яка показує послідовність кроків для реалізації трьох ключових функцій додатку: перегляд завдань, створення нових завдань та ведення діалогів.

На рисунку 2.2 показана схема маршрутизації сайту, яка демонструє всі доступні шляхи в додатку: login, dialogs, tasks, do-vidoma, do-vykonanny, dlya-vrakhuvannya. Також на схемі відображена логіка рендерингу компонентів інтерфейсу.

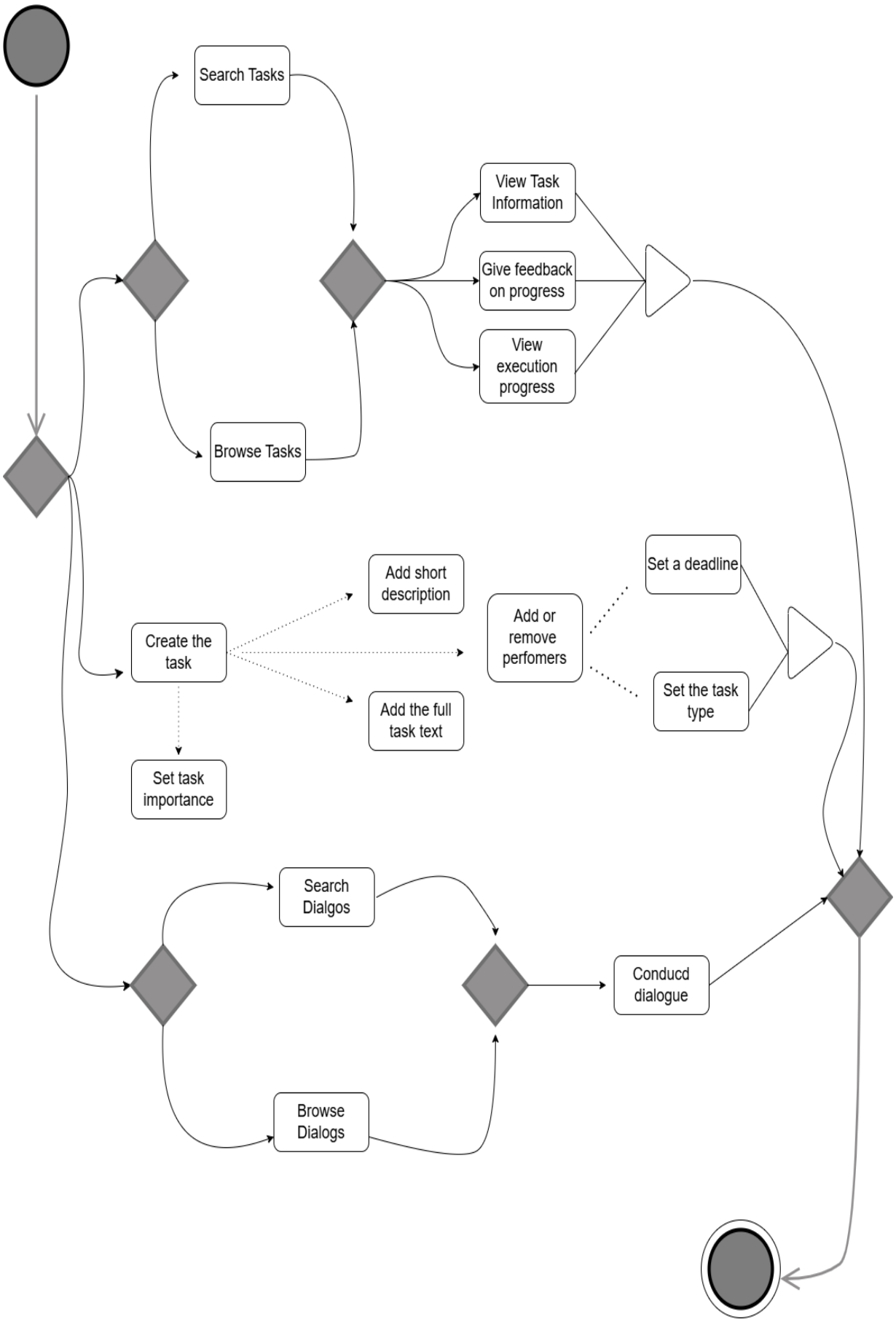

Рисунок 3.2 — Діаграма станів проекту

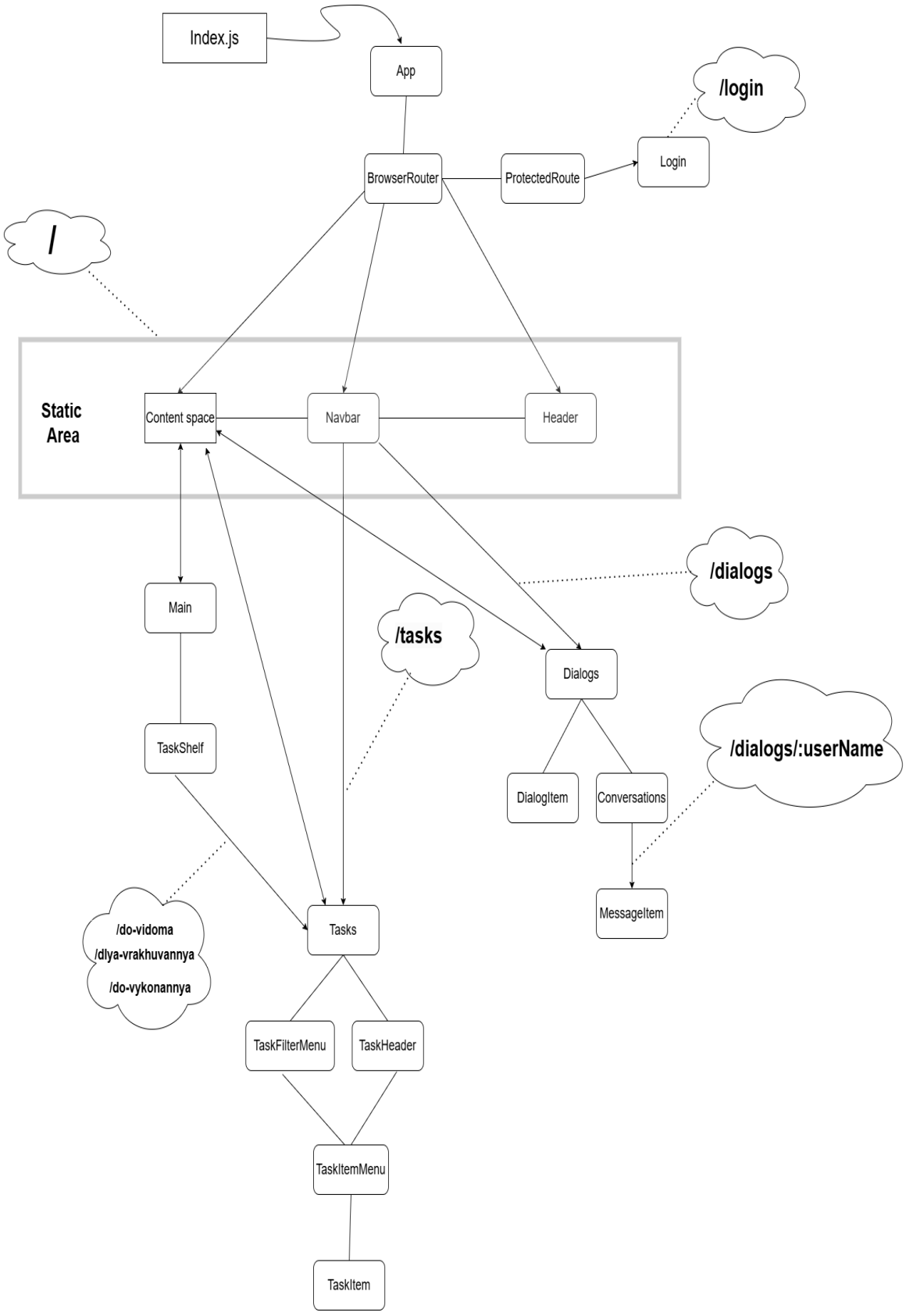

Рисунок 3.3 — Маршрутизація сайту та компоненти інтерфейс

4 ТЕСТУВАННЯ ЗАСТОСУНКУ

4.1 Тестування користувацького інтерфейсу та взаємодії із серверною частиною

Під час спроби доступу до сайту користувач автоматично перенаправляється на сторінку логіну, де йому необхідно пройти процес авторизації. Після успішного введення облікових даних, сервер генерує та надсилає користувачеві cookie, що містить JWT-токен (JSON Web Token). Цей токен використовується для аутентифікації користувача під час переходу на інші розділи сайту, забезпечуючи захищений доступ до ресурсів, що вимагають авторизації. Завдяки використанню JWT, сервер може легко перевіряти права доступу користувача, що значно підвищує безпеку і зменшує навантаження на систему, оскільки повторна авторизація не потрібна до завершення дії токену. Допоки користувач не здійснить авторизацію.

Меню логіна (рис. 4.1) відображається поверх основного вмісту сторінки у блоці з прозоро-білим фоном, що блокує можливість взаємодії з іншими елементами сайту до успішної авторизації.

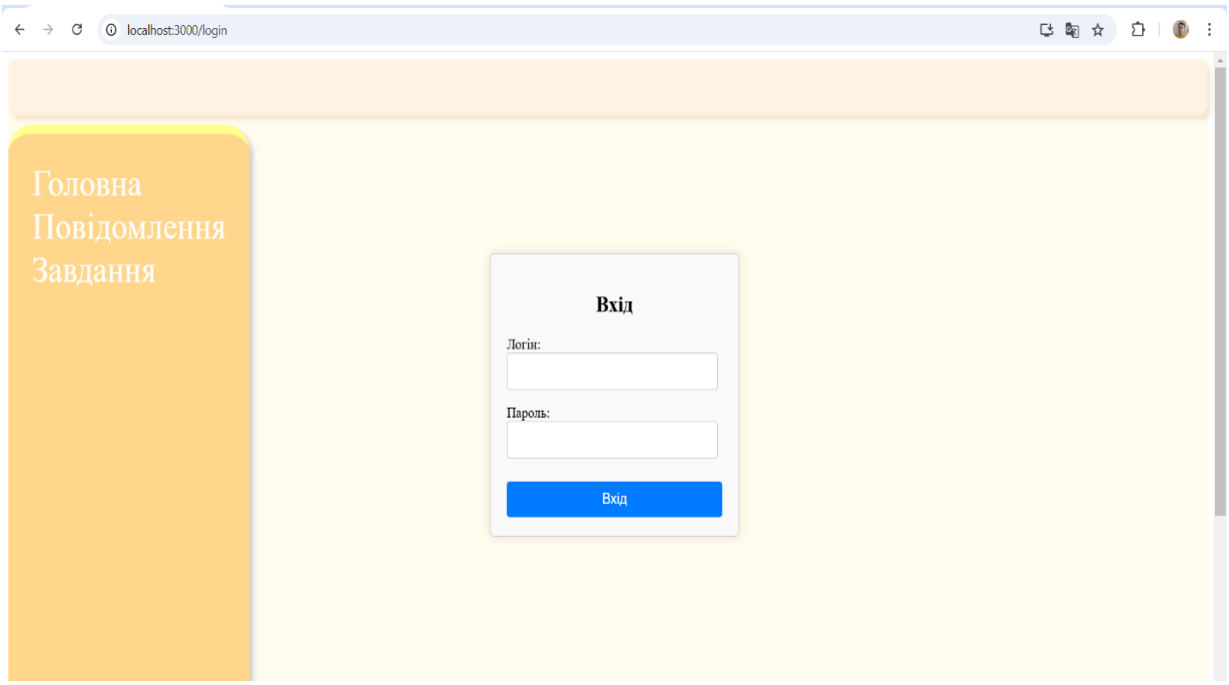

Рисунок 4.1 — Сторінка логіну

Такий підхід забезпечує чітке візуальне розмежування та допомагає користувачеві зосередитися на процесі реєстрації чи входу, створюючи зрозуміле враження щодо важливості цих кроків для подальшої взаємодії із сайтом.

Сторінка логіну складається із наступних структурних елементів:

1) контейнер сторінки:

— сторінка логіну містить контейнер з класом "overlay", який забезпечує фонову темну півтінь для підвищення контрасту та фокусування на вікні авторизації;

— усередині контейнера знаходиться основний елемент з класом "loginContainer", який містить усі елементи для введення даних та взаємодії з користувачем;

2) заголовок "Вхід":

— вгорі сторінки розташований заголовок з текстом "Вхід", який вказує на призначення сторінки. Заголовок оформлений через елемент <h2>;

3) форма для введення даних:

— форма авторизації організована в елементі <form>, що обробляє події подачі (submit) та виконує логіку входу через функцію handleLogin;

— поле для логіну: Текстове поле з ідентифікатором username, яке дозволяє користувачеві ввести свій логін, також воно вимагає обов'язкового заповнення та містить мітку "Логін", що допомагає користувачу зрозуміти, яку інформацію слід ввести;

— поле для паролю — текстове поле з типом "password" та ідентифікатором password має мітку "Пароль" і є обов'язковим для заповнення перед надсиланням форми;

4) кнопка "Вхід" — кнопка типу "submit" розташована після полів введення і дозволяє користувачеві подати форму. Кнопка має текст "Вхід", що чітко вказує на її функцію.

Після натискання кнопки "Вхід" система перевіряє введені дані. Якщо логін та пароль користувача правильні, то він автоматично перенаправляється на головну сторінку сайту. Якщо введені дані невірні, система виводить

повідомлення "Невірний логін або пароль!" за допомогою діалогового вікна alert, що інформує користувача про помилку у введених даних. Виведення загального повідомлення "Невірний логін або пароль!", замість того, щоб уточнювати, яке саме з введених значень (логін чи пароль) є неправильним, запобігає потенційним атакам, зокрема, від підбору паролів. Якщо система чітко вказує, чи неправильний логін або пароль, зловмисники можуть використати цю інформацію для визначення, яка частина їхніх даних помилкова, що спрощує спроби підбору правильного пароля. Загальне повідомлення робить процес підбору більш складним і менш ефективним для зловмисників, підвищуючи безпеку системи.

Головна сторінка додатку (рис 4.2) надає користувачеві зручний доступ до оперативної інформації про кількість отриманих завдань та швидкий перехід до відповідних розділів для їх перегляду та виконання. У шапці сайту справа є div-елемент із ім'я поточного користувача “Палько Артур Сергійович” і при кліку по ньому на екрані з'явиться меню із більш детальною інформацію про користувача, а саме: ім’я, назва підрозділу, пошта та мобільний телефон.

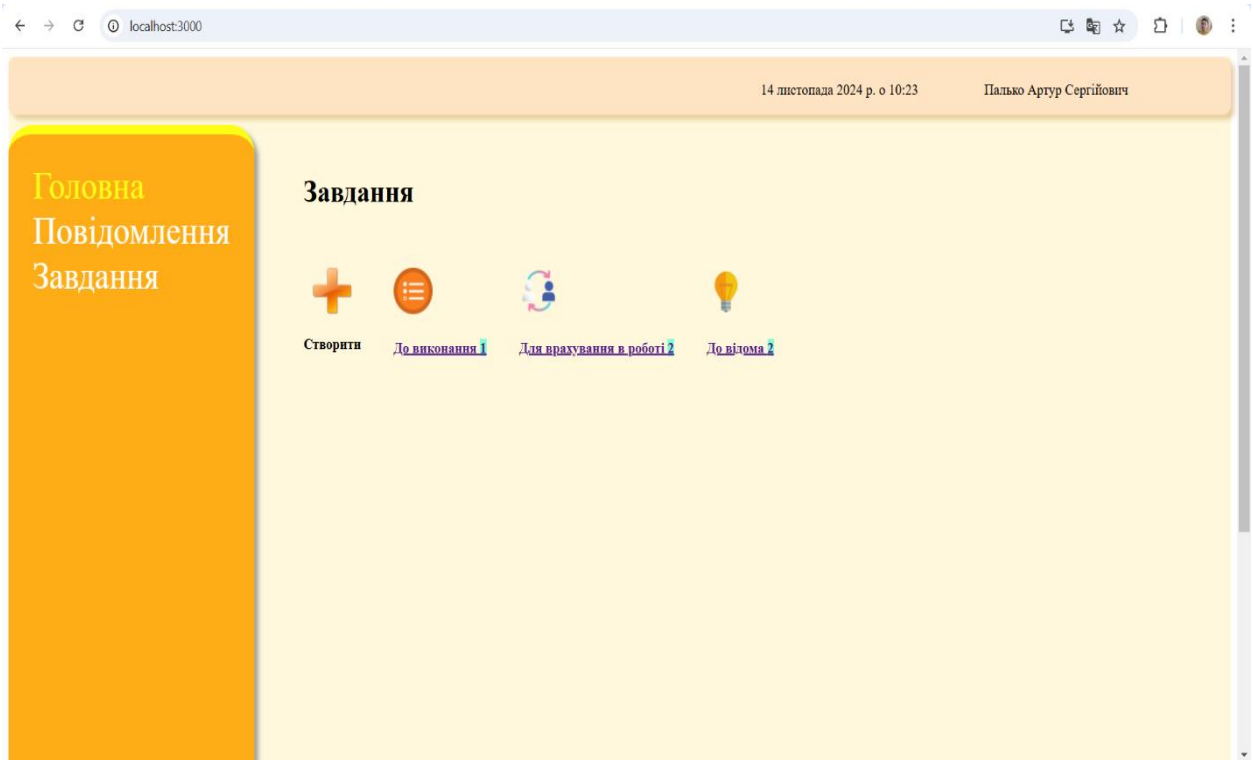

Рисунок 4.2 — Головна сторінка сайту

Також це меню містить кнопку “Вихід” для здійснення виходу із сайту, при кліку по ній відбувається вилогінення користувача шляхом видалення поточної куки із jwt-токеном (рисунок 4.3).

Загальний макет головної сторінки побудований за допомогою контейнера div з класом app-wrapper, що використовує CSS Grid для чіткого і зручного позиціонування елементів. Цей підхід дозволяє зручно розподіляти простір між секціями, а саме: шапкою сайту, навігаційним меню та основним вмістом. Важливим аспектом є використання властивості grid-template-areas, яка дає змогу легко визначити, де саме повинні розміщуватися кожна з цих частин. Шапка та навігаційне меню залишаються на кожній сторінці, тоді як вміст може змінюватися. У цьому макеті використовуються два рівні сітки: перший (верхній) для шапки та меню, другий (нижній) для вмісту.

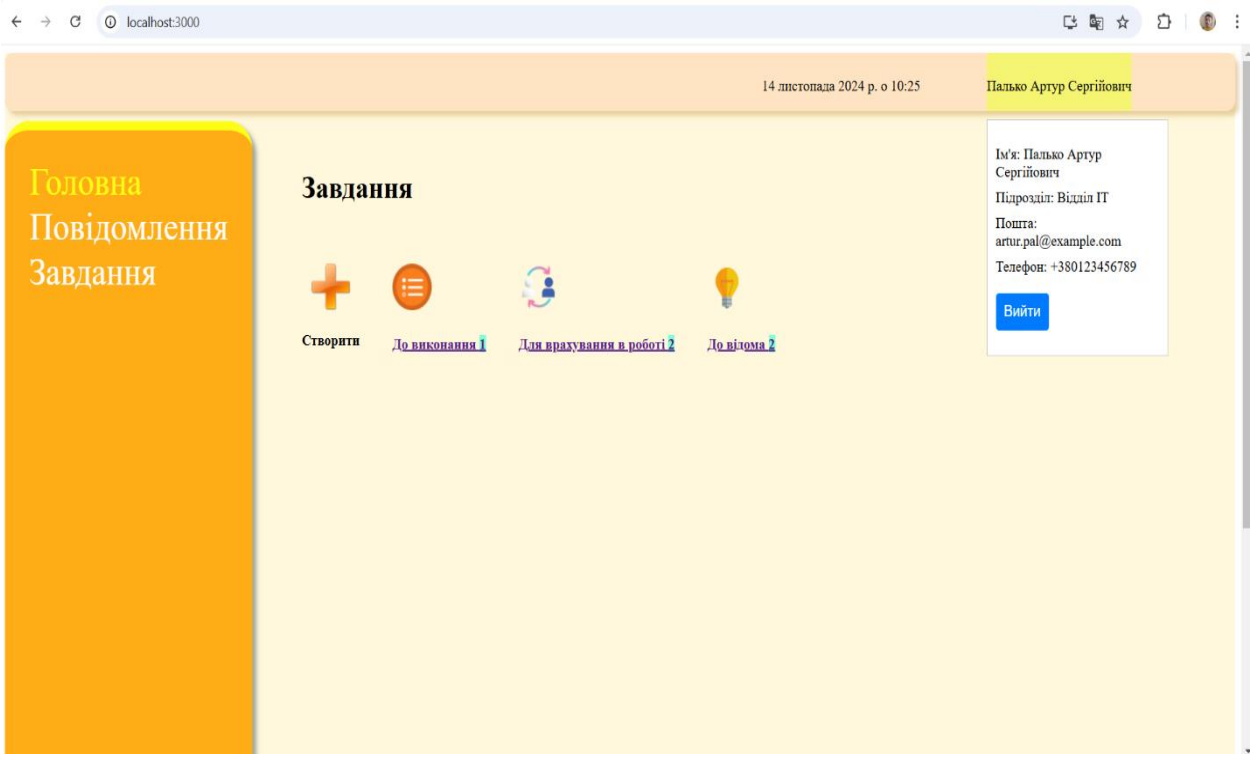

Рисунок 4.3 — Меню “Інформація про користувача”

Для шапки та меню використовуються ефекти, такі як тінь та округлені кути, що надає сторінці сучасного вигляду. Крім того, для меню визначено властивість opacity: 0.9, що надає йому напівпрозорість, створюючи враження легкості та

елегантності. CSS Grid [31] дозволяє зручно керувати розмірами та відступами між елементами, що робить макет адаптивним та зручним для користувача, забезпечуючи гармонійне поєднання фонових кольорів та відтінків. Використання CSS Grid дає величезну гнучкість в налаштуванні розміщення елементів, дозволяючи легко налаштовувати макет під різні екрани та пристрої.

Шапка сайту реалізується за допомогою компонента Header, який відповідає за відображення інформації про поточного користувача, а також поточної дати.

Навігаційне меню, яке реалізовано через бічне меню NavBar, розташоване ліворуч і надає навігацію по основних розділах сайту. Для цього використовуються елементи nav та div з класами nav-bar та navbar-container, що дозволяє користувачу переміщатися між сторінками (Головна, Завдання, Діалоги). Коли користувач переходить по певному лінку, до відповідного посилання додається CSS клас activeLink, який змінює стиль посилання, встановлюючи його колір gold, що допомагає користувачу орієнтуватися в тому, в якому розділі сайту він знаходиться.

Основний контент знаходиться в контейнері div з класом app-wrapper-content, де відображаються компоненти в залежності від вибраного шляху. Наприклад, коли користувач заходить на сторінку входу, рендериться компонент LoginPage, який відповідає за форму для введення логіну та паролю. При переході на сторінку задач відображається компонент Tasks, що містить список задач.

Вміст головної сторінки відображає заголовок "Завдання" та блоки для кожної категорії завдань. Створення нового завдання відбувається через іконку та кнопку, а кожна категорія завдань містить іконку та лічильник завдань цієї категорії. Для кожної категорії завдань (наприклад, "До виконання") використовується компонент NavLink, що дозволяє переходити до відповідної сторінки з відповідними завданнями.

Клацнувши на навігаційні панелі по посиланню “Завдання”, користувач переходить до меню де відображаються усі отримані та створені користувачем завдання (рисунок 4.4).

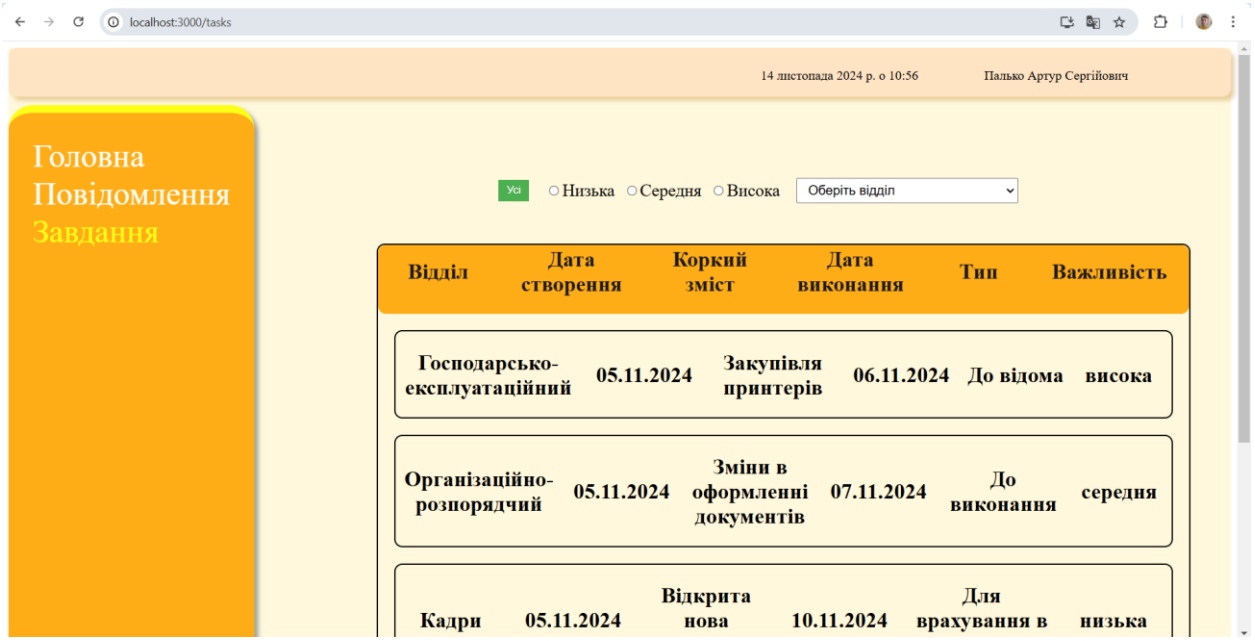

Рисунок 4.4 — Сторінка “Завдання”

Опис зовнішнього вигляду сторінки завдань:

1) Outlet — це вставка дочірнього маршруту, якщо він існує, і використовується для відображення завдань із певним типом, виконуючи рендеринг цієї ж сторінки із заздалегідь відфільтрованими даними;

2) TaskFilterMenu — компонент для вибору фільтрів, який передає обробник handleFilterChange через пропс onFilterChange, і дозволяє виконати фільтрацію завдань за важливістю (висока, середня, низька) та підрозділом користувача, що надіслав це завдання;

3) список завдань організований через компонент TaskHeader, який відповідає за відображення заголовків стовпців меню завдань, що полегшує навігацію для користувача, а кожне окреме завдання представлено елементом TaskItem, де основною структурою є контейнер div з класом taskTable, а основні дані завдання відображаються у вигляді таблиці, яка складається з tbody з одним рядком (tr), де кожен рядок містить комірки (td) з інформацією про завдання, включаючи такі поля, як;

— department (департамент, відповідальний за завдання);

— sentAt (дата/час відправлення завдання);

— description (короткий опис завдання);

— deadline (дата виконання завдання);

— goal (мета завдання);

— importance (важливість завдання).

Після подвійного кліку по завданню відкривається карта завдання, яка статично відображається по центру екрана (рисунок 4.5). Опис зовнішнього вигляду карти завдання "Перегляд" включає навігаційний блок, який реалізовано через з двома , що змінюють активний розділ, умовний рендеринг для перегляду завдання, який відображає інформацію про завдання, текстові поля, статус і результати, умовний рендеринг для прогресу виконання, що показує заголовок прогресу та список виконавців, а також кнопки управління, які відображаються тільки у режимі перегляду завдання.

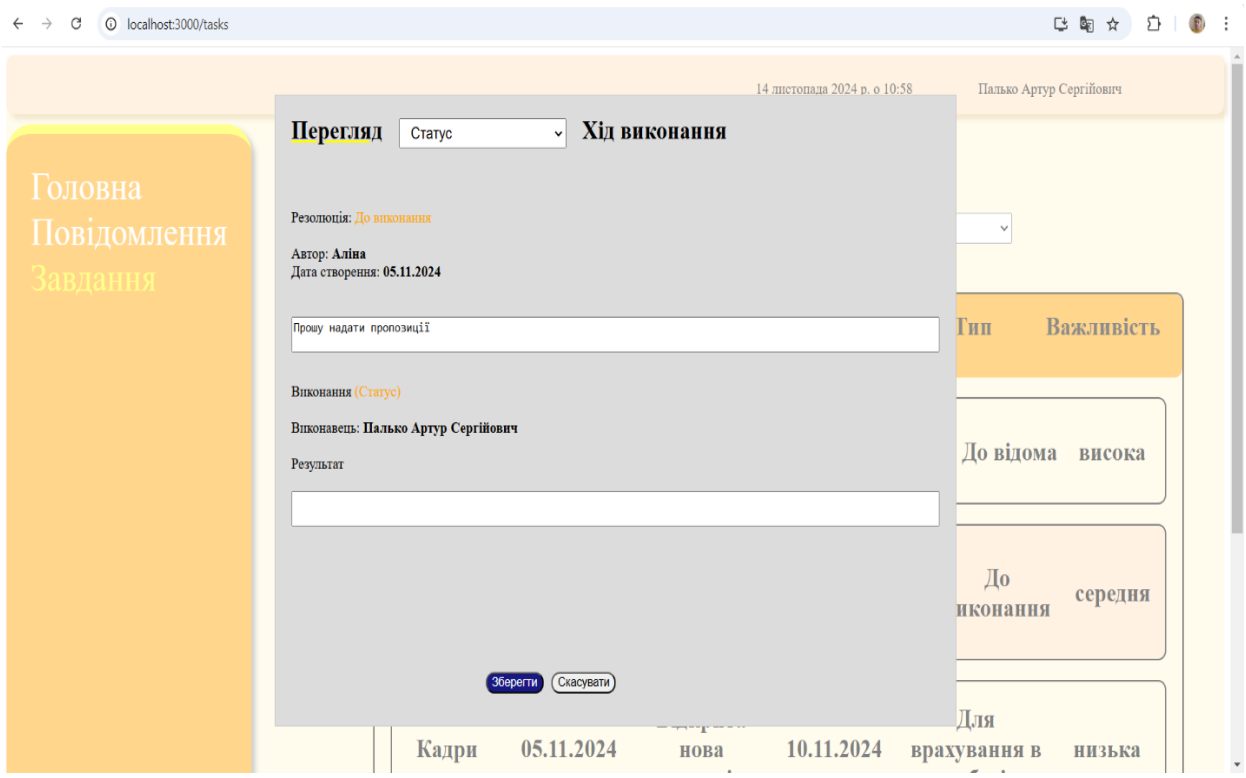

Рисунок 4.5 — Карта завдання “Зміни в оформленні документів”

У цей момент доступ до всіх інших елементів меню завдань блокується, однак вони залишаються видимими на фоні, створюючи ефект затемнення, подібний до вікна логіну, який був описаний раніше. Такий підхід дозволяє користувачу зосередитися на деталях завдання, не втрачаючи при цьому

контексту всього списку. Це допомагає зберігати орієнтацію у структурі завдань і забезпечує зрозумілу взаємодію з системою, акцентуючи увагу саме на вибраному завданні без відволікаючих факторів.

Структура карти завдання “Перегляд”:

— для відображення прогресу виконання завдання використовується окремий пункт "Хід виконання";

— кожен елемент виконавця представляється компонентом Progres, який виводить інформацію про виконавця — ім'я, статус виконання та термін.

Загалом, карта завдання складається із двох вкладок. На вкладці “Перегляд” відображається така інформація: автор завдання, дата створення завдання, повний текст завдаанння. Користувач може вибрати статус виконання завдання “Взяти на виконання”, “Виконати” або ж “Припинити виконання”. Перейшовши на вкладку “Хід виконання” користувач може переглянути дерево завдання, що представляє собою графічну схему, яка відображає усіх виконавців завдання та статус виконання кожного із них (рисунок 4.6).

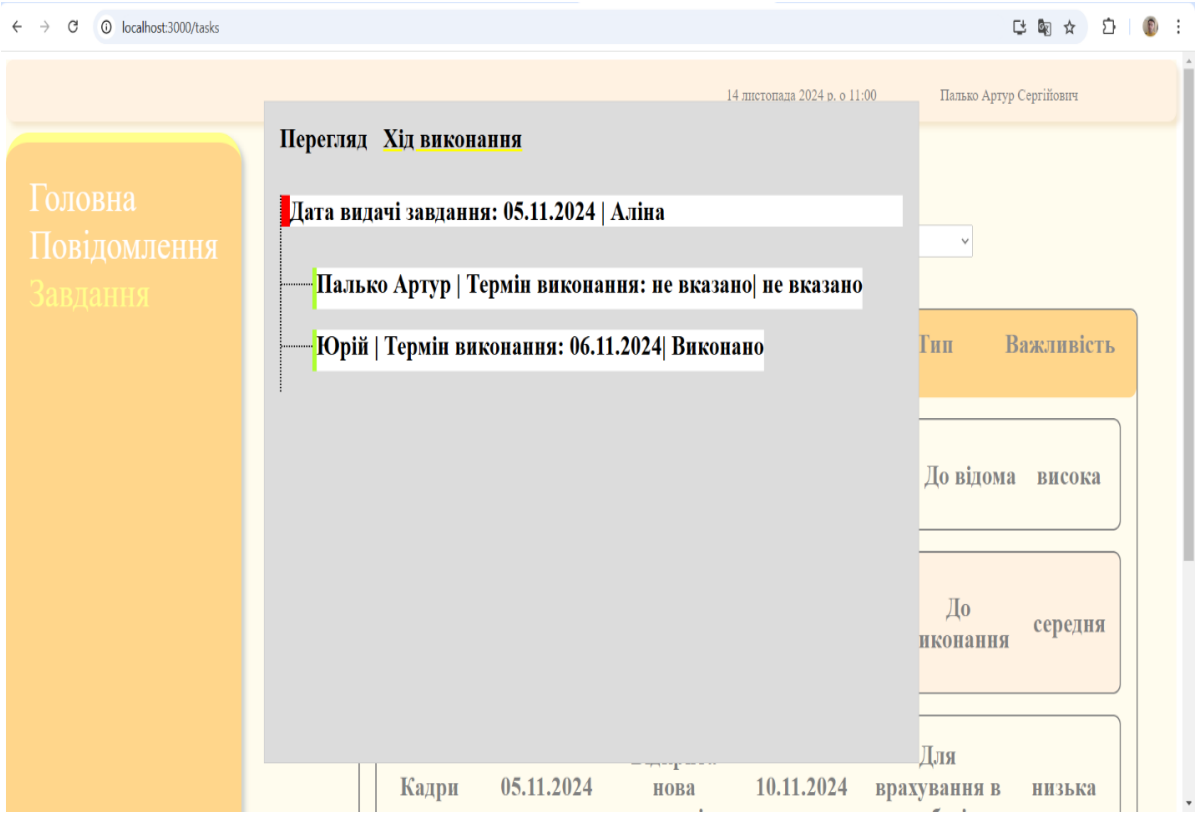

Рисунок 4.6 — Хід виконання завдання “Зміни в оформленні документів”

Візуалізація зв'язків виконавців реалізована наступним чином:

— кожен виконавець відображається окремо у компоненті Progres, де використовується стилізація для поділу завдань за допомогою пунктиру;

— в компоненті Progres присутня обгортка div із класом s.borderFlex, яка додає візуальний розділювач між виконавцями;

— елементи всередині мають відступи та стилі, що створюють візуальний поділ між кожним виконавцем, і таким чином виглядають як з'єднані пунктиром.

Проведемо експеримент зі створення нового завдання, для цього повернемось на головну сторінку та на панелі “Завдання” оберемо пункт меню “Створити”(рисунок 4.7).

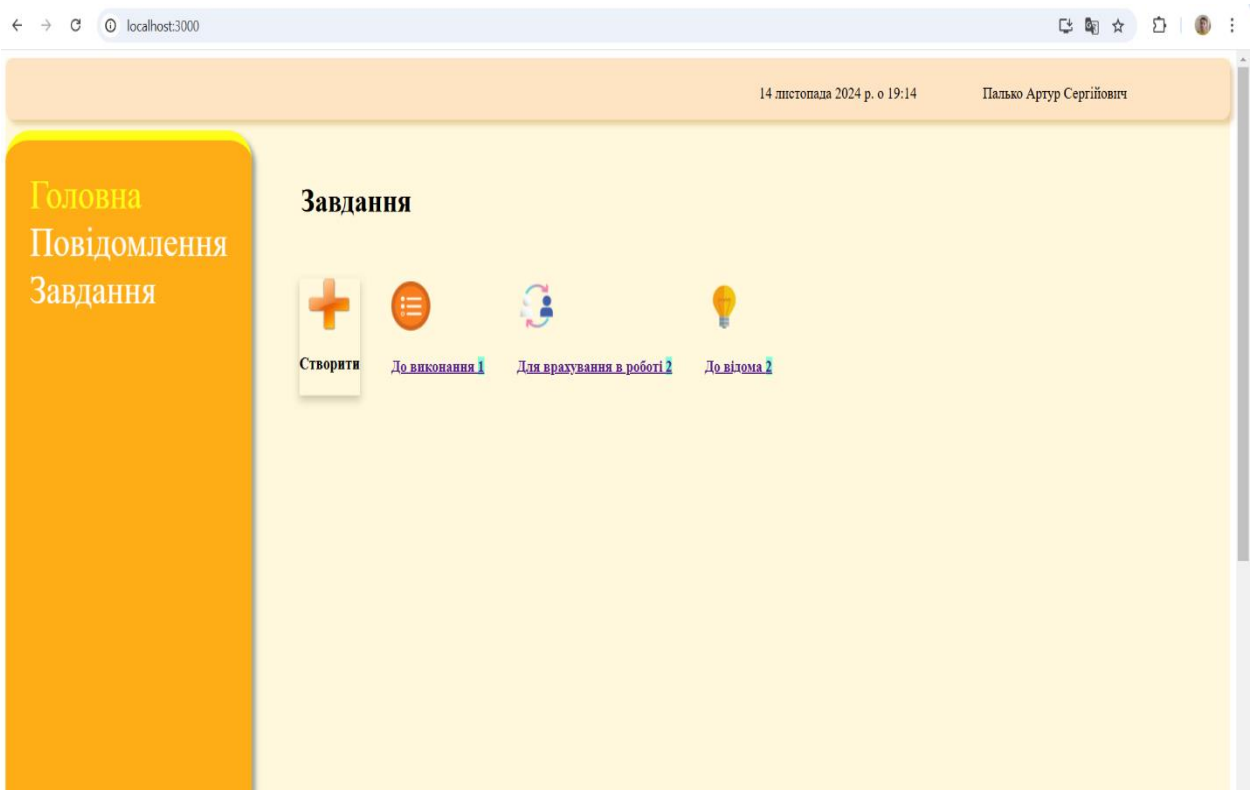

Рисунок 4.7 — Пункт меню “Створити завдання”

Після чого користувач переходить до модального вікна “Створити завдання” Це вікно складається із двох вкладок “Завдання” (рисунок 4.8) та “Погодження” (рисунок 4.9), на першій вкладці користувач може задати опис завдання, а на другій – призначити виконавців завдання.

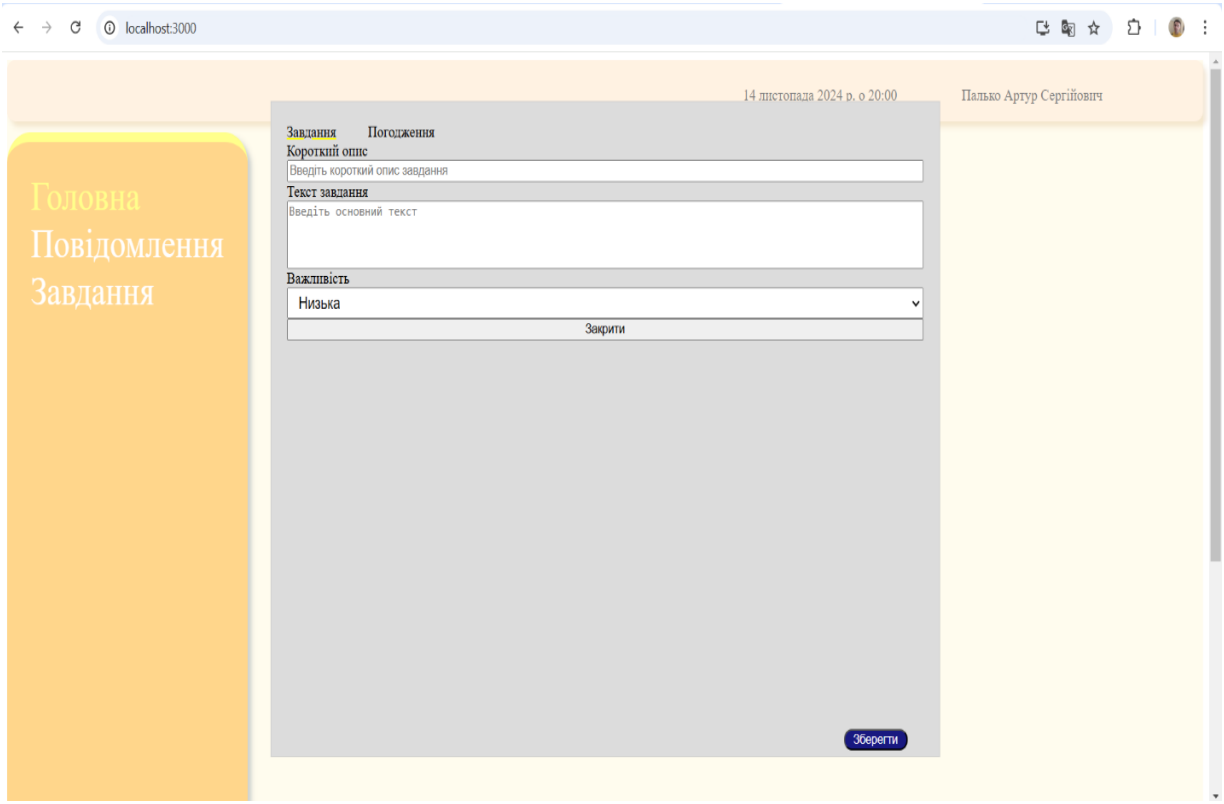

Рисунок 4.8 — Модальне вікно, вкладка “Завдання”

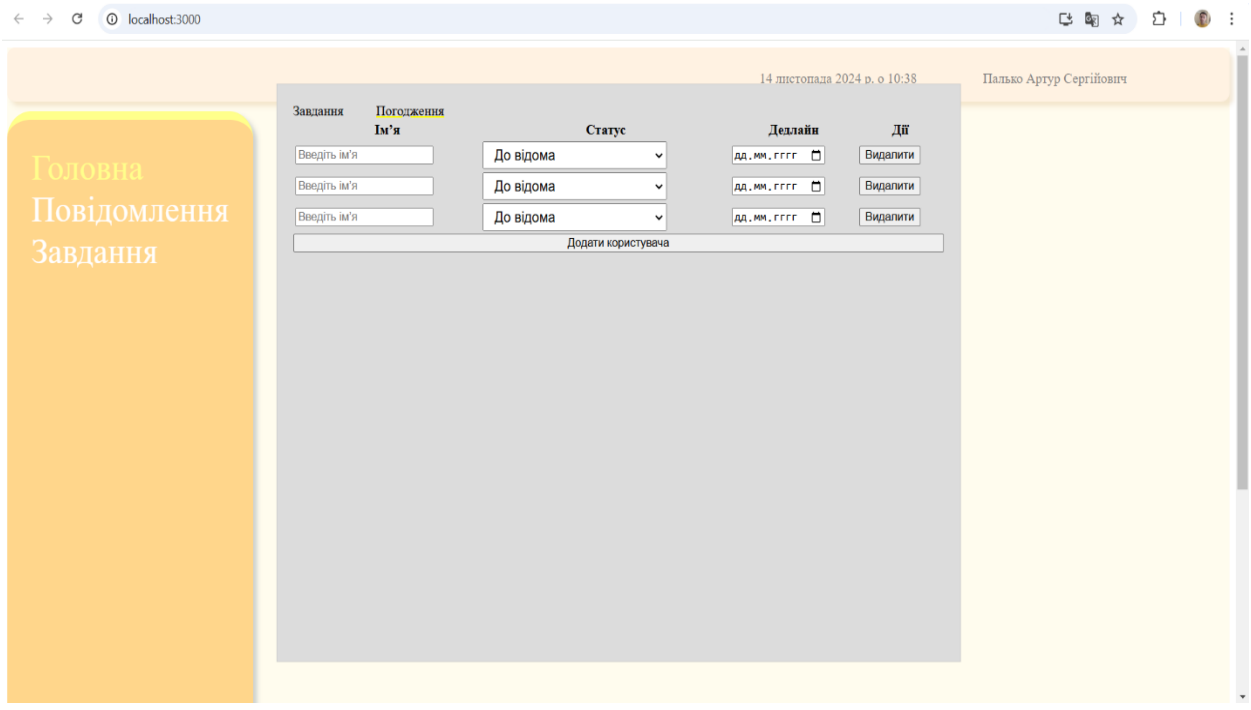

Рисунок 4.9 — Модальне вікно, вкладка “Погодження”

Опис зовнішнього вигляду модального вікна “Створити завдання”:

1) Секція "Завдання":

— короткий опис — текстове поле для введення опису завдання;

— текст завдання — текстове поле для введення основного тексту завдання;

— важливість завдання: випадаючий список для вибору важливості;

2) Секція "Погодження" містить таблицю де можна додавати завдання, змінювати та видаляти наступні поля:

— ім'я — текстове поле для введення імені завдання;

— статус — випадаючий список з можливими статусами ("До відома", "Для врахування в роботі", "До виконання");

— дедлайн — поле для вибору дати;

— дії — кнопка для видалення виконавця;

— додати користувача: кнопка яка створює поля для введення нового користувача.

Тобто, секція "Завдання" забезпечує внесення основної інформації про завдання, що є ключовою для його формування. Користувач має можливість ввести короткий опис, який слугує загальним представленням завдання, а також основний текст, що деталізує його зміст. Важливість завдання визначається через випадаючий список, що дозволяє систематизувати завдання за пріоритетами і спростити подальшу обробку та відстеження.

Секція "Погодження" спрямована на деталізацію та розподіл завдань між виконавцями. Основним інструментом тут є таблиця, яка надає гнучкий функціонал для управління записами. Користувач може додавати нові завдання, редагувати вже існуючі або видаляти непотрібні. Для кожного запису в таблиці передбачені поля для введення імені виконавця, вибору статусу завдання та встановлення дедлайну. Використання статусів, представлених як опції у випадаючому списку ("До відома", "Для врахування в роботі", "До виконання"), дає змогу чітко визначати, як саме має бути оброблене завдання.

Додатковий функціонал включає можливість динамічного додавання нових виконавців за допомогою кнопки "Додати користувача". Це дозволяє розширювати таблицю відповідно до потреб і кількості залучених осіб. Для

забезпечення контролю передбачена опція видалення виконавця через окрему кнопку в стовпці "Дії".

Для того щоб перейти на сторінку діалогів (рисунок 4.10) потрібно клацнути на навігаційній панелі по посиланню “Повідомлення Після цього відкриється сторінка, на якій будуть відображені всі діалоги користувача. У лівій колонці буде список контактів із користувачами системи, а праворуч - список останніх повідомлень для кожного діалогу.

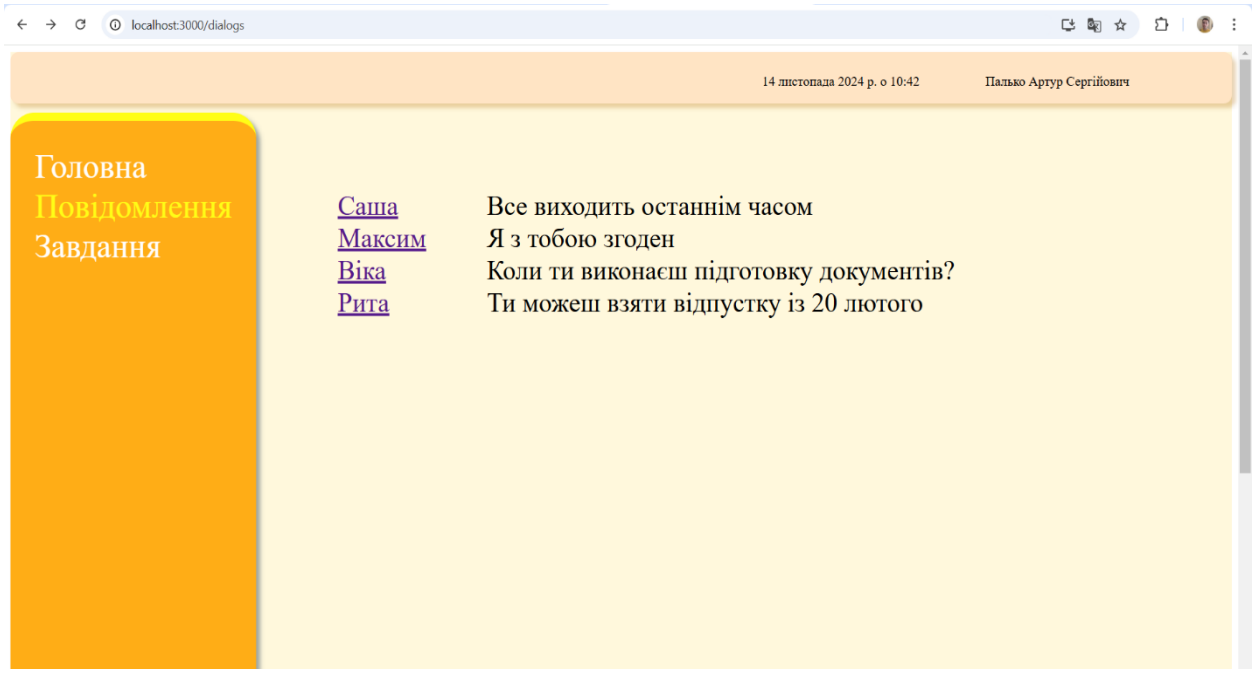

Рисунок 4.10 — Сторінка “Діалоги”

Клацнувши на імені користувача, ми перейдемо до відповідного діалогу, що відкриє сторінку чату (рисунок 4.11). Інтерфейс чату побудований таким чином, щоб забезпечити зручне спілкування та чітке візуальне розділення повідомлень між учасниками діалогу. Повідомлення, надіслані поточним користувачем, відображаються у лівій частині екрану та виділяються рожевим фоном, що допомагає користувачу легко орієнтуватися у власних повідомленнях. Водночас повідомлення, отримані від іншого користувача, також розміщуються ліворуч, проте для їхнього відображення використовується сірий фон, що створює контраст і забезпечує візуальне розрізнення між вхідною та вихідною кореспонденцією.

Для максимальної зручності під час введення тексту на сторінці передбачене поле введення повідомлення, яке динамічно розширюється в міру збільшення обсягу введеного тексту. Це дозволяє комфортно працювати як із короткими, так і з довгими повідомленнями без необхідності прокрутки або перемикання між інтерфейсами. Поруч із полем введення розташована кнопка для відправки повідомлення, яка забезпечує миттєву передачу введеного тексту до чату. Додатково інтерфейс може передбачати можливість відправлення повідомлення натисканням клавіші Enter, що прискорює процес комунікації та робить його більш інтуїтивним для користувача.

Загалом, структура сторінки чату поєднує в собі функціональність і мінімалістичний дизайн, що сприяє ефективному та комфортному обміну інформацією між користувачами.

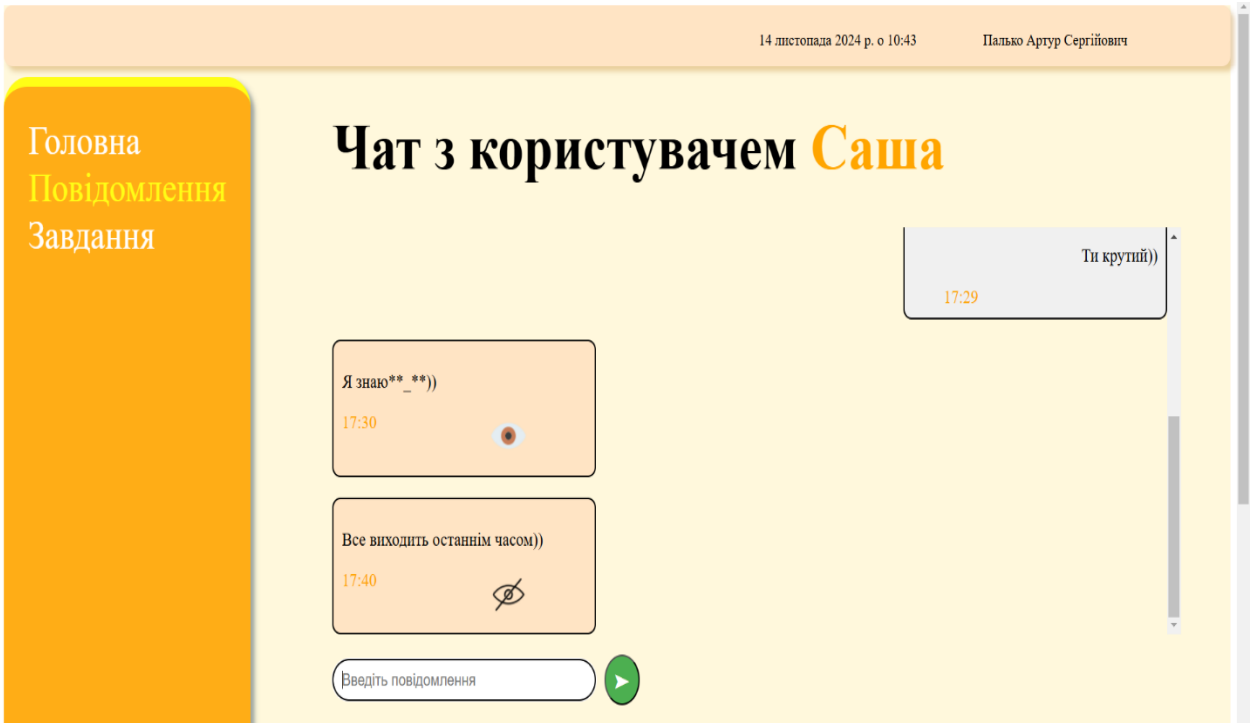

Рисунок 4.11 — Сторінка “Чат з користувачем”

Сторінка "Діалоги" має наступну структуру: компонент MessageItem є підкомпонентом, який використовується для відображення одного повідомлення і відповідає за рендеринг тексту повідомлення, часу його відправлення та іконки

перегляду, яка є кольоровою, якщо повідомлення було переглянуте адресатом, або сірою, якщо не переглянуте.

4.2 Перевірка роботи застосунку на тестових сценаріях

Спробуємо ввести наступні тестові значення для входу: логін “db” та пароль “password12”. Після введення цих даних і спроби авторизації система перевірить їх на відповідність вимогам валідації. У результаті отримаємо повідомлення про те, що введені дані не пройшли валідацію через некоректний формат логіну або паролю (рисунок 4.12).

Спробуємо ввести правильні тестові значення (рисунок 4.13) для входу: логін “ПалькоАС” та пароль “adf”. Після введення цих даних і спроби авторизації система перевірить їх на відповідність вимогам валідації. У результаті користувач одразу буде перенаправлений на головну сторінку додатку(рисунок 4.14).

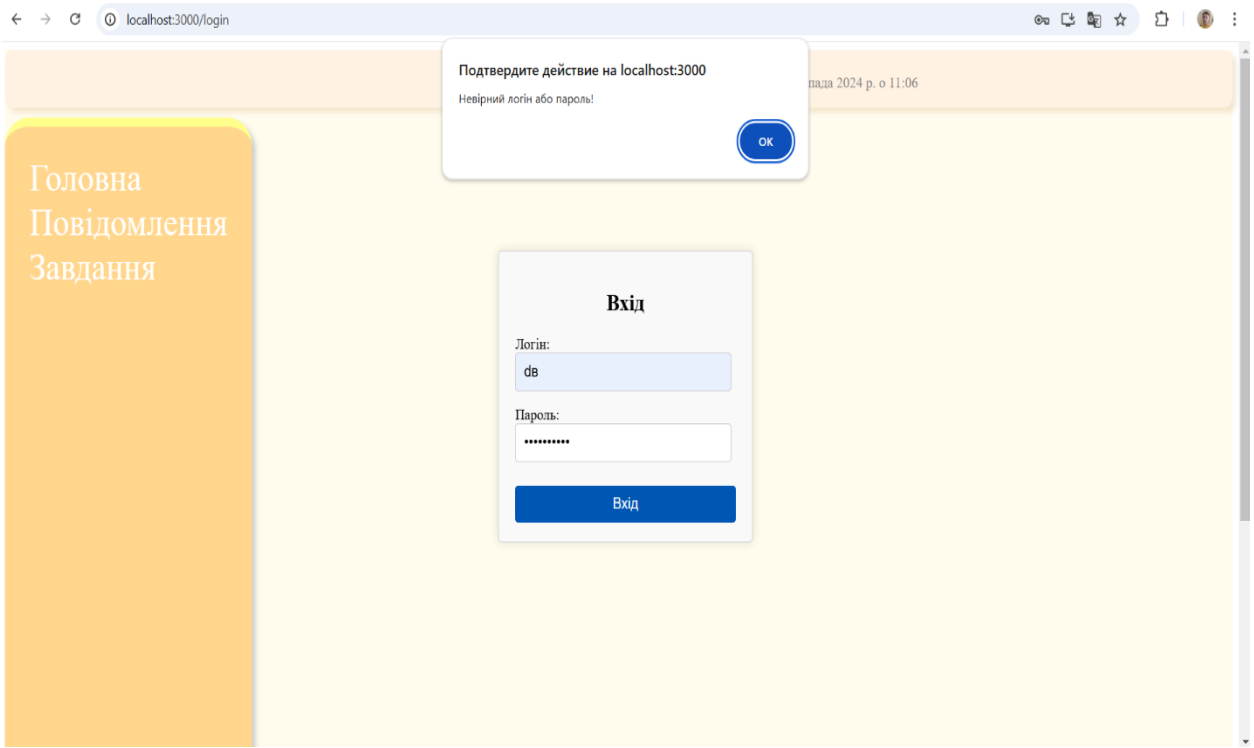

Рисунок 4.12 — Валідація входу (із неправильними даними)

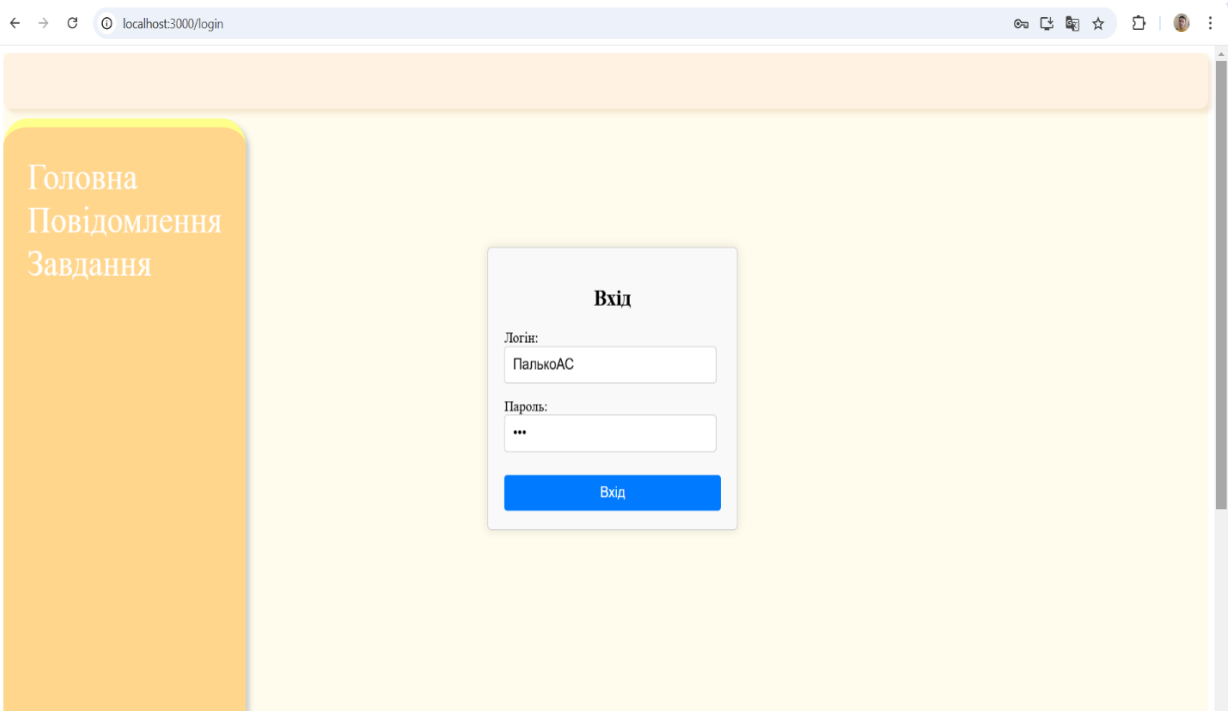

Рисунок 4.13 — Валідація входу (із правильними даними)

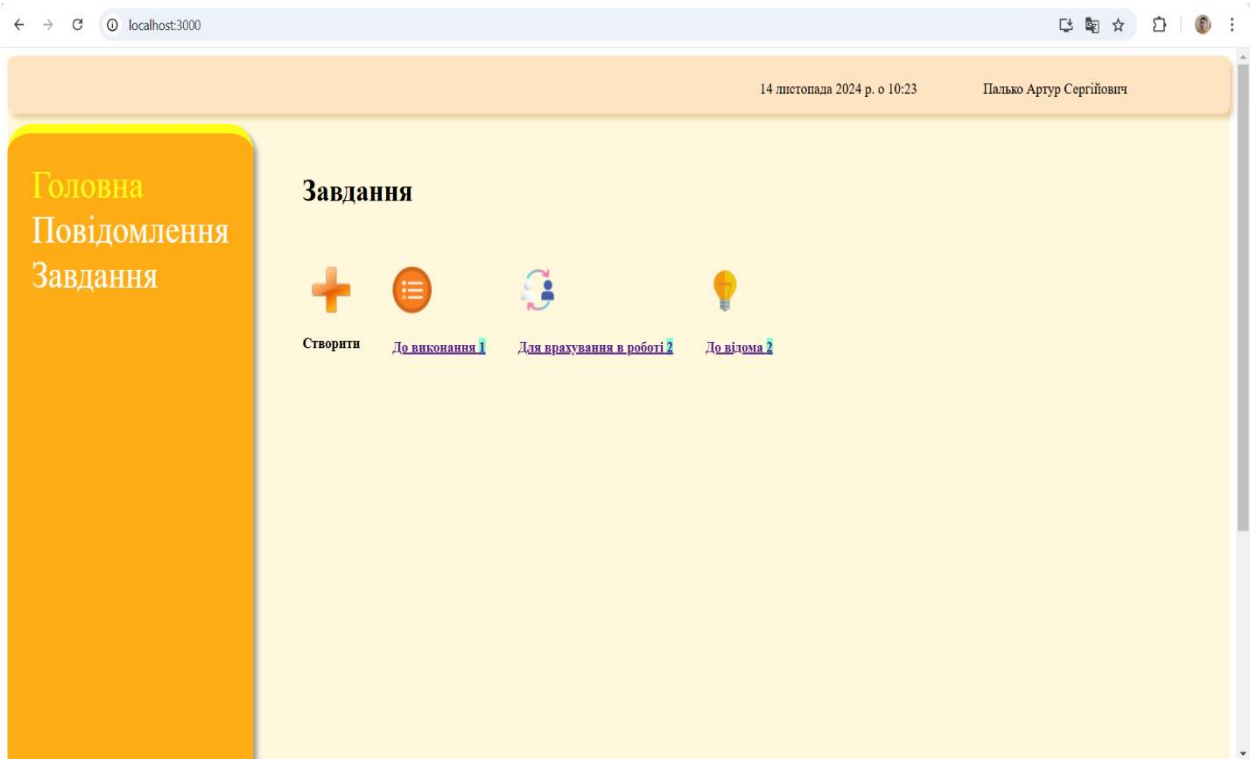

Рисунок 4.14 — Головна сторінка сайту

Перевіримо роботу меню, обравши пункт “Для врахування в роботі”, клікнувши у меню “Завдання” по блоку “Для врахування в роботі” (рисунок 4.15).

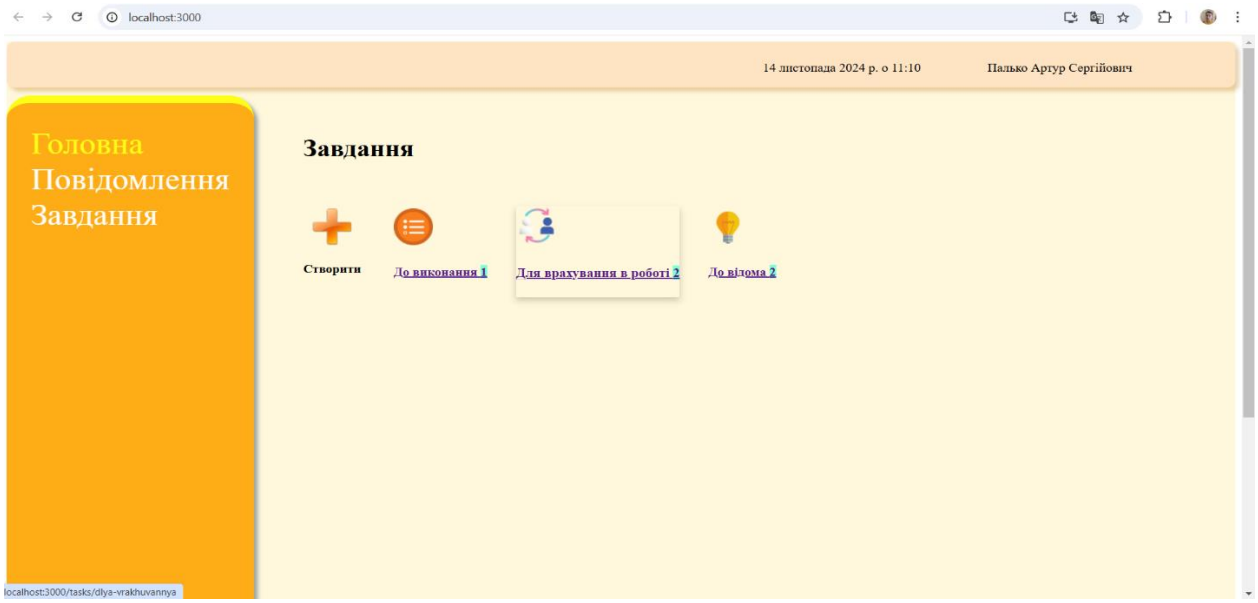

Рисунок 4.15 — Пункт меню “Для врахування в роботі”

Як результат відбувся успішний перехід до меню “Завдання”, де відображаються тільки ті завдання що мають тип “Для врахування в роботі” (рисунок 4.16).

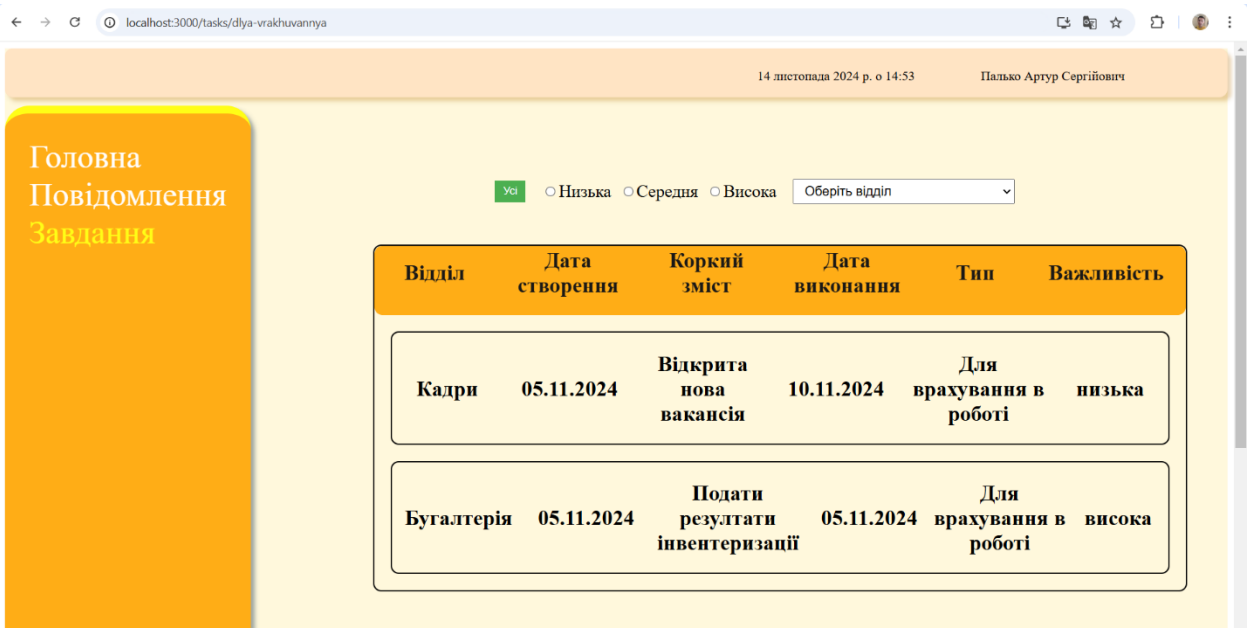

Рисунок 4.16 — Завдання відсортовані по типу “Для врахування в роботі”

Перевіримо роботу фільтру завдань, обравши спочату завдання із середньою важливістю (рисунок 4.17).

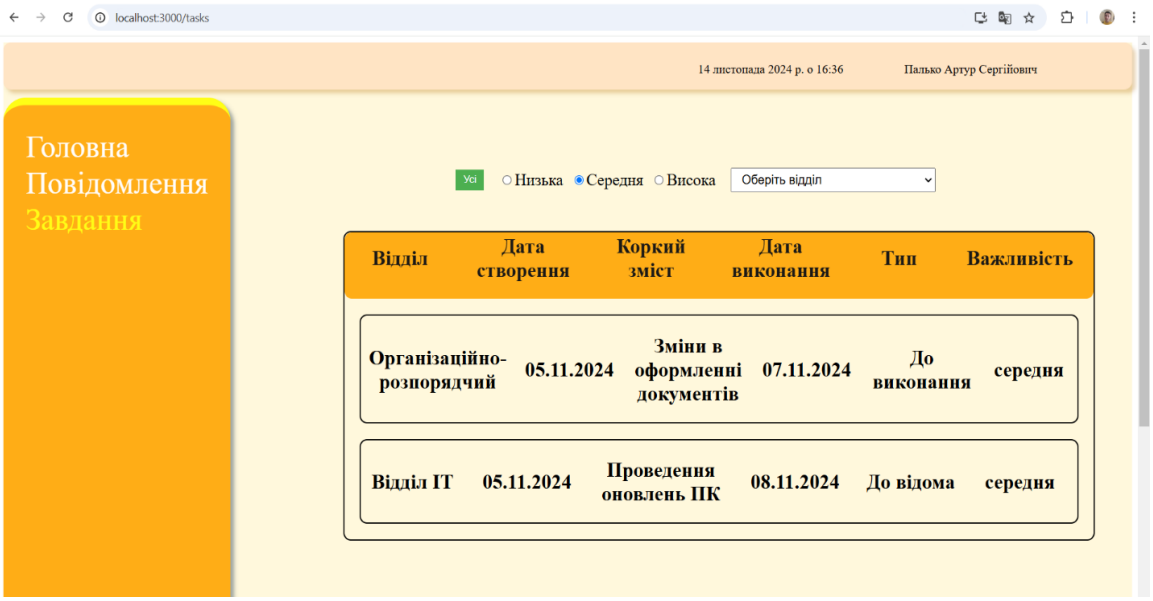

Рисунок 4.17 — Завдання відсортовані по значенню важливості “Середня”

Як результат, після обрання відповідної радіокнопки фільтру таблиця завдань містить лише ті завдання що мають середню важливість. Проведемо ще один експеримент, обравши відділ що надіслав завдання “Кадри” (рисунок 4.18).

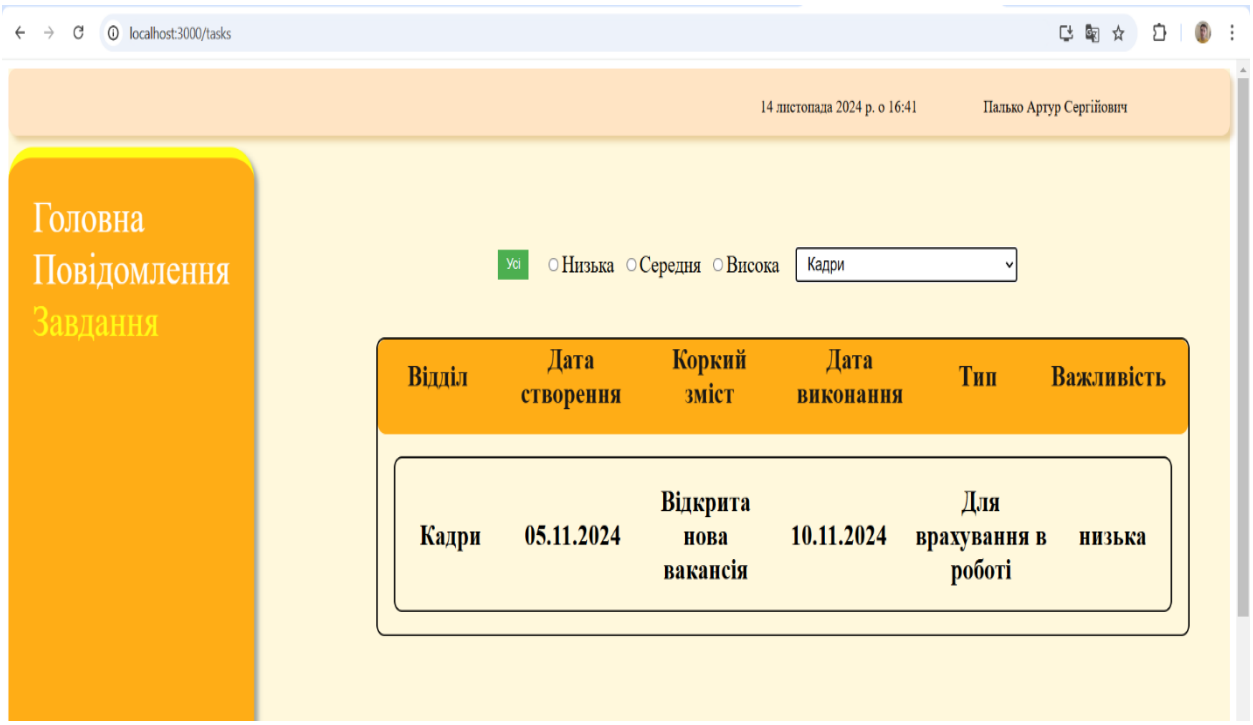

Рисунок 4.18 — Завдання відсортовані по відділу “Кадри”

Як результат, після обрання відділу “Кадри” із випадаючого списку фільтра у меню відображаються тільки ті завдання, що були створені працівником цього

відділу.

Переглянемо більш детальну інформацію про завдання “Зміни в оформленні документів” від організаційно-розпорядчого відділу після наведення на елемент завдання його колір змінюється на рожевий (рисунок 4.19).

Після подвійного кліку по завданню відкривається карта завдання, яка статично відображається по центру екрана.

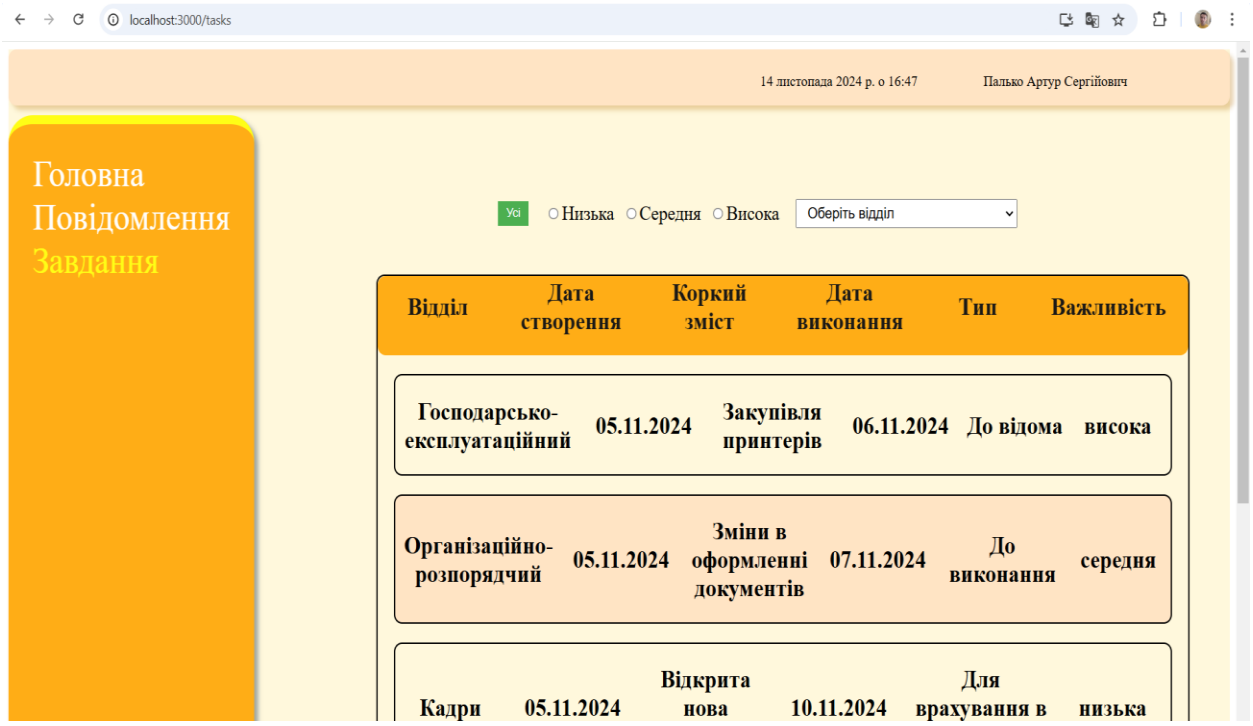

Рисунок 4.19 — Вибір завдання “Зміни в оформленні документів”

У цьому розділі було проведено комплексне тестування додатку, яке охоплювало перевірку функціонування механізму логінізації та доступу до ключових розділів сайту, зокрема сторінок "Головна", "Завдання" та "Діалоги".

Додатково оцінено коректність роботи основних функцій додатку, таких як облік вхідних завдань, створення нових завдань, їх призначення виконавцям та можливість відстеження виконання власних завдань.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку веб-додатку із функціями діалогів між користувачами, створення і розсилки завдань та контроль за їх виконанням. Для впровадження у локальній мережі підприємства. Особливістю розробки є те, що додаток об'єднує ключові підрозділи підприємства для управління внутрішніми комунікаціями, впровадженні механізмів опитувань для покращення процесу прийняття управлінських рішень та створенні власної системи комунікації в межах локальної мережі для підвищення інформаційної безпеки та збереження конфіденційності корпоративних даних.

Аналогом може бути Microsoft Teams – 24 000 грн/місяць.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах

Продовження табл. 5.1

Ринкові переваги					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною .	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	4	4
Наявність аналогів на ринку	3	3	4
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	4	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	4	4	3
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	3	4	4
Супровідна документація	4	3	4
Сума	43	42	44
Середньоарифметична сума балів	(43+42+44) / 3 = 43		

За даними таблиці 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.3.

Таблиця 5.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок автоматизація процесів взаємодії працівників меблевого підприємства шляхом впровадження власного механізму та аналізу відповідних алгоритмів.

5.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

5.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t, \qquad (5.1)$$

де М – місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ – число робочих днів за місяць, 22 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.4.

Таблиця 5.4 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	32500	1477,27	32	47272,727
Програміст	30000	1363,64	32	43636,364
Всього				90909,09

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

5.2.2 Додаткова заробітна плата розробників, які брати участь в розробці обладнання/програмного продукту.

Додаткову заробітну плату прийнято розраховувати як 14,3 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 14{,}3\ \% / 100\ \% \qquad (5.2)$$

$З_д = (90909{,}09 \cdot 14{,}3\ \% / 100\ \%) = 13000{,}00$ (грн.)

5.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (5.3)$$

$Н_з = (90909{,}09 + 13000{,}00) \cdot 22\ \% / 100\ \% = 22860{,}00$ (грн.)

5.2.4. Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

5.2.5 Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \qquad (5.4)$$

де Ц – балансова вартість обладнання, грн.;

Т – термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ – термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 20000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,45 міс.

$$A_{обл} = \frac{20000}{2} \times \frac{1{,}45}{12} = 1212{,}12 \; грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці

5.5. Так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $В_{нем.ак.}$ = 8000 грн.

Таблиця 5.5 – Амортизаційні відрахування на матеріальні та нематеріальні ресурси для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія	20000	2	1,45	1212,121
Офісне обладнання (меблі)	28000	4	1,45	848,485
Приміщення	1200000	20	1,45	7272,727
Всього				9333,33

5.2.6 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \qquad (5.5)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства з ПДВ в 2024 році для Вінницької області за даними Енера-Вінниця, В = 10,42 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,3 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9.

$В_е = 0,9 \cdot 0,3 \cdot 8 \cdot 32 \cdot 10,42 = 720,2304$ (грн.)

5.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.6)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$I_в$ = 90909,09 * 80% / 100% = 72727,27 (грн.)

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.7)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$Н_{нзв}$ = 90909,09 * 130 % / 100 % = 118182 (грн.)

5.2.9 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$B_{заг}$ = 90909,09+13000,00+22860,00+9333,33+8000+720,23+72727,27+
+118182 = 335731,75 грн.

5.2.11 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} \text{ (грн)}, \quad (5.8)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$ЗВ$ = 335731,75 / 0,5 = 671463 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить

потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

в) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб'єктів (зацікавлених осіб) цього попиту;

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов'язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

- абсолютного економічного ефекту (чистого дисконтованого доходу);
- внутрішньої економічної дохідності (внутрішньої норми дохідності);
- терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об'єднати, враховуючи визначені ситуації з відповідними умовами.

5.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.9)$$

де $\pm\Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Цб$ – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

p – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2024 році ϑ =18%.

Припустимо, що при прогнозованій ціні 8200 грн. за одиницю, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *500* грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 3000 шт., протягом другого року – на 2000 шт., протягом третього року на 1000 шт. До

моменту впровадження результатів наукової розробки реалізації продукту не було:

*ΔП₁ = (0*500 + (*8200 + 500 *)*3000)* 0,8333* 0,27) * (1 - 0,18) = 4538699,818 грн.*

*ΔП₂ = (0*500 + (*8200 + 500 *)*(3000+2000)* 0,8333* 0,27) * (1 - 0,18) = 8025749,679 грн.*

*ΔП₃ = (0*500 + (*8200 + 500 *)*(3000+2000+1000)* 0,8333* 0,27) * (1 - 0,18) = 9630899,615 грн.*

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 22195349,11 грн.

5.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t – період часу (в роках).

Збільшення прибутку ми отримаємо, починаючи з першого року:

ПП = (4538699,818/(1+0,1)[1])+(8025749,679/(1+0,1)[2])+(9630899,615/

$/(1+0,1)^3) = 4126090,74 + 6632850,974 + 7235837,427 = 17994779,14$ грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (5.11)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$PV = 2 * 671463 = 1342926,98$ *грн.*

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \quad (5.12)$$

$E_{абс} = 17994779,14 - 1342926,98 = 16651852,16$ грн.

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої но-рми

дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.13}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 16651852{,}16/1342926{,}98} - 1 = 1{,}375$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{5.14}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = (0,09…0,14);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0{,}14 + 0{,}05 = 0{,}19.$$

Так як $E_в > \tau_{\min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \quad (5.15)$$

$T_{ок}$ = 1 / 1,375 = 0,73 р.

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,73 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 671463 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,73 роки.

ВИСНОВКИ

У результаті виконання магістерської роботи було створено інтегровану комп’ютерну систему для взаємодії співробітників меблевого підприємства. Було проаналізовано популярні системи комунікації, такі як Jira, Teams та АСКОД, що допомогло визначити функції, які варто впровадити у додаток. Розглянуто вимоги корпоративної етики, конфлікти та їх вирішення, що стало основою для додавання функціоналу обговорення завдань.

Особливу увагу приділено інформаційній безпеці, проаналізовано загрози кібератак та витоку даних, що призвело до рішення розгортати платформу в локальній мережі підприємства.

Серверна частина платформи була розроблена з використанням C# та ASP.NET, а Frontend реалізовано на React.js. Результатом став веб-додаток, який враховує специфіку комунікаційних процесів підприємства, підвищуючи ефективність роботи персоналу та забезпечуючи безпеку даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. А. С. Палько, Л. А. Савицька, М. Г. Тарновський. Інтегрована комп’ютерна система для оперативної взаємодії персоналу виробничого підприємства з виготовлення меблів// [Електронний ресурс] / Л. А. Савицька // Матеріали LIV Всеукраїнської науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії, Вінниця, 24-27 березня 2025 р. – Електрон. текст. дані. – 2024. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/schedConf/presentations

2. Професійна та корпоративна етика: навч. посіб. / за ред., В.І. Панченко. К: 2019 ВПЦ «Київський університет», 2019

3. Конфліктологія: навчальний посібник / Г. В. Гребеньков [та ін.]; ред. Г. В. Гребеньков. - Львів : Магнолія, 2018. - 228 с.

4. Психологія конфлікту: навчальний посібник / Л. В. Долинська, Л. П. Матяш-Заяц. - 3-є вид. - Київ : Каравела, 2016. - 303 с.

5. Система управління проєктами та завданнями Jira для команд розробників. URL: https://www.atlassian.com/software/jira (дата звернення: 01.10.2024)

6. Сервіс Microsoft Teams для організації робочих зустрічей і спільної роботи. URL:https://www.microsoft.com/uk-ua/microsoft-teams/group-chat-software?ms.url=teamscom (дата звернення: 01.10.2024)

7. Система електронного документообігу АСКОД у хмарі. URL: https://askod.ua/home (дата звернення: 01.10.2024)

8. Кібербезпека: визначення, основні принципи та важливість для захисту інформаційних систем. URL :https://www.microsoft.com/uk-ua/security/business/security-101/what-is-cybersecurity

9. Основи та практичні заняття з мови запитів до баз даних. URL: http://moonexcel.com.ua/ (дата звернення: 31.05.2023)

10. Computer Viruses: types, methods of spread, and protection measures. URL: https://en.wikipedia.org/wiki/Computer_virus (дата звернення: 01.11.2024)

11. Trojan viruses: decteing and removing.

URL: https://us.norton.com/blog/malware/what-is-a-trojan (дата звернення: 01.10.2024)

12. What’s the Difference Between LAN and WAN. URL: https://aws.amazon.com/compare/the-difference-between-lan-and-wan/#:~:text=LANs%20connect%20users%20and%20applications,locations%20(across%20the%20globe). (дата звернення: 01.10.2024)

13. Web application: definition, functionality, and development principles URL: https://spdload.com/blog/how-to-develop-a-web-application/ (дата звернення: 31.11.2024)

14. Client-Server acrchitecture. URL: https://www.qamadness.com/knowledge-base/client-server-architecture-for-qa-engineers/

15. Cookie-authentication URL: https://learn.microsoft.com/en-us/aspnet/core/security/cookie-sharing?view=aspnetcore-9.0

16. JWT-authentication URL: https://www.c-sharpcorner.com/article/jwt-json-web-token-authentication-in-asp-net-core/

17. IDE Visual Studio URL: https://visualstudio.microsoft.com/ru/vs/getting-started/ (дата звернення: 31.11.2024)

18. C#: мова програмування для розробки програмного забезпечення та створення додатків на платформі .NET. URL: https://uk.wikipedia.org/wiki/C_Sharp (дата звернення: 31.05.2023)

19. JavaScript: мова програмування для створення інтерактивних веб-сторінок і клієнтської логіки URL: https://uk.wikipedia.org/wiki/JavaScript

20. Visual Studio Code: lightweight integrated development environment for programming in multiple languages URL: https://code.visualstudio.com/

21. ASP .NET Core. Fraemwork documentation URL: https://learn.microsoft.com/en-en/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-9.0 (дата звернення: 31.05.2023)

22. React documentation [Електронний ресурс]: Режим доступу: https://react.dev/learn

23. ASP .NET Core + React template URL: https://react.dev/learn

24. EntityFramework tutorial URL: https://www.tutorialspoint.com/entity_framework/index.htm (дата звернення: 31.05.2023)

25. MongoDB documentation URL: https://www.mongodb.com/docs/manual/

26. Microsoft SQL Server documentation URL: https://ru.wikipedia.org/wiki/Microsoft_SQL_Server (дата звернення: 31.11.2024)

27. RedisDb documentation URL: https://devdocs.io/redis/

28. OracleDb documentation URL: https://docs.oracle.com/en/database/

29. PostgreSQL documentation URL: https://www.postgresql.org/docs/

30. Діаграми станів та переходів[Електронний ресурс]: Режим доступу: https://studopedia.su/13_68418_tema-dIagrami-perehodIv-stanIv.html

31. CSS Grid Template: Основи використання та визначення шаблонів для макетів сторінок URL: https://css.in.ua/css/property/grid-template

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д.

_____

" ____ " _____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

“Інтегрована комп’ютерна система для оперативної взаємодії персоналу виробничого підприємства з виготовлення меблів”

08-54.МКР.033.00.000 ПЗ

Керівник роботи к.т.н. доц. каф. ОТ

_____ Савицька Л. А.

Студент групи 2КІ–22м

_____ Палько А.С.

ВНТУ 2024

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Розробка інтегрованої платформи для взаємодії працівників меблевого підприємства є актуальною для вдосконалення алгоритмів розподілу завдань та оптимізації внутрішніх комунікацій. Використання локальної мережі забезпечить швидкий і безпечний обмін інформацією, підвищить продуктивність праці та зменшить кількість помилок у процесі виконання завдань.

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — автоматизація процесів взаємодії працівників меблевого підприємства шляхом впровадження власного механізму та аналізу відповідних алгоритмів.

2.2 Призначення розробки — створення інтегрованої платформи для управління завданнями та комунікаціями між працівниками меблевого підприємства. Платформа буде використовуватися для автоматизованого розподілу завдань, обміну повідомленнями та спільної роботи в локальній мережі підприємства, що сприятиме підвищенню ефективності та зниженню кількості помилок у виконанні завдань.

3 Вихідні дані для виконання МКР

3.1 Призначення системи — автоматизація процесів взаємодії працівників меблевого підприємства для підвищення ефективності комунікацій та управління завданнями.

3.2 Використовувані інструменти — веб-інтерфейс, локальна мережа підприємства, засоби для розробки алгоритмів розподілу завдань.

3.3 Організація — функціонування в межах локальної мережі підприємства з можливістю інтеграції з внутрішніми інформаційними системами.

3.4 Архітектура — централізована з можливістю подальшого

масштабування та адаптації під специфічні потреби підприємства.

4 Вимоги до виконання МКР

4.1 Провести аналіз існуючих алгоритмів розподілу завдань та комунікаційних платформ.

4.2 Визначити оптимальну архітектуру веб-платформи для локальної мережі меблевого підприємства.

4.3 Розробити механізми автоматизації розподілу завдань та функціонал для внутрішніх комунікацій.

4.4 Провести тестування ефективності та надійності платформи в умовах локальної мережі.

4.5 Оцінити потенціал платформи для підвищення продуктивності підприємства.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз існуючих технологій, огляд аналогів системи.	19.09.2024	15.10.2024	Вступ, Розділ 1
2	Визначення архітектури веб-платформи для локальної мережі меблевого підприємства.	16.10.2024	29.10.2024	Розділ 2

Продовження таблиці А.1

3	Розробка механізмів автоматизації розподілу завдань та функціоналу для внутрішніх комунікацій (схеми маршрутизації та діаграми станів проекту)	30.10.2024	07.11.2024	Розділ 2
4	Проектування бази даних	08.11.2024	19.11.2024	Розділ 3
5	Тестування ефективності та надійності платформи	20.11.2023	26.11.2023	Розділ 3
6	Підготовка економічної частини	27.11.2024	03.12.2024	Розділ 4
7	Оформлення пояснювальної записки, графічного матеріалу і презентації	04.12.2024	10.12.2024	ПЗ, графічний матеріал і презентація
8	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	11.12.2024	15.12.2024	Оформленні документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлюванні МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104–2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ–03.02.02 П.001.01:21

ДОДАТОК Б

Діаграма станів проекту

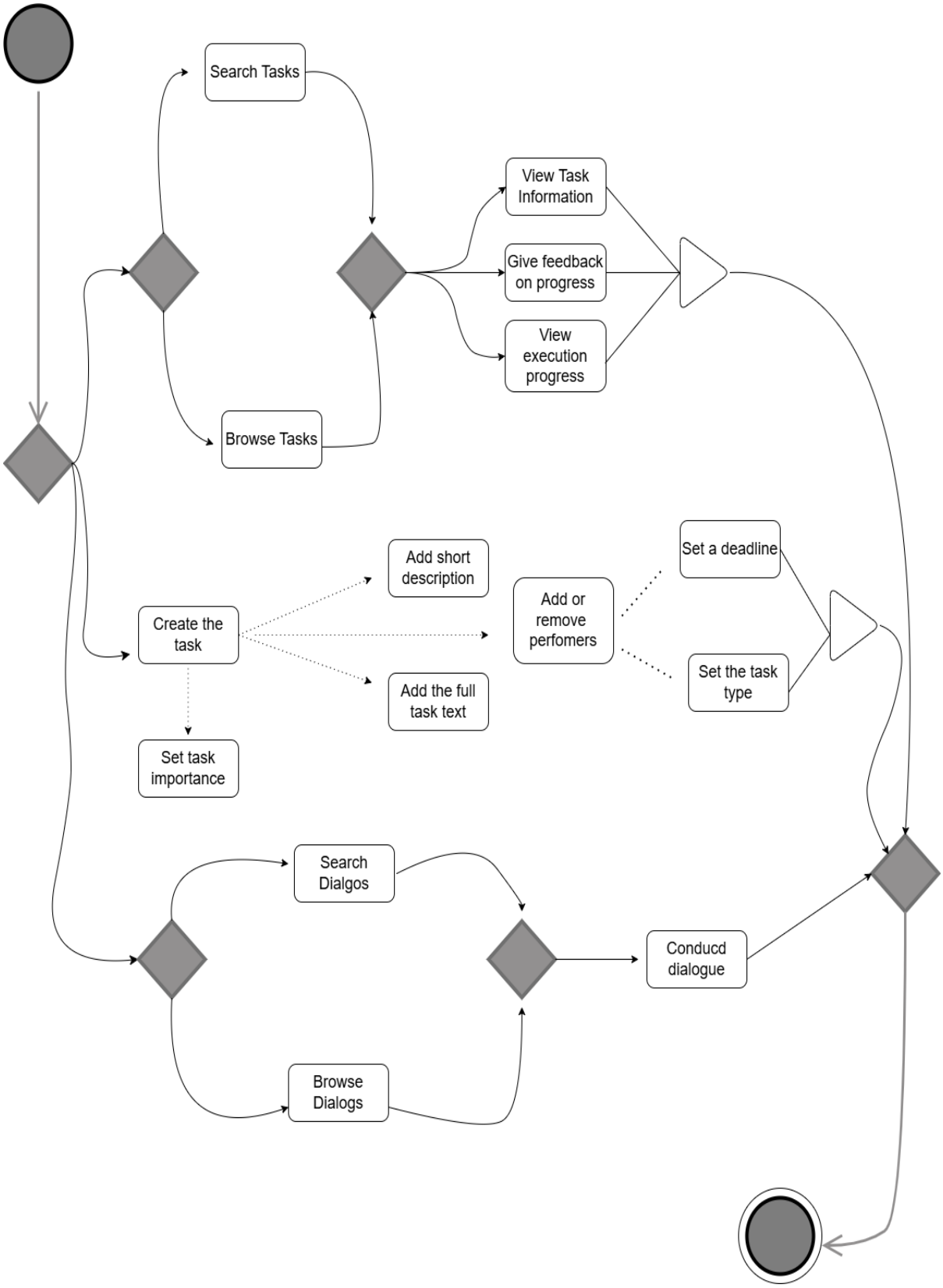

Рисунок Б.1 — Діаграма станів проекту

ДОДАТОК В

Маршрутизація сайту

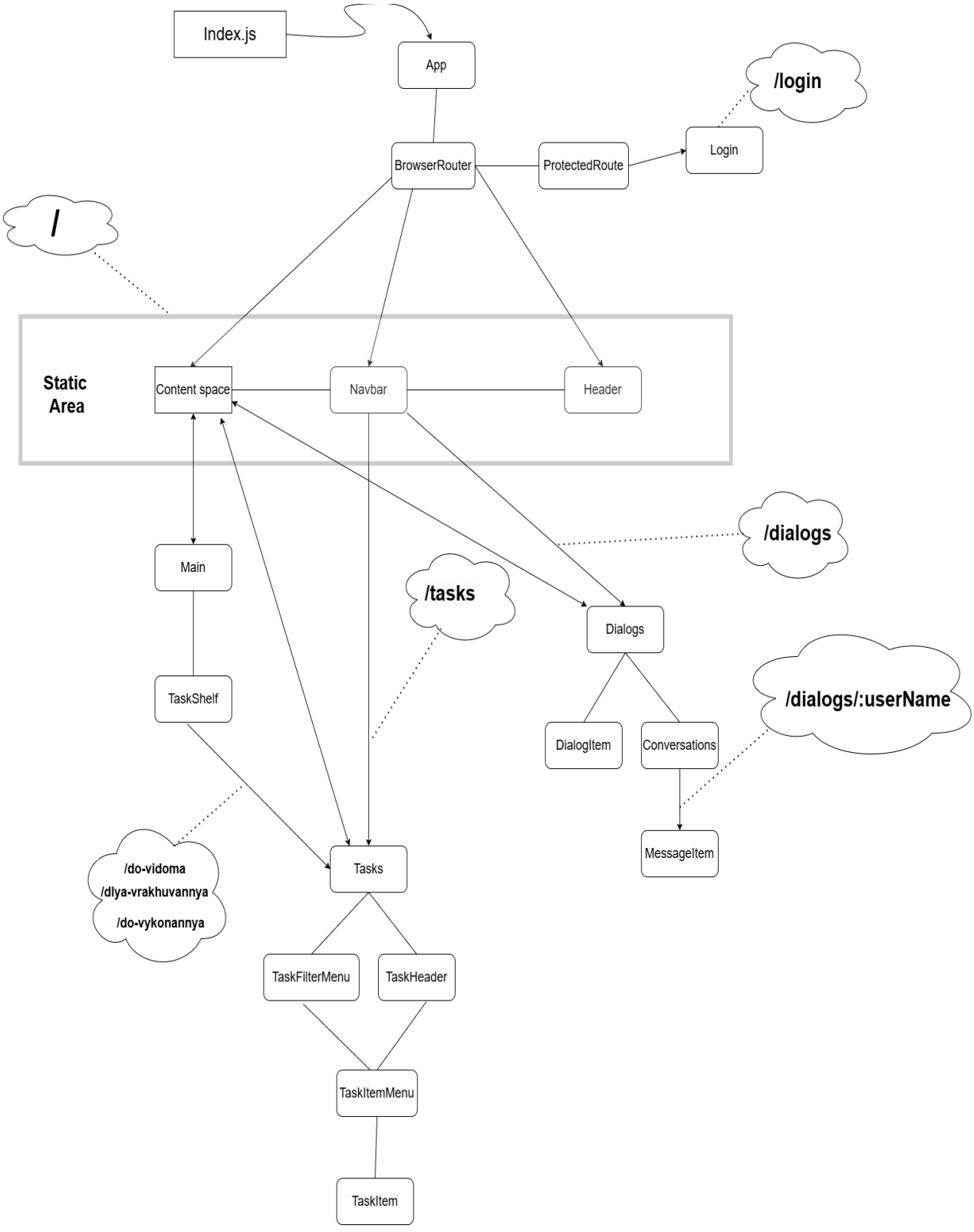

Рисунок В.1 — Маршрутизація сайту

ДОДАТОК Г

Лістинг LoginController.cs

```
using Microsoft.AspNetCore.Mvc;

[ApiController]
[Route("api/[controller]")]
public class AuthController : ControllerBase
{
    private readonly AuthService _authService;

    public AuthController(AuthService authService)
    {
        _authService = authService;
    }

    [HttpPost("login")]
    public async Task<IActionResult> Login([FromBody] LoginRequest request)
    {
        var token = await _authService.AuthenticateUserAsync(request.Login, request.Password);

        if (token == null)
        {
            return Unauthorized(new { message = "Invalid login or password" });
        }

        return Ok(new LoginResponse { Token = token });
    }
                                                                    }

public class LoginRequest
{
    public string Login { get; set; }
    public string Password { get; set; }
}

// LoginResponse.cs
public class LoginResponse
{
    public string Token { get; set; }
}


using Microsoft.AspNetCore.Identity.Data;
using Microsoft.AspNetCore.Mvc;

[ApiController]
[Route("api/[controller]")]
public class AuthController : ControllerBase
{
    private readonly AuthService _authService;

    public AuthController(AuthService authService)
    {
        _authService = authService;
    }

    [HttpPost("login")]
    public async Task<IActionResult> Login([FromBody] LoginRequest request)
    {
        var token = await _authService.AuthenticateUserAsync(request.Login, request.Password);
```

```
            if (token == null)
            {
                return Unauthorized(new { message = "Invalid login or password" });
            }

            return Ok(new LoginResponse { Token = token });
        }
                                                    }


    using Microsoft.IdentityModel.Tokens;
    using System.IdentityModel.Tokens.Jwt;
    using System.Security.Claims;
    using System.Text;

    public class AuthService
    {
        private readonly YourDbContext _context;
        private readonly string _secretKey = "aakeg";

        public AuthService(YourDbContext context)
        {
            _context = context;
        }

        // Метод для перевірки користувача і створення JWT
        public async Task<string?> AuthenticateUserAsync(string login, string password)
        {
            var user = await _context.Users
                .FirstOrDefaultAsync(u => u.Login == login && u.Password == password);

            if (user == null)
            {
                return null;
            }

            // Генерація JWT токену
            var tokenHandler = new JwtSecurityTokenHandler();
            var key = Encoding.UTF8.GetBytes(_secretKey);
            var tokenDescriptor = new SecurityTokenDescriptor
            {
                Subject = new ClaimsIdentity(new[]
                {
                    new Claim(ClaimTypes.Name, user.UserName),
                    new Claim("UserId", user.Id.ToString()),
                    new Claim("Department", user.Department),
                    new Claim("Mail", user.Mail),
                    new Claim("Phone", user.Phone)
                }),
                Expires = DateTime.UtcNow.AddHours(1),
                SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key),
SecurityAlgorithms.HmacSha256Signature)
            };

            var token = tokenHandler.CreateToken(tokenDescriptor);
            return tokenHandler.WriteToken(token);
        }
                                                        }
```

ДОДАТОК Д

Лістинги App.js

```
import './App.css';
import NavBar from './Components/NavBar/NavBar.jsx';
import Header from './Components/Header/Header.jsx';
import Main from './Components/Main/Main.jsx';
import Dialogs from './Components/Dialogs/Dialogs.jsx';
import Tasks from './Components/Tasks/Tasks.jsx';
import Conversations from './Components/Conversations/Conversations.jsx';
import { BrowserRouter, Route, Routes } from 'react-router-dom';
import { AuthProvider } from './AuthContext';
import ProtectedRoute from './ProtectedRoute';
import Login from './Components/Login/Login';
import { useState, useEffect } from 'react';

function App() {
  const [tasks, setTasks] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    // Завантаження даних із сервера
    const fetchData = async () => {
      try {
        const response = await fetch('http://localhost:5000/api/tasks', {
          headers: {
            'Authorization': `Bearer ${localStorage.getItem('token')}`,
          },
        });
        if (response.ok) {
          const data = await response.json();
          setTasks(data);
        }
      } catch (error) {
        console.error('Помилка завантаження даних:', error);
      } finally {
        setLoading(false);
      }
    };

    fetchData();
  }, []);

  if (loading) return <div>Завантаження...</div>;

  const tasksToExecute = tasks.filter(task => task.goal === 'До виконання');
  const tasksToConsider = tasks.filter(task => task.goal === 'Для врахування в роботі');
  const tasksForInformation = tasks.filter(task => task.goal === 'До відома');

  return (
    <AuthProvider>
      <BrowserRouter>
        <div className='app-wrapper'>
          <Header />
          <NavBar />
          <div className='app-wrapper-content'>
            <Routes>
              <Route path="/login" element={<Login />} />
              <Route path="/" element={<ProtectedRoute><Main /></ProtectedRoute>} />
              <Route path="/dialogs" element={<ProtectedRoute><Dialogs /></ProtectedRoute>}>
                <Route path=":username" element={<Conversations />} />
              </Route>
              <Route path="/tasks" element={<ProtectedRoute><Tasks data={tasks} /></ProtectedRoute>} />
```

```
                    <Route      path="/tasks/do-vykonannya"      element={<ProtectedRoute><Tasks      data={tasksToExecute}
/></ProtectedRoute>} />
                    <Route    path="/tasks/dlya-vrakhuvannya"    element={<ProtectedRoute><Tasks    data={tasksToConsider}
/></ProtectedRoute>} />
                    <Route      path="/tasks/do-vidoma"      element={<ProtectedRoute><Tasks      data={tasksForInformation}
/></ProtectedRoute>} />
                  </Routes>
                </div>
              </div>
            </BrowserRouter>
          </AuthProvider>
        );
      }

      export default App;
```

ДОДАТОК Е

Лістинг Header.jsx

```
using Microsoft.Extensions.Options;
using static System.Runtime.InteropServices.JavaScript.JSType;
using System.Reflection.PortableExecutable;
using System;

import React, { useState, useEffect } from 'react';
import s from './Header.module.css';
import { useNavigate } from 'react-router-dom';

const Header = () => {
const [currentTime, setCurrentTime] = useState(new Date());
const [isPanelOpen, setIsPanelOpen] = useState(false);
const [currentUser, setCurrentUser] = useState(null);
const navigate = useNavigate();

useEffect(() => {
   const intervalId = setInterval(() => {
      setCurrentTime(new Date());
   }, 60000);

   return () => clearInterval(intervalId);
}, []);

useEffect(() => {
const fetchUserData = async () => {
try
{
   // Отримайте токен з локального сховища
   const token = localStorage.getItem('jwtToken');

   // Запит до сервера для отримання даних користувача
   const response = await fetch('https://your-api-endpoint.com/currentUser', {
   method: 'GET',
        headers:
       {
          'Content-Type': 'application/json',
          // Додаємо токен в заголовок запиту
          'Authorization': `Bearer ${token}`
        }
       });

if (!response.ok)
{
   throw new Error('Помилка отримання даних користувача');
}

const data = await response.json();
setCurrentUser(data);
     } catch (error) {
   console.error('Помилка завантаження даних користувача:', error);
}
   };

fetchUserData();
  }, []);

const formatDate = (date) => {
const options = { day: 'numeric', month: 'long', year: 'numeric', hour: '2-digit', minute: '2-digit', hour12: false };
return date.toLocaleString('uk-UA', options);
  };
```

```
const togglePanel = () => {
   setIsPanelOpen(!isPanelOpen);
};

const handleLogout = () => {
   // Видалення токену з локального сховища при виході
   localStorage.removeItem('jwtToken');
   navigate('/login');
};
if (!currentUser)
{
   return < div > Завантаження...</ div >;
}

return (

  < div className = "header" >

    < div id ={ s.time}>

      < span >{ formatDate(currentTime)}</ span >

    </ div >

    < div id ={ s.userName}
onClick ={ togglePanel}>

      < span >{ currentUser.name}</ span >

    </ div >
      {
    isPanelOpen && (
       < div className ={ s.panel}>
         < ul >
           < li > Ім'я: {currentUser.name}</li>
           < li > Підрозділ: { currentUser.department}</ li >
           < li > Пошта: { currentUser.email}</ li >
           < li > Телефон: { currentUser.phone}</ li >
           < li >< button onClick ={ handleLogout}> Вийти </ button ></ li >
         </ ul >
       </ div >
     )}
    </ div >
  );
};

          export default Header;
```

ДОДАТОК Ж

Лістинг ModalWindow.jsx

```
using Microsoft.AspNetCore.DataProtection.KeyManagement;
using static System.Runtime.InteropServices.JavaScript.JSType;
using System.Reflection.Emit;
using System.Threading.Tasks;
using System.Threading;
using System.Xml.Linq;
using System;

import React, { useState } from 'react';
import s from './Main.module.css';

const ModalWindow = ({ closeModal }) => {
   const [taskIsVisible, setTaskIsVisible] = useState(true);
   const [implementationVisible, setImplementationVisible] = useState(false);
   const [activeIndex, setActiveIndex] = useState(null);
   const [tasks, setTasks] = useState([
   { name: '', status: 'До відома', deadline: '' }
  ]);

   // Стани для основної інформації завдання
   const [shortDescription, setShortDescription] = useState('');
   const [taskText, setTaskText] = useState('');
   const [priority, setPriority] = useState('high');

   const handleClick = (index) => {
      setActiveIndex(activeIndex === index ? null : index);
      if (index === 0)
      {
         setTaskIsVisible(true);
         setImplementationVisible(false);
      }
      if (index === 1)
      {
         setTaskIsVisible(false);
         setImplementationVisible(true);
      }
   };

   const handleAddTask = () => {
      setTasks([...tasks, { name: '', status: 'До відома', deadline: '' }]);
   };

   const handleChangeTask = (index, field, value) => {
      const updatedTasks = tasks.map((task, i) =>
        i === index ? { ...task, [field]: value } : task
      );
setTasks(updatedTasks);
  };

const handleDeleteTask = (index) => {
   const updatedTasks = tasks.filter((_, i) => i !== index);
   setTasks(updatedTasks);
};

// Обробник події для збереження завдання
const handleSaveTask = async () => {
// Зчитування даних із форми
const dataToSend = {
     shortDescription,
     taskText,
     priority,
```

```
      tasks: tasks.map(task => ({
name: task.name,
        status: task.status,
        deadline: task.deadline
      }))
    };

try
{
    // Відправка даних на сервер
    const response = await fetch('https://localm/api/tasks', {
    method: 'POST',
        headers:
        {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(dataToSend)
      });

    if (response.ok)
    {
        alert('Завдання успішно збережено!');
        closeModal();
    }
    else
    {
        alert('Сталася помилка при збереженні завдання.');
    }
}
catch (error)
{
    console.error('Помилка при відправці даних:', error);
}
  };

return (

  < div className ={ s.overlay}>

    < div className ={ s.createWindow}>

      < div className ={ s.modalOverlay}
onClick ={ closeModal}></ div >

      < div className ={ s.modalWindow}>

        < div className ={ s.taskNav}>

          < span
            className ={ activeIndex === 0 ? s.active : ''}
onClick ={ () => handleClick(0)}
          >
            Завдання
          </ span >
          < span
            className ={ activeIndex === 1 ? s.active : ''}
onClick ={ () => handleClick(1)}
          >
            Погодження
          </ span >
        </ div >
        {
    implementationVisible && (
          <>
            < table className ={ s.taskTable}>
              < thead >
```

```
            < tr >
              < th > Ім’я </ th >
              < th > Статус </ th >
              < th > Дедлайн </ th >
              < th > Дії </ th >
            </ tr >
          </ thead >
          < tbody >
           {
    tasks.map((task, index) => (
             < tr key ={ index}>
              < td >
               < input
                type = "text"
                placeholder = "Введіть ім’я"
                value ={ task.name}
    onChange ={
      (e) =>
     handleChangeTask(index, 'name', e.target.value)
                }
               />
              </ td >
              < td >
               < select
                value ={ task.status}
    onChange ={
      (e) =>
     handleChangeTask(index, 'status', e.target.value)
                }
              >
               < option value = "До відома" > До відома </ option >
               < option value = "Для врахування в роботі" >
                Для врахування в роботі
               </ option >
               < option value = "До виконання" > До виконання </ option >
              </ select >
             </ td >
             < td >
               < input
                type = "date"
                value ={ task.deadline}
    onChange ={
      (e) =>
     handleChangeTask(index, 'deadline', e.target.value)
                }
               />
             </ td >
             < td >
               < button onClick ={ () => handleDeleteTask(index)}>
                Видалити
               </ button >
             </ td >
            </ tr >
           ))}
          </ tbody >
        </ table >
        < button onClick ={ handleAddTask}> Додати користувача </ button >
       </>
      )}
{
  taskIsVisible && (

 <>

  < label > Короткий опис </ label >
```

```
    < input
            type = "text"
            id = "username"
            name = "username"
            placeholder = "Введіть короткий опис завдання"
            value ={ shortDescription}
    onChange ={ (e) => setShortDescription(e.target.value)}
          />
          < label > Текст завдання </ label >
          < textarea
            id = "message"
            name = "message"
            rows = "4"
            cols = "50"
            placeholder = "Введіть основний текст"
            style ={ { resize: 'none' } }
    value ={ taskText}
    onChange ={ (e) => setTaskText(e.target.value)}
          ></ textarea >
          < label > Важливість </ label >
          < select
            id = "priority"
            name = "priority"
            value ={ priority}
    onChange ={ (e) => setPriority(e.target.value)}
          >
            < option value = "high" > Низька </ option >
            < option value = "medium" > Середня </ option >
            < option value = "low" > Висока </ option >
          </ select >
          < button onClick ={ closeModal}> Закрити </ button >
          < button id ={ s.saveBtn}
    onClick ={ handleSaveTask}> Зберегти </ button >
        </>
      )}
     </ div >
   </ div >
  </ div >
 );
};

export default ModalWindow;
```

ДОДАТОК И

Лістинг Main.jsx

```
import React, { useState } from 'react';
import s from './Main.module.css';
import TaskShelf from './TaskShelf';
import ModalWindow from './ModalWindow';


const Main = () => {
   const [isModalVisible, setIsModalVisible] = useState(false);
   const [activeIndex, setActiveIndex] = useState(null);

   const handleClick = (index) => {
      setActiveIndex(activeIndex === index ? null : index);
   };



   const openModal = () => {
      setIsModalVisible(true);
   };

   const closeModal = () => {
      setIsModalVisible(false);
   };

   return (

     < div >

      < TaskShelf openModal ={ openModal}/>
    { isModalVisible && < ModalWindow closeModal ={ closeModal}/>}
   </ div >
  );
};

                                                export default Main;
```

ДОДАТОК К

Лістинг SaveTaskController.jsx

```
using Microsoft.AspNetCore.Mvc;

namespace WebApplication1.Controllers
{
   [Route("/saveTask")]
   [ApiController]
   public class SaveTasksController : ControllerBase
   {
      private readonly YourDbContext _context;

      public TasksController(YourDbContext context)
      {
         _context = context;
      }

      [HttpPost]
      public async Task<IActionResult> CreateTask([FromBody] TaskDto taskDto)
      {
         if (!ModelState.IsValid)
         {
            return BadRequest(ModelState);
         }

         try
         {
            var task = new Task
            {
               Name = taskDto.Name,
               Status = taskDto.Status,
               Deadline = taskDto.Deadline,
               Description = taskDto.Description,
               Priority = taskDto.Priority
            };

            _context.Tasks.Add(task);
            await _context.SaveChangesAsync();

                        foreach (var userId in taskDto.AssignedUserIds)
                        {
                           var taskAssignment = new TaskAssignment
                           {
                              TaskId = task.TaskId,
                              UserId = userId
                           };
                           _context.TaskAssignments.Add(taskAssignment);
                        }

                        await _context.SaveChangesAsync();

                        return Ok(task);
                        }
                        catch (Exception ex)
                        {
                           return StatusCode(500, $"Виникла помилка при збереженні завдання: {ex.Message}");
                        }
                        }
                        }

                        public class TaskDto
                        {
```

```
        public string Name { get; set; }
        public string Status { get; set; }
        public DateTime? Deadline { get; set; }
        public string Description { get; set; }
        public string Priority { get; set; }
        public List<int> AssignedUserIds { get; set; }

}
```

ДОДАТОК Л

Лістинги Navbar.jsx

```
import React from 'react'
import { NavLink } from 'react-router-dom';
import s from './Navbar.module.css';

const NavBar = () => {
   return (
      < div className = "nav" >
         < nav >
            < div className ={ s.item}>
               < NavLink to = "/" className ={ ({ isActive }) => (isActive ? s.activeLink : undefined)}> Головна </
NavLink >
            </ div >
            < div className ={ s.item}>
               < NavLink to = "/dialogs" className ={ ({ isActive }) => (isActive ? s.activeLink : undefined)}>
Повідомлення </ NavLink >
            </ div >
            < div className ={ s.item}>
               < NavLink to = "/tasks" className ={ ({ isActive }) => (isActive ? s.activeLink : undefined)}> Завдання
</ NavLink >
            </ div >
         </ nav >
      </ div >
   );
}

export default NavBar;
```

ДОДАТОК М

Лістинги UserController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Linq;
using System.Threading.Tasks;
using System.Security.Claims;

namespace WebApplication1.Controllers
{
   [ApiController]
   [Route("[controller]")]
   public class UserController : ControllerBase
   {
      private readonly ApplicationDbContext _context;

      public UserController(ApplicationDbContext context)
      {
         _context = context;
      }

      [HttpGet("currentUser")]
      [Authorize]
      public async Task<IActionResult> GetCurrentUser()
      {
         var userId = User.Claims.FirstOrDefault(c => c.Type == ClaimTypes.NameIdentifier)?.Value;

         if (string.IsNullOrEmpty(userId))
         {
            return Unauthorized("Не вдалося знайти користувача.");
         }

         return Ok(userId);
   }
                  var user = await _context.Users.FindAsync(int.Parse(userId));

                  if (user == null)
                  {
                     return NotFound("Користувач не знайдений.");
                  }

                  return Ok(new
                  {
                     UserId = user.UserId,
                     Name = user.UserName,
                     Email = user.Email,
                     Department = user.Department,
                     Phone = user.Phone
                  });
```

ДОДАТОК Н

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Інтегрована комп'ютерна система для оперативної взаємодії персоналу виробничого підприємства з виготовлення меблів

Тип роботи: Магістерська кваліфікаційна робота

(кваліфікаційна робота, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ кафедра обчислювальної техніки

(кафедра, факультет (інститут), навчальна група)

Керівник Савицька Л.А., доц. каф. ОТ

(прізвище, ініціали, посада)

Показники звіту подібності

Turnitin	
Оригінальність	98%
Загальна схожість	2%

Аналіз звіту подібності (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ _____ Палько А.С._____

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

_____

_____

_____

_____

_____

Особа, відповідальна за перевірку_____ _Савицька Л.А.__

(підпис) (прізвище, ініціали)

Експерт _____ _____

(за потреби) (підпис) (прізвище, ініціали, посада)