

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
КЕРУВАННЯ СПІЛКУВАННЯМ КОРИСТУВАЧІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
MANAGING USER COMMUNICATION

Виконав: студент(ка) 2 курсу, групи КНТ-119м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Лобода Р.В.

(прізвище та ініціали)

Керівник Гофман Є.О.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “_____” _____ 2020 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Лободи Романа Віталійовича
 (прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів
керування спілкуванням користувачів
Research and Software Implementation of Methods for Managing User
Communication

керівник проєкту (роботи) Гофман Євгеній Олександрович, к.т.н.
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “3” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) 10 грудня 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження.
2. Матеріали і методи. 3. Опис програми. 4. Розробка програми. 5. Експлуатація
та тестування програми. 6. Економіко-організаційна частина, 7. Охорона праці
та безпека в надзвичайних ситуаціях.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Гофман Є.О., доцент		
Економіко-організаційна частина	Пожуєва Т.О., професор		
Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Калініна М.В., асистент		

7. Дата видачі завдання “ 7 ” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка методу.	4–5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7–8 тижні	Розділ 4
6	Тестування та відлагодження програмного забезпечення.	9 тиждень	Розділ 5
7	Розробка економіко-організаційної частини.	10 тиждень	Розділ 6
8	Розробка заходів з охорони праці та безпеки в надзвичайних ситуаціях.	11 тиждень	Розділ 7
9	Оформлення пояснювальної записки та документів до неї.	12-13 тиждень	Додатки
10	Нормоконтроль та рецензування.	14–15 тижні	
11	Захист роботи.	16 тиждень	

Студент(ка)

(підпис)

Лобода Р.В.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Гофман Є.О.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
116 с., 28 рис., 10 табл., 3 дод., 37 джерел.

КОМУНІКАЦІЯ КОРИСТУВАЧІВ, ФІЛЬТРУВАННЯ
ПОВІДОМЛЕНЬ, МЕТОД ОПОРНИХ ВЕКТОРІВ, ІНТЕЛЕКТУАЛЬНИЙ
АНАЛІЗ ДАНИХ, JAVA, MYSQL.

Об'єкт дослідження – процес керування спілкуванням користувачів.
Предмет дослідження – програмне забезпечення корпоративної комунікації користувачів.

Мета роботи – розроблення програмного забезпечення керування спілкуванням користувачів для забезпечення дисциплінування спілкування працівників у робочий час.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування Java, система керування базами даних MySQL, метод опорних векторів, TF-IDF.

Результати. Проаналізовано методи інтелектуального аналізу даних, які можуть застосовуватися для керування спілкуванням користувачів, розроблено власний метод, виконано проєктування та реалізацію програмного забезпечення, описано прийняті рішення.

Висновки. Розроблено та програмно реалізовано метод керування спілкуванням користувачів, який базується на застосуванні фільтрації повідомлень шляхом визначеного набору слів, словосполучень та конструкцій речень та автоматичному аналізу цензурованих повідомлень, обчисленні статистики кожного користувача в кожен період часу, винесенні рекомендованого рішення щодо користувача на основі застосування методу опорних векторів для класифікації користувачів в поточний момент.

Галузь використання – робоча комунікація працівників підприємства.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 116 pages, 28 figures, 10 tables, 3 appendixes, 37 sources.

USER COMMUNICATION, MESSAGE FILTERING, SUPPORT VECTOR MACHINE, DATA MINING, JAVA, MYSQL.

The object of the research is the process of managing user communication. The subject of the research is the software for corporate communication of users.

The purpose of the work is to develop user communication management software to ensure the discipline of employee communication during working hours.

Materials, methods and technical means: object-oriented programming, Java programming language, MySQL database management system, support vector machines, TF-IDF.

Results. Methods of data mining that could be used to manage user communication are analyzed. The own method is developed, software design and implementation is performed, the adopted decisions are described.

Conclusions. Developed and software-implemented method of user communication management, which is based on the application of message filtering by a defined set of words, phrases and sentence constructions and automatic analysis of censored messages, calculation of each user's statistics in each time period, making a recommended user decision based on reference vectors. to classify users at the moment.

Field of usage – working communication of employees of the enterprise.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Правила використання корпоративних програмних засобів для комунікації користувачів	10
1.2 Огляд корпоративних програмних засобів комунікації користувачів ..	14
1.3 Висновки за розділом 1	19
2 Матеріали і методи.....	21
2.1 Методи інтелектуального аналізу даних для керування спілкуванням користувачів.....	21
2.2 Метод опорних векторів.....	24
2.3 Метод керування спілкуванням користувачів	26
2.4 Висновки за розділом 2	29
3 Опис програми.....	30
3.1 Визначення вимог до програмного забезпечення.....	30
3.2 Вибір мови програмування	33
3.3 Проєктування бази даних	35
3.4 Висновки за розділом 3	39
4 Розробка програми	41
4.1 Опис реалізації структури програмного забезпечення	41
4.2 Опис розроблених модулів.....	43
4.3 Висновки за розділом 4	46
5 Експлуатація та тестування програми	47
5.1 Призначення програми	47
5.2 Умови виконання програми	47
5.3 Виконання програми.....	47
6 Економіко-організаційна частина.....	56
6.1 Планування розробки програмного виробу	56

6.2 Визначення витрат на розробку програми	59
6.3 Оцінка економічної ефективності	65
7 Охорона праці та безпека в надзвичайних ситуаціях	67
7.1 Аналіз потенційних небезпек.....	67
7.2 Заходи з забезпечення безпеки	67
7.3 Заходи з виробничої санітарії і гігієни праці	68
7.4 Заходи з пожежної безпеки	76
7.5 Заходи з забезпечення безпеки у надзвичайних ситуаціях.....	77
Висновки	78
Перелік джерел посилання	79
Додаток А Технічне завдання	83
Додаток Б Текст програми	88
Додаток В Слайди презентації.....	109

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ЕОМ – електронно-обчислювальна машина;

КТ – комп'ютерна техніка;

ОЦ – обчислювальний центр;

ПК – персональний комп'ютер.

ВСТУП

Корпоративний месенджер – це застосунок для обміну повідомленнями, створення групових чатів, інформаційних каналів, а також вирішення багатьох інших робочих завдань, він може мати десктопну, мобільну і вебверсію, важливими особливостями такого застосунка є високий рівень захищеності і можливість інтеграції з бізнес-сервісами компанії, корпоративний месенджер може працювати в локальній мережі і розміщуватися на власних серверах організації, це дозволяє захистити застосунок від зломів, і запобігти витоку інформації або її скачування недобросовісними співробітниками, впровадження даного рішення допоможе спростити комунікацію між працівниками різних відділів і філій компанії, узгодження завдань і прийняття рішень в застосунку відбувається за допомогою миттєвих текстових повідомлень, весь персонал організації отримує одночасний доступ до ключової інформації, за рахунок чого робочі питання вирішуються швидше, ніж по електронній пошті [1].

Месенджери стали частиною корпоративної культури, зрослися з особистим спілкуванням: коли ведеться звичайний діалог на загальні теми, ні працівники, ні керівники не сприймають чат як офіційну зону спілкування, про це згадують лише при конфлікті, коли починається розгляд конфлікту, але фактично, коли співробітники об'єднуються для спілкування в чаті, такий чат вже служить інтересам компанії і є офіційним каналом [2].

Тож організація спілкування користувачів, по-перше, вимагає створення спеціалізованого застосунку, який компанія зможе використовувати, розгортати і підтримувати самостійно, а також вимагає керування таким спілкуванням, направленим на фільтрацію повідомлень та автоматизоване прийняття рішень щодо отриманих результатів.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Правила використання корпоративних програмних засобів для комунікації користувачів

В сучасних умовах компанії намагаються максимально повно використовувати всі доступні комунікаційні можливості, які дозволяють прискорити бізнес-процеси і поліпшити обмін інформацією між співробітниками: традиційні телефони, електронна пошта та навіть месенджери для різних компаній в силу цілого ряду причин перестають задовольняти сучасним потребам, тому все більше і більше компаній приходять до усвідомлення необхідності створення власного корпоративного чату, особливо це актуально для організацій, що мають мережу віддалених філій або підрозділів, до причин зокрема належать:

- відсутність засобів управління правами доступу;
- неконтрольованість глобальних систем передачі повідомлень;
- немає можливості встановити фільтри спілкування;
- не можна вплинути на обслуговування системи: корпоративна система повинна працювати стабільно, а ІТ-фахівці повинні мати можливість керувати нею, що неможливо в сучасних глобальних месенджерах;
- потенційна загроза для інформаційної безпеки компанії: сучасні месенджери є одним з каналів витоку конфіденційних даних (причому складно контрольованим каналом);
- фрагментація месенджерів серед співробітників компанії;
- неможливість для користувача відділити приватні розмови від робочої комунікації [3].

Тому заважаючи на перелічені причини, варто використовувати спосіб організації корпоративного чату на основі спеціального програмного забезпечення, що розгортається безпосередньо в локальній мережі організації, таке рішення має технологічні та організаційні переваги, прямо протилежні недолікам глобальних месенджерів: вони володіють всіма

необхідними можливостями, повністю контрольовані і не можуть стати причиною витоку конфіденційної інформації, але при цьому і у них є деякі мінуси – додаткові витрати на придбання, розгортання і обслуговування системи [3].

Під час комунікації важливо не забувати про правила, дотримання яких зокрема є важливим у корпоративному месенджері:

- тільки в робочий час: не слід плутати відправку поштового повідомлення з відправкою повідомлення в месенджері, на повідомлення, що приходять в месенджер, звертають більше уваги, а сам формат комунікації зобов'язує одержувача не затягувати з відповіддю, отримувати ділові повідомлення пізно ввечері або рано вранці – дивно і некомфортно, крім того, написавши користувачеві вночі, є ризик його розбудити – багато людей тримають телефон поруч і не відключають повідомлення перед сном [3];

- обсяг повідомлень: месенджери в принципі не призначені для відправки великих текстових повідомлень, враховуйте, що швидше за все користувач буде читати повідомлення з екрану смартфона, приберіть все зайве – тут не потрібні довгі вітання та детальні описи, тільки найголовніше, для пересилання великих текстів існує електронна пошта [4];

- одне запитання – одне повідомлення: не слід відправляти повідомлення виду «Добрий день, Микола Іванович!» і чекати відповіді, також не потрібно дробити послання на декілька коротких повідомлень – по-перше, це відволікає і дратує, по-друге, заважає сприймати текст цілісно [5];

- конкретика і лаконічність: чим менше тексту, тим краще, намагайтеся висловлюватися максимально ясно і коротко, це безпрограшна комбінація – співрозмовник не тільки зрозуміє, що саме ви від нього хочете, але і оцінить ваш лаконічний діловий стиль, якщо необхідно поставити декілька запитань або позначити ряд тез, структуруйте повідомлення, писати суцільну стіну тексту, яка містить декілька тем для обговорення, – не найкраща ідея [5];

– знак питання – найпотужніший знак пунктуації, різко збільшує шанси на отримання відповіді: найчастіше люди викладають свої думки, не позначаючи те, що саме вони хочуть отримати від адресата, у цьому випадку людина може не зрозуміти, чи потрібна від неї будь-яка реакція, або це просто інформація до відома [4];

– емодзі і стікери: наскільки допустимо використовувати смайли в діловому листуванні – питання спірне, однозначно можна сказати, що починати ділове спілкування з відправки стікерів або смайлів не варто, в іншому, все залежить від ставлення співрозмовника до такого способу спілкування, близькості знайомства і корпоративних стандартів, якщо мова йде про груповий чат, якщо в контексті конкретної бесіди використання смайлів і стікерів допустимо, то можна відправляти їх в ті моменти, коли потрібно зняти напругу і додати в спілкування трохи емоцій [3];

– стиль повідомлень: він повинен залишатися діловим, не варто писати незнайомій людині текст з використанням слів на кшталт «лол», «плиз», «ASAP» і т.д., у кращому випадку вона може їх не зрозуміти, в гіршому – розцінити це як фамільярність і невихованість, більш вільно спілкуватися можна буде, коли краще дізнаєтеся співрозмовника і його ставлення до сленгу, універсальне правило – наслідуйте стиль співрозмовника, це найпростіший спосіб швидко знайти спільну мову і повернути до себе людину, не важливо, кому саме ви пишете – начальнику, підлеглому, колезі або партнеру, наслідування стилю все спрощує: наприклад, якщо людина вважає за краще відправляти всю інформацію в одному повідомленні, то його будуть дратувати люди, які відправляють декілька коротких повідомлень за раз [2];

– грамотність і помилки: в рамках дружнього листування в месенджерах ми часто не надаємо значення допущеним помилкам, друкарським помилкам, у діловому спілкуванні помилки швидко зіпсують враження про вас, перевіряйте повідомлення перед відправкою, якщо було

відправлено текст з помилкою – відредагуйте повідомлення, і стежте за автокорекцією – вона може поставити вас у незручне становище [5];

– ніяких серйозних рішень: можна комфортно обговорювати угоду або проєкт в месенджері, однак для закріплення домовленостей цей канал зв'язку не підходить, в рамках ділового етикету, повідомлення з месенджера мають меншу вагу, ніж офіційний електронний лист, для прийняття остаточного рішення з якого-небудь питання використовуйте електронну пошту, крім усього іншого, лист нікуди не зникне і його буде легко знайти, чого не скажеш про повідомленні в месенджері – домовленості губляться в довгому листуванні, а сам месенджер може бути заблокований в будь-який час [4];

– фотографія в профілі: переконайтеся, що вона прийнятна для ділового листування, звичайно, месенджер – це менш офіційний канал зв'язку, ніж та ж пошта, і є велика свобода для самовираження, однак не варто забувати про те, що співрозмовника може відштовхнути недоречна фотографія в профілі [5];

– конкретні відповіді: якщо вам поставили запитання або поставили завдання – відповідайте розгорнуто («ок» або «зрозуміло» – відповіді не завжди доречні), намагайтеся відповідати конкретно («зроблю завтра до кінця робочого дня», «вивчу матеріали і дам відповідь в понеділок» і т.д.) [5];

– цитування: цитуйте співрозмовника, якщо в його повідомленні міститься декілька запитань або тез, структурована відповідь також важлива, як і структуроване запитання, переписка співробітників в груповому чаті – дуже ефективний інструмент комунікації, завдяки їй члени команди, які працюють над одним проєктом, зближуються і починають спілкуватися більш вільно і залучено, це підвищує продуктивність їх роботи, крім того, спілкуючись в груповому чаті, співробітники можуть максимально швидко вирішувати виникаючі питання або проблеми, до листування в корпоративному чаті застосовні всі описані вище правила, варто додати, що в загальний чат не можна відправляти конфіденційні документи або

інформацію, що не призначається для деяких учасників чату, також в груповому листуванні недоречно обговорювати особисті питання, робити все те, що погано впливає на робочий настрій учасників [4]-[5].

1.2 Огляд корпоративних програмних засобів комунікації користувачів

Slack (рис. 1.1) – зручний і гнучкий месенджер, з простим і інтуїтивно зрозумілим інтерфейсом і широкою функціональністю, призначений саме для корпоративної взаємодії, хоча і є глобальним рішенням, до переваг можна віднести простоту установки: досить завантажити застосунок або відкрити в будь-якому браузері сторінку Slack, зареєструватися і почати роботу, Slack можна використовувати навіть безкоштовно, причому як завгодно довго, хоча функціональність безкоштовної версії обмежена [6]-[7].

Slack – це не просто месенджер, а справжній робочий інструмент, який дозволяє спростити спілкування всередині команди, розподілити і диверсифікувати його по відділах [8].

До основних особливостей Slack, що відрізняють сервіс від інших месенджерів, належать:

- для кожного відділу (або певної теми) можна створити окремий канал і запросити в нього скільки завгодно людей;
- Slack можна синхронізувати з чималою кількістю інших сервісів, необхідних для роботи, зокрема з Google Drive, Google Docs, Twitter, Trello, Google Hangouts, DropBox і іншими;
- функції аудіо- та відеодзвінків і конференцій;
- можливість проводити опитування і голосування;
- мобільні версії для iOS і Android;
- можливість приватних чатів для двох співробітників і цілої групи;
- функція створення завдань, управління ними та відстеження готовності [7].

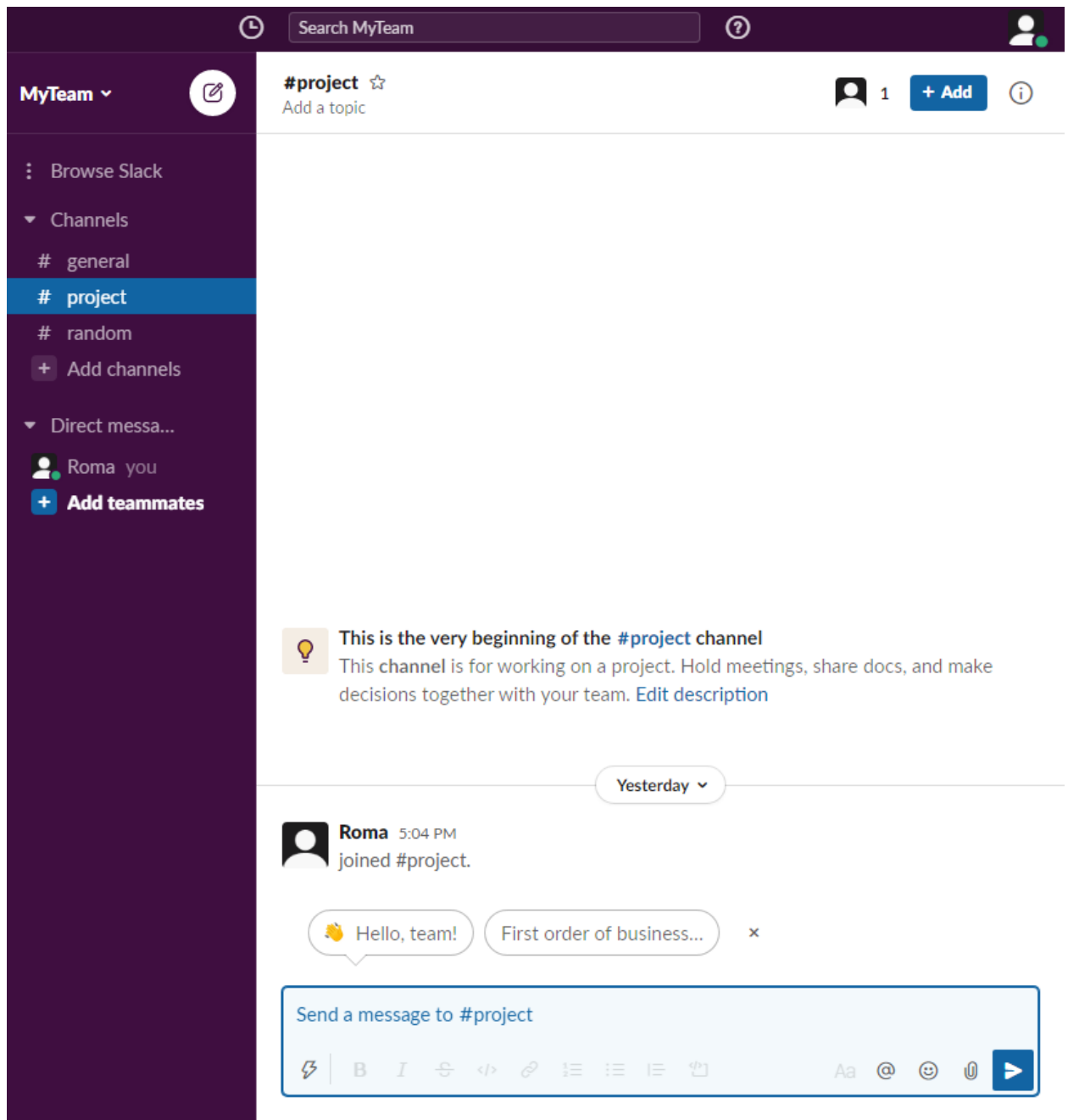


Рисунок 1.1 – Корпоративний месенджер Slack

Окрім того до особливостей Slack належать:

- робочі календарі з можливістю внесення подій: користувачі можуть переглядати календарі інших каналів і ділитися подіями;
- широкі можливості для налаштування особистого статусу: користувач може дати знати колегам, коли перебуває у відпустці або працює віддалено, заблокувати і розблокувати повідомлення на певний термін;
- додавання і збереження посилань і файлів;

- можливість згадувати окремих користувачів в групових чатах – з цією функцією шанси, що людина пропустить важливе повідомлення, адресоване їй, значно знижуються;

- можливість демонструвати іншим користувачам робочий екран в режимі реального часу під час відеодзвінків і конференцій;

- функція пошуку серед повідомлень;

- статистика із користування послугою [7].

Прикладом повністю корпоративного застосунку, який може використовуватися в розподіленій, але локальній системі і контролюватися локально, є зокрема застосунок MyChat.

MyChat відзначається наступним:

- можливі як приватні переговори користувачів один з одним, так і спілкування в форматі конференцій: при цьому набір учасників може бути вільним (до бесіди може приєднатися будь-хто) або обмеженим (брати участь можуть тільки строго визначені користувачі), такий підхід дозволяє використовувати корпоративний чат практично для будь-якого спілкування співробітників між собою, окремо можна відзначити, що система може вести детальну історію всіх повідомлень, а доступ можуть отримати не тільки учасники, а й користувачі, яким адміністратор надав на це право, ця функція може використовуватися, наприклад, для контролю керівниками переговорів своїх співробітників, в ході спілкування користувачі можуть застосовувати різні можливості, до них відноситься вставка в повідомлення смайликів, картинок і заздалегідь підготовлених фраз, пряма передача один одному файлів, перевірка правопису, можливість просигналізувати співрозмовнику звуком (привернути увагу до комп'ютера);

- в MyChat реалізовано відразу дві книги контактів: одна з них спільна, у ній перераховані всі зареєстровані користувачі, розбиті по підрозділах із зазначенням керівників (природно, якщо підрозділи і керівники були вказані під час налаштування системи), друга книга –

особиста, її складом повністю керують самі користувачі, вносячи потрібні їм контакти;

– в даному продукті передбачено декілька способів фільтрації, покликаних зробити спілкування в чаті більш чистим: перший з них налаштований на виявлення лайки (використовується для запобігання появи нецензурних або взагалі будь-яких небажаних слів, готовий набір фраз вже входить в комплект поставки, крім того, він може доповнюватися і змінюватися адміністратором), другий фільтр – антифлуд (дозволяє створити набір правил, при яких користувач буде вважатися порушником правил чату та тимчасово блокуватися, в налаштуваннях можна вказати максимальну кількість повідомлень за вказаний проміжок часу і граничну кількість смайликів і однакових слів в одному повідомленні);

– в MyChat передбачений також персональний фільтр – так званий ігнор-список: кожен користувач веде його самостійно, заносячи в нього співробітників, взаємодію з якими він хоче обмежити: ігнор-список – дуже гнучкий інструмент, з його допомогою можна заборонити окремо отримання звукових сигналів від певного користувача, особистих повідомлень, приватів, повідомлень в каналах і іншого;

– додатковим засобом спілкування є так звані оповіщення, які можуть відправлятися як одиночним користувачам, так і цілим групам відразу, вони відрізняються від чату тим, що відправляються без відкриття вікна спілкування, даний спосіб дуже зручний для відправки завдань, доручень, повідомлень та іншого: наприклад, можна попросити бухгалтера провести документ із зазначеним номером, юриста підготувати договір на певну годину, зібрати потрібних співробітників на нараду, при бажанні відправник може вказати необхідність підтвердження отримання повідомлення, у цьому випадку воно відкривається поверх всіх вікон і не закривається, поки абонент не підтвердить його прочитання, це гарантує відсутність спірних ситуацій, коли співробітник стверджує, що не отримував і не бачив повідомлення;

– ще однією додатковою функцією є дошка оголошень: вона дозволяє створювати повідомлення, які показуються всім користувачам чату протягом зазначеного періоду часу, цей інструмент може використовуватися для оповіщення співробітників компанії про будь-які події, наприклад, про майбутні корпоративи, необхідності пройти медогляд;

– вбудований FTP-сервер: можна використовувати вбудований в MyChat інструмент, що дозволяє створювати спільні та особисті сховища і не вимагає додаткового адміністрування (всі користувачі отримують облікові записи для доступу до FTP-сервера), крім того, вбудований FTP-сервер використовується для організації автоматичного оновлення клієнтів чату, а також для автоматичного розподілу навантаження при вставці великих зображень в текст повідомлень, особливо в каналах, де можуть знаходитися сотні користувачів;

– система управління сервером: змінювати параметри роботи можна за допомогою спеціального застосунку, що запускається локально, однак найчастіше для адміністраторів набагато зручніше управляти сервером віддалено, для цього в MyChat реалізовано відразу дві функції: перша з них – спеціальний вебсервер, що забезпечує доступ до адміністративних можливостей за допомогою звичайного браузера, функціональність вебклієнта аналогічна функціональності програми для управління сервером, друга можливість віддаленого управління – спеціальна консоль, вона відкривається безпосередньо в клієнті чату і дозволяє управляти сервером за допомогою набору текстових команд, можливості цієї системи дещо обмежені, команди в основному розраховані на отримання інформації про сервер і на управління його поточною роботою, так, наприклад, з їх допомогою можна перезавантажити застосунок, отримати дані про зареєстровані канали, відключити користувача, перевірити з'єднання з клієнтом тощо;

– всі повідомлення, передані в рамках чату, шифруються, – це важливо, оскільки дозволяє не боятися перехоплення трафіку при передачі

його через незахищені канали зв'язку, крім того в системі можна дуже гнучко налаштовувати права користувачів для кожної функції окремо, в тому числі є можливість заборонити вивантажувати програму-клієнта з пам'яті комп'ютера, змінювати її налаштування, причому всі зміни в права доступу вносяться «на льоту» і не вимагають перезавантаження сервера або клієнтів [9].

1.3 Висновки за розділом 1

Розглянуто проблеми сучасних засобів комунікації відносно корпоративної сфери, правила поведінки у корпоративному месенджері.

Виконано огляд програмних засобів, які можуть використовуватися для підтримки комунікації користувачів у корпоративній мережі, зокрема Slack та MyChat. Slack є глобальним рішенням, призначеним для корпоративної мережі, а MyChat локальним.

В подальшому для забезпечення виконання цілі роботи потрібно реалізувати виконання наступних завдань.

Дослідження методів керування спілкуванням користувачів. Необхідно дослідити в роботі, які засоби можуть використовуватися для реалізації фільтрування повідомлень і впровадити актуальні механізми інтелектуального аналізу даних шляхом створення власного відповідного методу.

Проектування структури програмного забезпечення. Необхідно визначити структуру програмного забезпечення, обрати мову для реалізації, визначити структуру збереження даних та виділити функціональні вимоги до програми.

Розробка програмного забезпечення. Необхідно реалізувати відповідні класи для практичного впровадження в результаті розробленого методу у вигляді програмного застосунку.

Тестування та відлагодження програми. Необхідно впровадити на практиці розроблений застосунок.

Економічне обґрунтування доцільності розробки та забезпечення заходів охорони праці та безпеки в надзвичайних ситуаціях.

2 МАТЕРІАЛИ І МЕТОДИ

Керування спілкуванням користувачів базується на виконанні фільтрації цього спілкування. З одного боку фільтрація дозволяє керувати повідомленнями в тому сенсі, щоб вилючити з них ту інформацію, яку вони не повинні містити: зокрема образи, погані слова, дані, які не мають розповсюджуватися на підприємстві (наприклад, інформація про зарплату працівників). З іншого боку інформація про виявлені випадки фільтрації є основою керування спілкуванням в подальшому, адже дозволяє виділяти користувачів, для яких потрібно вводити обмеження, а також тих, на спілкування яких потрібно звернути найбільшу увагу.

2.1 Методи інтелектуального аналізу даних для керування спілкуванням користувачів

Алгоритми попередньої обробки тексту базуються на тому, що крім базових підходів з нормалізацією форм слів і видалення шумових слів, для роботи деяких алгоритмів можуть запитати текст або слова в числовому вигляді – у вигляді вектора або послідовності векторів, для цього можуть використовуватися такі алгоритми:

- one-hot encoding – найпростіший алгоритм, що переводить слова тексту в вектора довжиною N з ненульовим значенням в i -й позиції, де N – розмір словника мови, а i – номер слова в цьому словнику;

- word2vec – група алгоритмів, яка також переводить слово в вектор, але підсумковий вектор має набагато меншу розмірність, ніж в разі one-hot encoding, і містить дійсні числа, при цьому слова, близькі за змістом, матимуть вектора, близькі один до одного;

- bag of words – даний алгоритм будує вектор частот, з якими слова зустрічаються в тексті, це дозволяє отримати вектор, що описує текст в цілому, але майже повністю ігнорується інформація про структуру тексту;

– TF-IDF – для кожного тексту будує вектор, кожен елемент якого визначається як відношення частоти слова в тексті до зворотної частоти слова в усіх документах, міра зворотної частоти може визначатися різними способами, але її основна ідея наступна: якщо слово широко вживається в мові, то воно часто зустрічається в документах і навряд чи має цінність для їх класифікації, тому такі слова матимуть меншу зворотну частоту, рідкісні слова, навпаки, будуть мати велику зворотну частоту [10].

Алгоритми пошуку іменованих сутностей і зв'язків між ними дозволяють розв'язувати завдання, яке полягає в пошуку конкретних термінів (імена компаній, продуктів, послуг і т.д.) і призначенні їм тегів відповідної категорії, а також зв'язків між ними, прикладами алгоритмів і моделей, що використовуються для цих цілей, є:

– Hidden Markov Model являє статистичну модель у вигляді спрямованого графа;

– conditional random field – як і Hidden Markov Model, є статистичною моделлю, але використовує ненаправлений граф;

– нейронні мережі, зокрема довга короткочасна пам'ять і згорткові нейронні мережі: довга короткочасна пам'ять враховує контекст за рахунок того, що вона є рекурентною і зберігає інформацію про контекст в процесі роботи, згорткові нейронні мережі отримують дані на основі патернів, які виявлятимуть суттєві ознаки;

– загальні методи машинного навчання: при використанні змінного вікна стає можливим використання загальних методів, таких як logistic regression, support vector machines (методів опорних векторів), naïve bayes, decision tree і інші, які вимагають на вхід фіксований набір ознак [10].

Алгоритми класифікації і тематичного моделювання документів мають на меті короткий опис оброблених документів (наприклад, визначення типу документа або тем документів), для цього можна використовувати:

– Latent Dirichlet Allocation – статистична модель, заснована на розподілі Дирихле, розглядає текст як суміш тематик;

– Latent Semantic Analysis – метод, заснований на поданні корпусу у вигляді терм-документної матриці і її розкладанні з метою визначення тем документів;

– Additive Regularization of Topic Models – даний метод використовує існуючі статистичні моделі, але додає до них регуляризацію, щоб краще враховувати структуру тесту;

– загальні методи машинного навчання використовують в якості вхідних даних результати роботи Bag of words, TF-IDF або інших алгоритмів [10].

Усі описані методи можуть використовуватися на різних етапах фільтрування тексту, зокрема фільтрування текстових повідомлень користувачів.

При фільтруванні на основі вмісту передбачається, що кожен користувач працює самостійно:

– як результат, представлення документів у системах фільтрації на основі вмісту можуть використовувати лише інформацію, яка може бути отримана із вмісту документа, зокрема однією з таких систем є система фільтрації тексту на основі вмісту для статей в Інтернеті, що має назву SIFT: профілі для SIFT були побудовані вручну, вказавши слова, яким слід віддати перевагу або уникати, і їх потрібно було оновлювати вручну, якщо користувач бажав їх змінити;

– для кожного профілю щодня було доступно двадцять статей у ранжированому вихідному форматі, статті можна було обирати інтерактивно за допомогою веббраузера;

– SIFT запропонував два засоби для допомоги користувачам у побудові профілю: спочатку користувачам була запропонована можливість застосовувати профілі кандидатів відносно виділених статей, щоб визначити, чи приймаються, чи відхиляються відповідні набори статей, ітеративне вдосконалення дозволило користувачеві створити профіль, який би перемістив відповідні статті у верхню частину списку, щоб полегшити

підтримку профілів з часом, слова, які сприяли позиції кожної статті в рейтинговому списку, були виділені (техніка, відома як «Ключове слово в контексті») при використанні веббраузера для доступу до статей, вивчаючи контекст слів, що виникли із значеннями, які були непередбачуваними на момент побудови профілю, користувачі могли вибрати додаткові слова, що з'явилися в тому ж контексті, щоб додати їх до списку слів, яких слід уникати.

– у SIFT великі колекції профілів порівнювались із кожною статтею, що надсилалась центральним сервером, ефективність була отримана шляхом групування профілів таким чином, щоб це дозволяло виконувати частини процесу фільтрування на групах профілів, а не окремо, SIFT не відділяв слова, що з'являються в статті, тому слова, що з'являються в заголовку статті, тексті статті, включеному тексті або інформації про “підпис”, яка регулярно додається до кожного документа деякими користувачами, з однаковою ймовірністю могли призвести до високого рангу для документа [11].

2.2 Метод опорних векторів

Метод опорних векторів – це дуже потужна і універсальна модель машинного навчання, здатна виконувати лінійну або нелінійну класифікацію, регресію і навіть виявлення викидів [12].

Метод опорних векторів – це метод машинного навчання, метою якого є спроба класифікувати вхідні набори даних в один з двох класів, що відзначається наступними характеристиками:

– для ефективної роботи методу спочатку необхідно використовувати навчальну вибірку, що складається з вхідних і вихідних даних, яка необхідна для побудови моделі методу опорних векторів, і яку в подальшому можна буде використовувати для класифікації нових даних;

– для побудови моделі методу опорних векторів потрібно взяти навчальні вхідні дані, відобразити їх у багатовимірний простір, а потім використовувати регресію, щоб знайти гіперплощину (гіперплощина – це поверхня в n -вимірному просторі, яка розділяє його на два підпростори), яка найкраще розділяла б два класи вхідних даних, після навчання моделі вона здатна класифікувати нові вхідні дані в один з класів за допомогою розділяючої гіперплощини;

– метод опорних векторів є методом входів/виходів: користувач вводить вхідні дані, і на основі розробленої (за допомогою навчання) моделі отримує вихідні результати, теоретично, число входів для методу опорних векторів лежить в діапазоні від одного до нескінченності, однак, в практичному застосуванні, є певні обмеження на розмір вхідної вибірки, що залежать від обчислювальної потужності, наприклад, нехай для конкретного застосування методу опорних векторів використовуються N входів (N – ціле число, в діапазоні від 1 до нескінченності), тоді задача методу опорних векторів полягає в тому, щоб зіставити всі вхідні дані розмірності N і знайти таку гіперплощину розмірності $N-1$, яка найкращим чином розділяла б навчальну вибірку [13].

Опорними векторами називаються об'єкти множини, що лежать на кордонах областей, при цьому класифікація вважається хорошою, якщо область між кордонами порожня, відносно класифікації визначається наступне:

– рішення завдання бінарної класифікації за допомогою методу опорних векторів полягає в пошуку деякої лінійної функції, яка правильно поділяє набір даних на два класи;

– розглянемо задачу класифікації, де число класів дорівнює двом: завдання можна сформулювати як пошук функції $f(x)$, що приймає значення менше нуля для векторів одного класу і більше нуля – для векторів іншого класу, в якості вихідних даних для вирішення поставленого завдання, тобто пошуку класифікуючої функції $f(x)$, дано тренувальний набір векторів

простору, для яких відома їх приналежність до одного з класів, сімейство класифікуючих функцій можна описати через функцію $f(x)$, гіперплощина визначена вектором a й значенням b , тобто $f(x) = ax + b$;

– в результаті рішення задачі, тобто побудови моделі опорних векторів, знайдена функція, що приймає значення менше нуля для векторів одного класу і більше нуля – для векторів іншого класу, для кожного нового об'єкта негативне або позитивне значення визначає приналежність об'єкта до одного з класів;

– найкращою функцією класифікації є функція, для якої очікуваний ризик мінімальний, поняття очікуваного ризику в даному випадку означає очікуваний рівень помилки класифікації;

– безпосередньо оцінити очікуваний рівень помилки побудованої моделі неможливо, це можна зробити за допомогою поняття емпіричного ризику, однак слід враховувати, що мінімізація останнього не завжди призводить до мінімізації очікуваного ризику, цю обставину слід пам'ятати при роботі з відносно невеликими наборами тренувальних даних;

– емпіричний ризик – рівень помилки класифікації на тренувальному наборі;

– в результаті рішення задачі методом опорних векторів для лінійно поділюваних даних отримується функція класифікації, яка мінімізує верхню оцінку очікуваного ризику;

– метод опорних векторів дозволяє: отримати функцію класифікації з мінімальною верхньою оцінкою очікуваного ризику (рівня помилки класифікації), використовувати лінійний класифікатор для роботи з даними, що нелінійно розділяються, поєднуючи простоту з ефективністю [14]-[15].

2.3 Метод керування спілкуванням користувачів

Для керування спілкуванням користувачів потрібно спочатку сформувати статистику за спілкуванням кожного користувача. На подальших

етапах статистика має оновлюватися для визначення наступних керуючих рішень.

Для формування статистики спочатку мають бути проаналізовані повідомлення. Етап аналізу складається з двох підетапів:

- виявлення повідомлень, в яких зустрічаються слова, які входять в стандартний фільтр: лайливі, образливі слова, при цьому застосовується техніка стемінгу, тобто враховуються словоформи, окрім того може застосовуватися контекстний аналіз, коли для визначення зв'язку слів застосовується структура речення (наприклад, структура речення, в якому обговорюється зарплата працівників);

- аналіз повідомлень, які було віднесено до цензурованих експертом.

Під час аналізу повідомлень, які було віднесено до цензурованих, застосовується алгоритм TF-IDF, виділяються слова, які найчастіше зустрічаються. При цьому з даного набору вилучаються загальні слова, які найчастіше зустрічаються в цій мові в будь-якому тексті.

Далі виконується аналогічний аналіз для повідомлень зі спілкування користувачів, які не було цензуровано. Серед них аналогічним чином виділяються слова, які найчастіше зустрічаються.

Далі з першого отриманого набору слів, що найчастіше зустрічаються, вилучаються слова, які входять до другого набору. Слова, що лишилися, вважаються характерними і за допомогою них виконується розмічування повідомлень, які ще не були оцінені експертом.

За проаналізованими повідомленнями виконується накопичення статистики за кожним користувачем:

- кількість повідомлень, які були визначені як цензуровані;
- кількість підозрілих повідомлень, тобто повідомлень, які були автоматично визначені як такі, що мають бути цензуровані;
- кількість нецензурованих повідомлень;
- кількість випадків попередньої заборони спілкування;
- час з останньої повної заборони спілкуватися користувачу;

– кількість позитивних оцінок, які були виставлені користувачу учасниками, які не мали заборон на спілкування.

Схема роботи методу керування спілкуванням користувачів приведена на рис. 2.1.



Рисунок 2.1 – Схема роботи методу керування спілкуванням користувачів

Отриманий масив даних використовується для формування вибірки даних шляхом створення зрізів на кожен місяць. На виході для кожного користувача визначається його стан в поточний момент: спілкування або заборона спілкування.

Вибірка даних використовується для роботи методу опорних векторів.

Далі виконується аналіз нових повідомлень, відбувається переобчислення статистики, а на основі переобчисленої статистики виконується визначення класу для кожного користувача. За визначеним класом виносяться рекомендовані рішення – тимчасово заборонити спілкування користувача або спостерігати за його активністю, або дозволити спілкування повністю.

2.4 Висновки за розділом 2

Проаналізовано метод інтелектуального аналізу даних, які можуть використовуватися для фільтрації та обробки тексту. Проаналізовано метод опорних векторів. Розроблено власний метод керування спілкуванням користувачів. Метод базується на застосуванні фільтрації повідомлень шляхом визначеного фільтру слів, словосполучень та конструкцій речень та автоматичному аналізі цензурованих повідомлень, обчисленні статистики кожного користувача в кожен період часу, винесенні рекомендованого рішення щодо користувача на основі застосування методу опорних векторів для класифікації користувачів в поточний момент.

3 ОПИС ПРОГРАМИ

3.1 Визначення вимог до програмного забезпечення

Розроблювана програма повинна виконувати такі функції:

- створення акаунтів працівників;
- пошук акаунтів;
- авторизація акаунтів;
- відображення переліку приватних комунікацій;
- відкриття приватного спілкування;
- вилучення приватного спілкування;
- надсилання повідомлення в приватну комунікацію;
- додавання приватного спілкування;
- створення відділу;
- редагування даних акаунту;
- включення учасників до відділу;
- вилучення учасників з відділу;
- відображення переліку доступних комунікацій відділів;
- надсилання повідомлення в комунікацію всередині відділу;
- висловлення скарги про повідомлення;
- прогнозування статусу користувача;
- обмеження на доступ користувача;
- надсилання попередження користувачу.

Адміністратор взаємодіє з програмою шляхом виконання таких дій:

- створення акаунтів працівників;
- пошук акаунтів;
- відображення переліку комунікацій відділів;
- створення відділу;
- редагування даних акаунту;
- включення учасників до відділу;
- вилучення учасників з відділу;

- висловлення скарги про повідомлення;
- перегляд повідомлень з комунікації відділу;
- прогнозування статусу користувача;
- обмеження на доступ користувача;
- надсилання попередження користувачу.

На рис. 3.1 приведено діаграму прецедентів взаємодії адміністратора з програмою.



Рисунок 3.1 – Діаграма прецедентів взаємодії адміністратора з програмою

На рис. 3.2 приведено діаграму прецедентів взаємодії користувача з програмою.

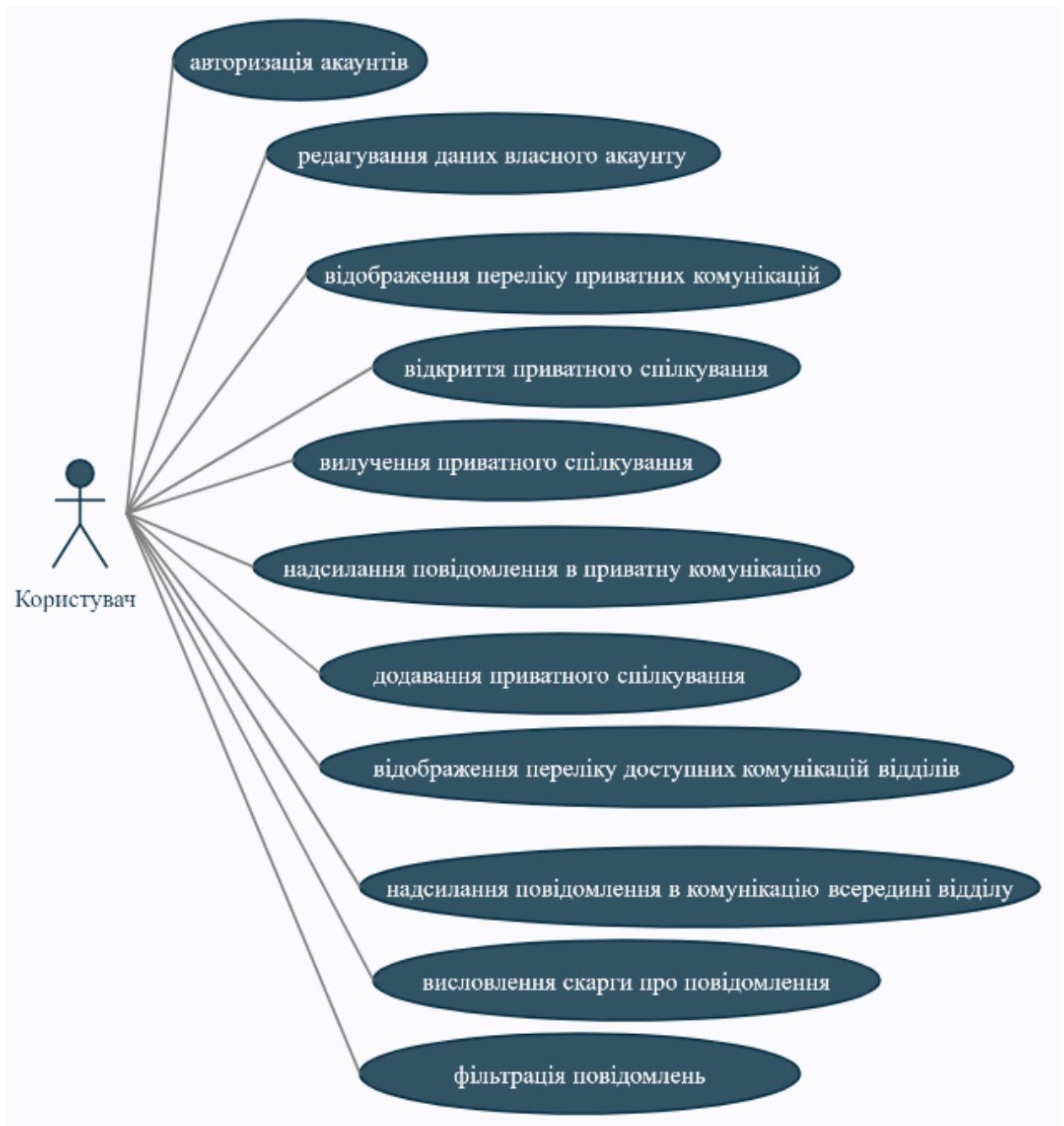


Рисунок 3.2 – Діаграма прецедентів взаємодії користувача з програмою

Користувач взаємодіє з програмою шляхом виконання таких дій:

- авторизація акаунтів;
- відображення переліку приватних комунікацій;
- відкриття приватного спілкування;

- вилучення приватного спілкування;
- надсилання повідомлення в приватну комунікацію;
- додавання приватного спілкування;
- редагування даних власного акаунту;
- відображення переліку доступних комунікацій відділів;
- надсилання повідомлення в комунікацію всередині відділу;
- висловлення скарги про повідомлення;
- фільтрація повідомлень.

3.2 Вибір мови програмування

У процесі вибору мови програмування для реалізації програмного забезпечення порівнювались мови програмування Java та C#, які використовуються для створення застосунків з графічним інтерфейсом для настільних систем.

Основні особливості мови програмування C#:

- була отримана з C++ та Java;
- це частина Microsoft Visual Studio;
- проста: жодних покажчиків, немає таких операторів, як “::” чи “->”, змінювані діапазони примітивних типів, управління пам'яттю та збір сміття відбувається автоматично і не потребує явного коду;
- сучасна: побудована відповідно до сучасних тенденцій, потужна, масштабована, надійна, будь-який компонент можна перетворити на вебсервіс із вбудованою підтримкою;
- об'єктно-орієнтована: підтримує такі концепції об'єктно-орієнтованого програмування, як успадкування, інкапсуляція, поліморфізм, інтерфейси тощо, C# вводить структури, що дозволяють примітивним типам стати об'єктами;
- типобезпечна: це не дозволяє виконувати небезпечний код, посилавні типи ініціалізуються null-значенням, а типи зі значеннями

компілятором ініціалізуються до нуля, границі масивів перевіряються та індексуються на основі нуля;

– сумісність: включає вбудовану підтримку COM та застосунків на базі Windows, компоненти з VB NET можна безпосередньо використовувати в C# [16]-[18].

У свою чергу мова програмування Java має такі особливості:

- проста: вона має простий синтаксис і його легко вивчити;
- об'єктно-орієнтована: вона дотримується таких концепцій об'єктно-орієнтованого програмування, як успадкування, абстракція, поліморфізм, інкапсуляція;
- надійна: Java пропонує управління пам'яттю та обробки винятків за допомогою автоматичного збору сміття та обробки винятків;
- незалежність від платформи: програми запускаються на будь-якій платформі за допомогою Java Runtime Environment;
- безпечні стабільні функції Java дозволяють розробити безвірусну систему;
- багатопотоковість: багатопотоковість Java дозволяє розробляти програму для виконання декількох завдань одночасно, таким чином, вона використовує одну і ту ж пам'ять та інші ресурси для виконання декількох потоків одночасно;
- архітектурна нейтральність: компілятор генерує байткоди, що не залежать від архітектури комп'ютера;
- висока продуктивність: Java забезпечує високу продуктивність за допомогою компілятора;
- розподіленість: програми можуть бути розроблені для роботи в комп'ютерних мережах [18].

Описані характеристики порівнювались за основними критеріями, за якими має прийматися рішення щодо вибору. Результати зведено до табл. 3.1.

Таблиця 3.1 – Порівняння мов програмування

Критерій	C#	Java
Об'єктно-орієнтована парадигма	+	+
Безкоштовне рішення	—	+
Створення графічного інтерфейсу	+	+
Засоби роботи в мережі	+	+
Робота з вільними системами керування базами даних	+	+
Кросплатформність	—	+

Мова Java виділена для розробки, оскільки на відміну від мови C#, вона дозволяє створювати безкоштовні рішення, які мають під собою кросплатформну основу, тобто з часом таку систему можна буде розширити за необхідності, хоча у всьому іншому це актуальні і достатньо співмірні мови.

3.3 Проєктування бази даних

Системою керування базами даних буде виступати MySQL. Такий вибір пояснюється факторами, які відповідають наявності у неї наступних переваг:

– MySQL підтримує декілька механізмів зберігання, кожен зі своїми специфікаціями, тоді як інші системи, такі як SQL-сервер, підтримують лише один механізм зберігання;

– MySQL має високу продуктивність у порівнянні з іншими системами реляційних баз даних: це пояснюється простотою конструкції та підтримкою двигунів із декількома накопичувачами;

– захист даних: MySQL відома у всьому світі як найбезпечніша та найнадійніша система керування базами даних, що використовується у популярних в тому числі вебзастосунках, включаючи WordPress, Drupal, Joomla, Facebook та Twitter, захист даних та підтримка обробки транзакцій, які супроводжують останню версію MySQL, можуть принести велику користь будь-якому бізнесу, особливо якщо це електронна комерція, яка передбачає часті грошові перекази;

– економічно ефективне рішення, що порівняно дешевше з точки зору вартості в порівнянні з іншими реляційними базами даних: насправді рішення є безкоштовним, але комерційне видання має плату за ліцензію, що також є економічно вигідним порівняно з платою за ліцензування таких продуктів як Microsoft SQL Server;

– кросплатформність – система керування базами даних MySQL працює на багатьох платформах, що означає, що його можна розгорнути на більшості машин: інші системи, такі як MS SQL Server, працюють лише на платформі Windows [19]-[20].

Таблиці зпроектованої бази даних програмного застосунку:

- таблиця Account – акаунт працівника;
- таблиця PrivateMessage – приватне повідомлення;
- таблиця Department – відділ компанії;
- таблиця DepartmentAccounts – акаунти, що входять до відділу компанії;
- таблиця DepartmentMessage – повідомлення всередині відділу;
- таблиця Ban – заборони на доступ.

Таблиця Account містить наступні стовпці:

- id – номер;
- login – логін;
- password – пароль;
- surname – повне ім'я працівника;
- phone – номер телефону;
- position – посада;
- level – рівень акаунту;
- type – тип акаунту.

Таблиця PrivateMessage включає стовпці:

- id – номер;
- from – відправник повідомлення;
- to – отримувач повідомлення;
- moment – день і час надсилання повідомлення;
- messtext – текст повідомлення;
- type – тип повідомлення (для цензурування).

Стовпець from є зовнішнім ключем, що визначає зв'язок з таблицею Account. Стовпець to визначає такий самий зовнішній ключ.

Таблиця Department містить наступні стовпці:

- id – номер;
- name – назва відділу;
- company – назва компанії (або підрозділу, в межах якого існує відділ);
- address – адреса відділу;
- phone – телефон відділу.

Таблиця DepartmentAccounts включає стовпці:

- id – номер;
- dep – відділ;
- user – працівник.

Таблиця DepartmentAccounts створена для реалізації зв'язку багато-до-багатьох між таблицями Department і Account. Кожен підрозділ може включати багато користувачів, але в той же час кожен користувач може входити в декілька підрозділів. Останнє пов'язане з тим, що в деяких компаніях працівник може входити в декілька відділів одночасно (бути сумісником), а з іншого боку відділи можуть позначати не тільки відділи, а групи, які створюються в компанії зокрема для комунікації за різними проблемами.

Стовпець dep є зовнішнім ключем, що визначає зв'язок з таблицею Department.

Стовпець user є зовнішнім ключем, що визначає зв'язок з таблицею Account.

Таблиця DepartmentMessage містить наступні стовпці:

- id – номер;
- from – відправник повідомлення;
- moment – день і час надсилання повідомлення;
- messtext – текст повідомлення;
- type – тип повідомлення (для цензурування).

Стовпець from є зовнішнім ключем, що визначає зв'язок з таблицею DepartmentAccounts.

Таблиця Ban включає стовпці:

- id – номер;
- user – працівник, якому виставлено заборону;
- frommoment – день і час початку заборони;
- tomoment – день і час завершення заборони;
- type – тип заборони.

Стовпець user є зовнішнім ключем, що визначає зв'язок з таблицею Account.

Структура описаної бази даних зведена до схеми на рис. 3.3.

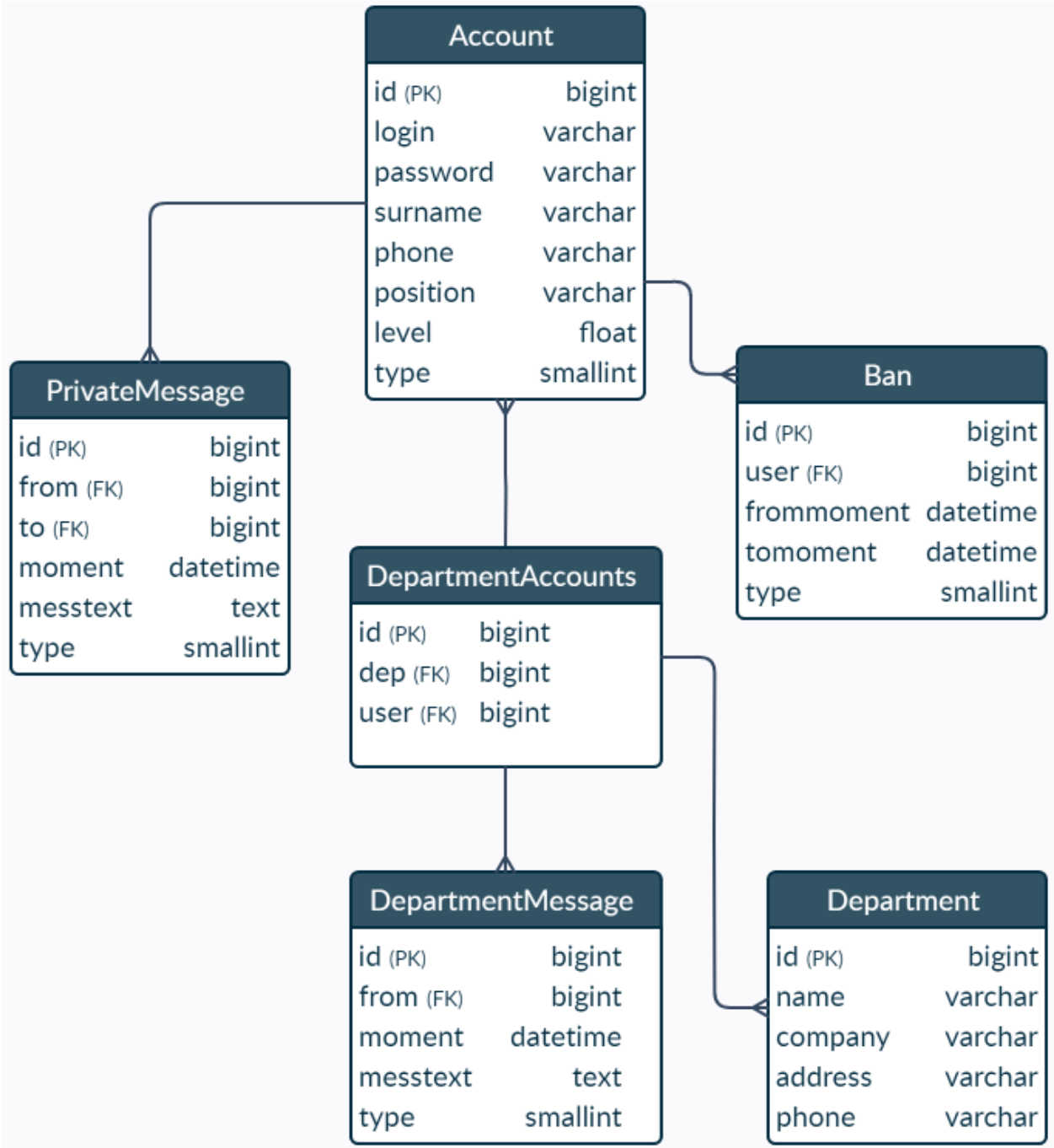


Рисунок 3.3 – Структура бази даних

3.4 Висновки за розділом 3

Визначено функції, які повинна виконувати програма. Створено діаграми прецедентів для моделювання функціональних вимог для роботи користувача та адміністратора з програмою.

Обрано мову програмування Java для реалізації програми. Системою керування базами даних обрано MySQL. Визначено набір таблиць, їх структуру та побудовано структурну схему бази даних.

4 РОЗРОБКА ПРОГРАМИ

4.1 Опис реалізації структури програмного забезпечення

Програмне забезпечення керування спілкуванням користувачів вимагає віддаленої комунікації користувачів, тому в основі визначення структури програмного забезпечення клієнт-серверна модель.

Програмний проєкт в результаті повинен складатися з наступних пакетів:

- waiter – сервер;
- connection – клієнт;
- папка PDO – модель предметної області;
- participant – контроль за повідомленнями на сервері;
- libs – бібліотечні засоби.

Модель предметної області включає зокрема такі класи:

- Database.java – клас роботи з базою даних;
- Ban.java – клас заборони на спілкування;
- Department.java – клас відділення, в межах якого відбувається групове спілкування;
- Communic.java – базовий клас, який використовується сервером для збереження повідомлення як такого (без розбиття на приватне чи групове);
- PrivateMessage.java – модель приватного повідомлення;
- DepartmentMessage.java – модель повідомлення для групи користувачів (у межах відділення);
- Account.java – модель акаунту.

На рис. 4.1 приведена діаграма розгортання.

На клієнтських комп'ютерах працює тільки клієнтський застосунок з пакету connection. На сервері працює серверний застосунок пакету waiter, окрім того сама система управління базою даних, яка накопичує і надає доступ до всіх потрібних даних, а також модуль керування спілкуванням

користувачів participant, в якому реалізується відфільтровування повідомлень та прийняття рішень.

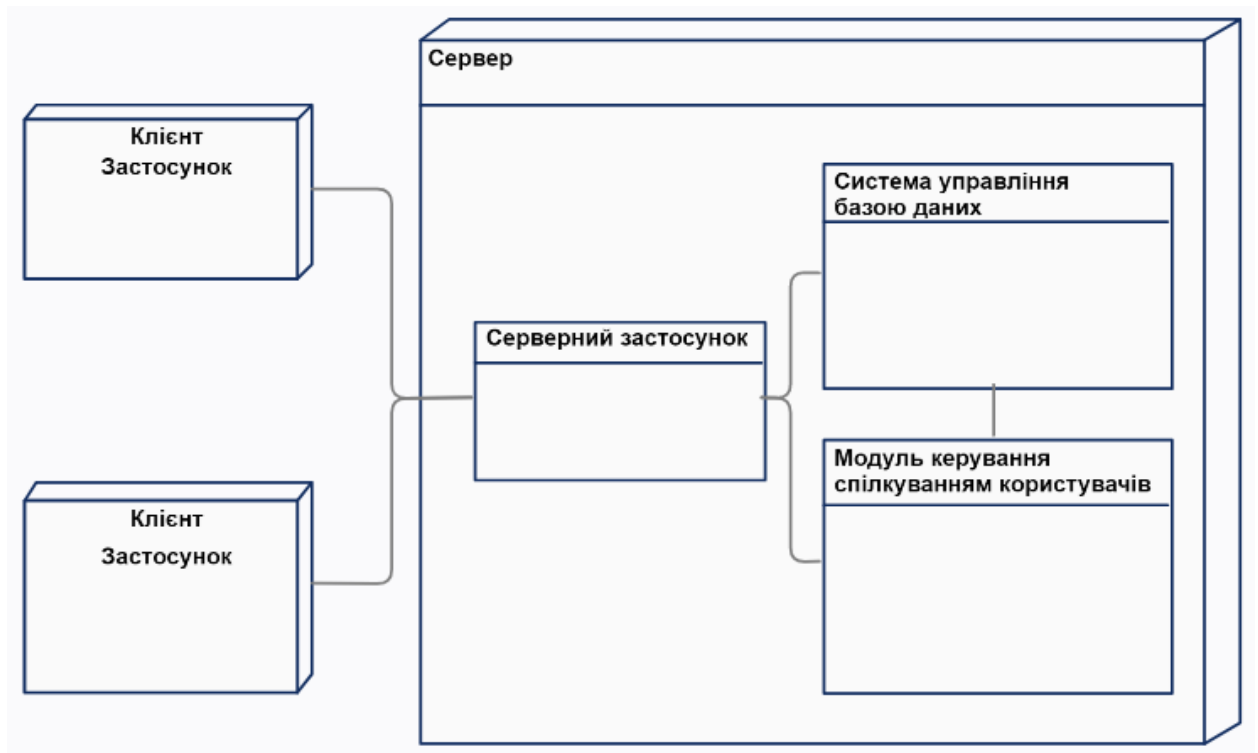


Рисунок 4.1 – Діаграма розгортання

До складу пакету connection входить клас Connection.java, який реалізує з'єднання з сервером, а також набір засобів забезпечення графічного інтерфейсу, зокрема:

- Auth.java – форма аутентифікації акаунту;
- Profile.java – форма для роботи над власним профілем;
- Private.java – форма для роботи з приватними повідомленнями;
- Department.java – форма для роботи з повідомленнями в межах відділення.

До складу даного пакету також входить підпакет admin, який має наступний набір засобів забезпечення графічного інтерфейсу:

- Auth.java – форма аутентифікації акаунту;
- Accounts.java – форма роботи з акаунтами користувачів;
- Departments.java – форма для роботи з даними відділень.

4.2 Опис розроблених модулів

Для реалізації клієнт-серверної взаємодії використано в основі класи `ServerSocket` і `Socket` з пакету `java.net.Socket`, який створює програмний інтерфейс комунікації між сервером та клієнтом. Для цього в класі `Waiter` створюється серверний сокет на основі класу `ServerSocket`, а далі реалізується нескінченний цикл (сервер працює в режимі очікування) очікування з'єднання від клієнтів, кожен з яких реалізується на основі класу `Connection`:

```
ServerSocket sS = new ServerSocket (pnum);
while (true)
{
    Socket cS = sS.accept ();
    Connection connection = new Connection (this, cS);
    connection.start ();
}
```

На рис. 4.2 приведено схему взаємодії між сервером та клієнтами в межах сокетів. Для цього спочатку сервер має прийняти таке з'єднання від клієнта, повідомити йому про успішність цієї операції, а далі прочитати повідомлення від клієнта і відповісти йому надсиланням відповідної інформації. Тобто будь-які повідомлення мають пересилатися під час контактування хоч між окремими клієнтами, хоч у межах відділу на сервер, де вони аналізуються, записуються в базу даних, а відповідь надсилається клієнту. Даний цикл продовжується до того моменту, поки клієнт продовжує існувати, тобто допоки адміністратор або користувач не закриють свій екземпляр програмного застосунку. Потенційно сервер має працювати при цьому нескінченно, обробляючи будь-які вхідні запити.

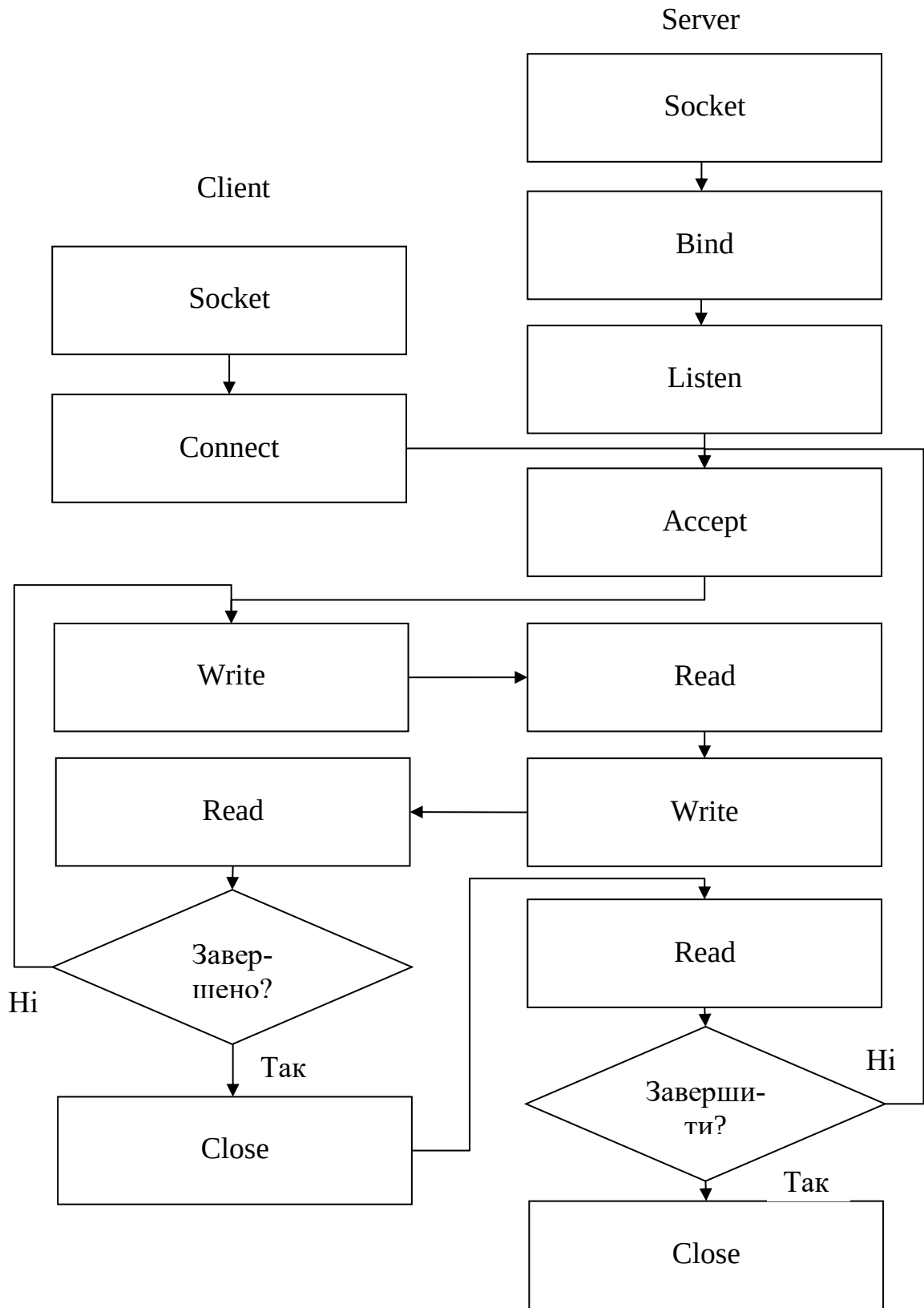


Рисунок 4.2 – Схема взаємодії між сервером та клієнтами в межах сокетів

Клас Connection реалізує клієнтську сторону взаємодії, включає наступні атрибути:

- Socket cS – клієнтський сокет підключення;
- Waiter waiter – дані серверу, з яким встановлюється з'єднання;
- int num – номер порту, через який відбувається комунікація;
- Account acc – акаунт користувача, через який виконується робота;
- PrintWriter pw – механізм запису даних в потік;
- BufferedReader br – механізм читання даних з потоку.

Реалізовано такі методи класу:

- Account getAcc() – метод, який повертає об'єкт акаунту, під яким реалізується робота клієнта;
- void setAcc (Account ac) – метод, який встановлює в якості акаунту клієнта ac;
- int getPNum () – метод, який повертає номер порту, під яким відбувається взаємодія клієнта з сервером;
- Connection (Waiter waiter, Socket cS) – конструктор, який виконує визначення параметрів сервера waiter та сокета cS;
- void run () – метод, який забезпечує запуск клієнта;
- void CSobtainer () – метод, який виконує керування з'єднанням з сервером;
- void oneAction (String act) – метод, який виконує розбір рядка команд act, виділення окремих команд та ініціалізацію роботи відповідних обробників команд;
- void processAuth (String[] param) – обробник аутентифікації клієнта, в якому забезпечується перевірка введених даних акаунта користувача, після чого до сервера додається з'єднання з клієнтом;
- void processNoConn () – метод, який забезпечує від'єднання від сервера;
- void processCommunic (String[] param) – метод, який реалізує направлення приватного повідомлення серверу;
- void processGroupCommunic (String[] param) – метод, який реалізує направлення групового повідомлення серверу;

- void processSignup (String[] param) – метод, який виконує реєстрацію акаунта;
- void Answer (String text) – метод, який виконує відправлення в потік сформованого повідомлення;
- void post (String info, int mode, String param) – метод, який виконує підготовку повідомлення для відправлення у з'єднання з сервером.

4.3 Висновки за розділом 4

Реалізовано програмне забезпечення керування спілкуванням користувачів, описано розгортання програмного забезпечення, описано схему взаємодії між клієнтами і сервером та принципи реалізації взаємодії між ними на основі сокетів, описано основні реалізовані класи.

5 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

5.1 Призначення програми

Програма призначена для реалізації робочого спілкування в корпоративній мережі з забезпеченням можливості передачі повідомлень в приватному режимі, в режимі спілкування в межах відділів, фільтрування спілкування для уникнення заборонених тем, а також образливих, лайливих слів, при чому основа для прийняття таких рішень реалізується автоматичним чином.

5.2 Умови виконання програми

Для роботи сервера мають забезпечуватися наступні мінімальні вимоги:

- процесор з 4 ядрами з частотою 2 ГГц;
- обсяг оперативного запам'ятовуючого пристрою – 8 Гб;
- вільний дисковий простір – 100 Гб;
- пропускна здатність мережного інтерфейсу – 1 Гбіт/с.

5.3 Виконання програми

Програмне забезпечення складається з трьох елементів: сервера, клієнтського застосунку користувача та клієнтського застосунку адміністратора.

Для того щоб розпочати роботу з корпоративним чатом, потрібно спочатку запустити на виконання сервер, який не має користувацького інтерфейсу, після цього користувачі можуть запускати клієнтські застосунки.

Якщо запустити клієнтський застосунок користувача, то спочатку потрібно ввести дані акаунту користувача, а тоді буде відкрито вікно доступу до основних функцій (рис. 5.1).

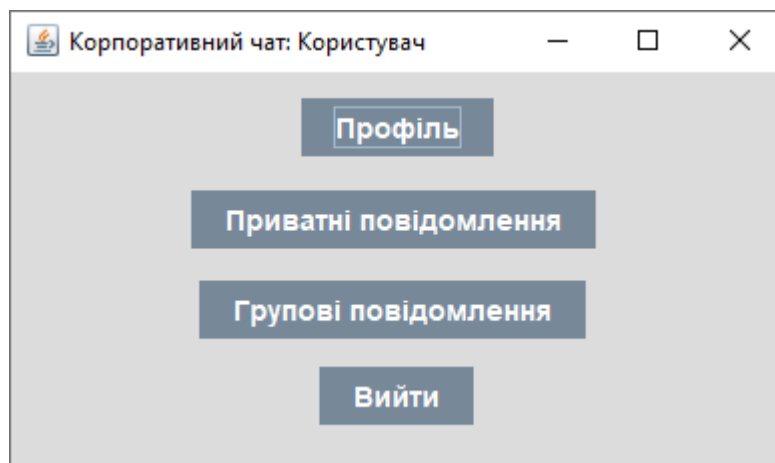


Рисунок 5.1 – Основне меню користувача

При натисканні на кнопку «Приватні повідомлення» відкривається вікно перегляду приватного спілкування (рис. 5.2).

У цьому вікні зліва розташований еперлік всіх приватних бесід. Звідси можна обрати будь-яку (двічі натиснути на відповідного користувача), після чого повідомлення, які було передано між авторизованим користувачем і обраним будуть виведені у найбільшому елементі, розташованому правіше. Тут виводяться всі повідомлення, при чому темним кольором і з вирівнюванням по правій частині виводяться власні повідомлення, а з вирівнюванням по лівій частині і білим кольором виводяться повідомлення, які було отримано від іншого користувача. Для кожного повідомлення відображається дата і час його відправлення.

У текстовому полі такого ж синього кольору, але розташованому нижче, можна ввести будь-яке повідомлення, після чого можна натиснути на кнопку «Відправити» і повідомлення буде відіслано користувачу, приватне спілкування з яким було відкрито.

Окрім того користувач може ввести логін користувача в нижньому лівому полі, після чого натиснути на кнопку «Відкрити», тоді відкриється нове приватне спілкування (якщо воно ще не було створено), а в перелік користувачів буде додано введений логін.

Якщо з переліку користувачів обрати одне, а потім натиснути на кнопку «Вилучити», то приватне спілкування більше не буде відображатися користувачу.

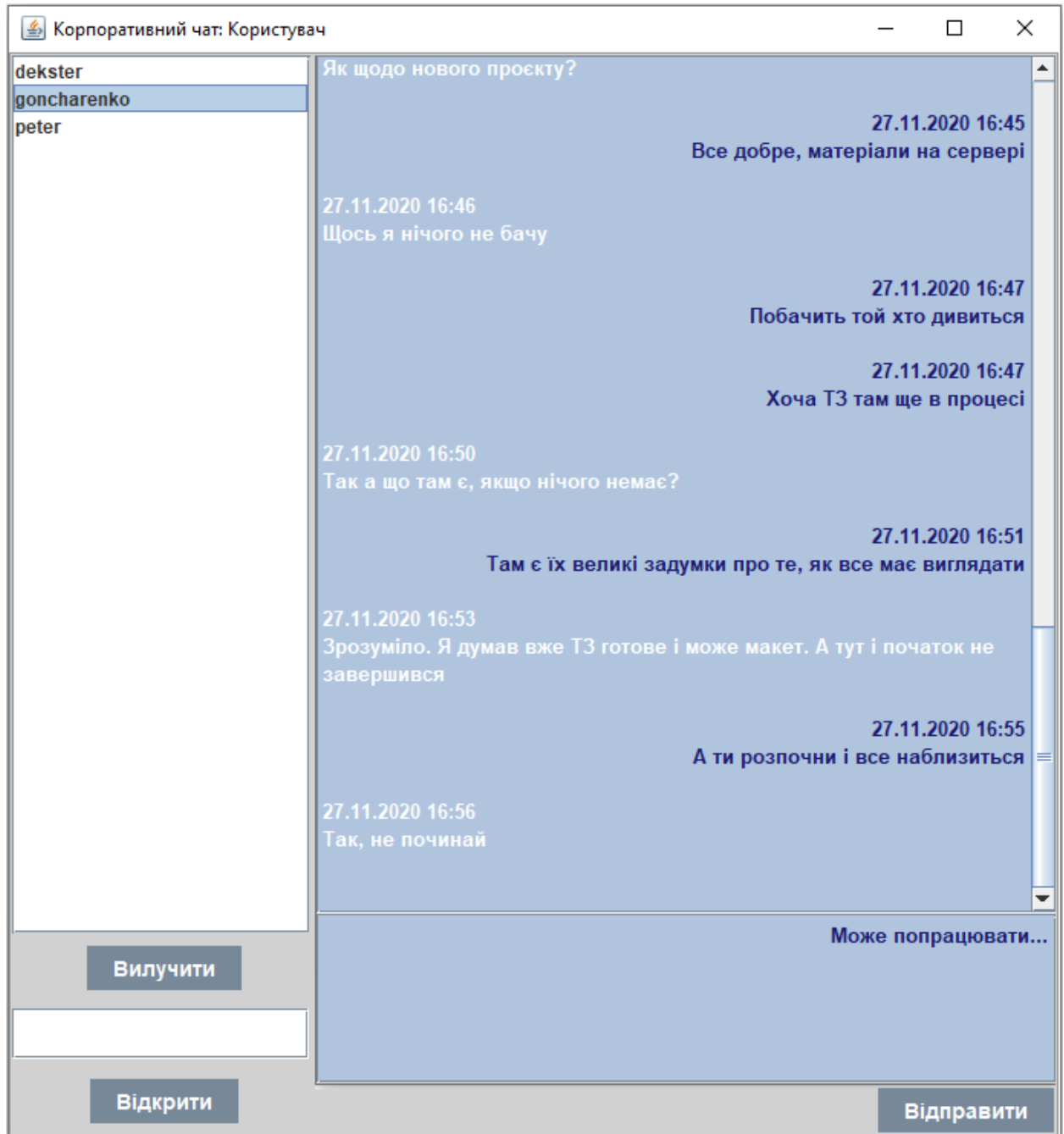


Рисунок 5.2 – Приватні повідомлення користувача

Якщо доступ для користувача заблоковано, то він зможе відкрити дане вікно, але при цьому кнопка «Відправити» буде неактивною, тобто надіслати будь-яке повідомлення користувач не зможе. На рис. 5.3 приведено

приклад такої ситуації. У даному випадку користувач відправив повідомлення з розміром своєї зарплатні, тому доступ для нього було заблоковано. Фільтрація повідомлення відбулась автоматично. Повідомлення віднесено до тієї групи (про розмір зарплатні), яке призводить до автоматичної заборони відправляти нові повідомлення. Адміністратор може в своєму інтерфейсі переглянути автоматично винесені заборони та скасувати за потреби.

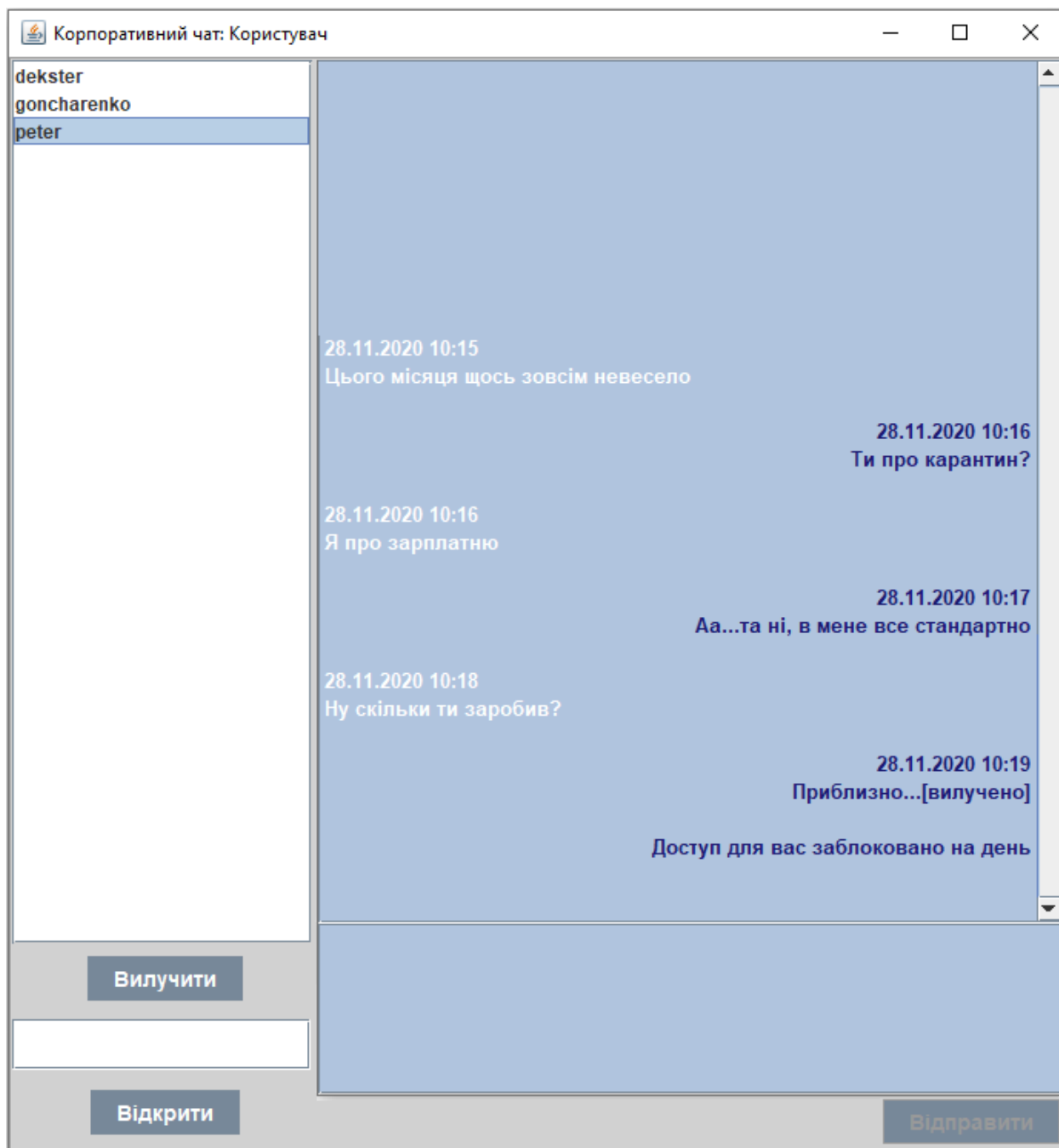


Рисунок 5.3 – Фільтрація спілкування з заборною доступу

Якщо користувач має відмінний рейтинг і в історії його спілкування ніколи не було відфільтрованих повідомлень, то у нього в інтерфейсі також з'являється кнопка «Рекомендувати», що дозволяє підвищити рейтинг користувача, з яким відкрито приватне спілкування.

Якщо користувач натиснув на кнопку «Групові повідомлення» на своїй основній формі, то йому відкривається форма групового спілкування (рис. 5.4).

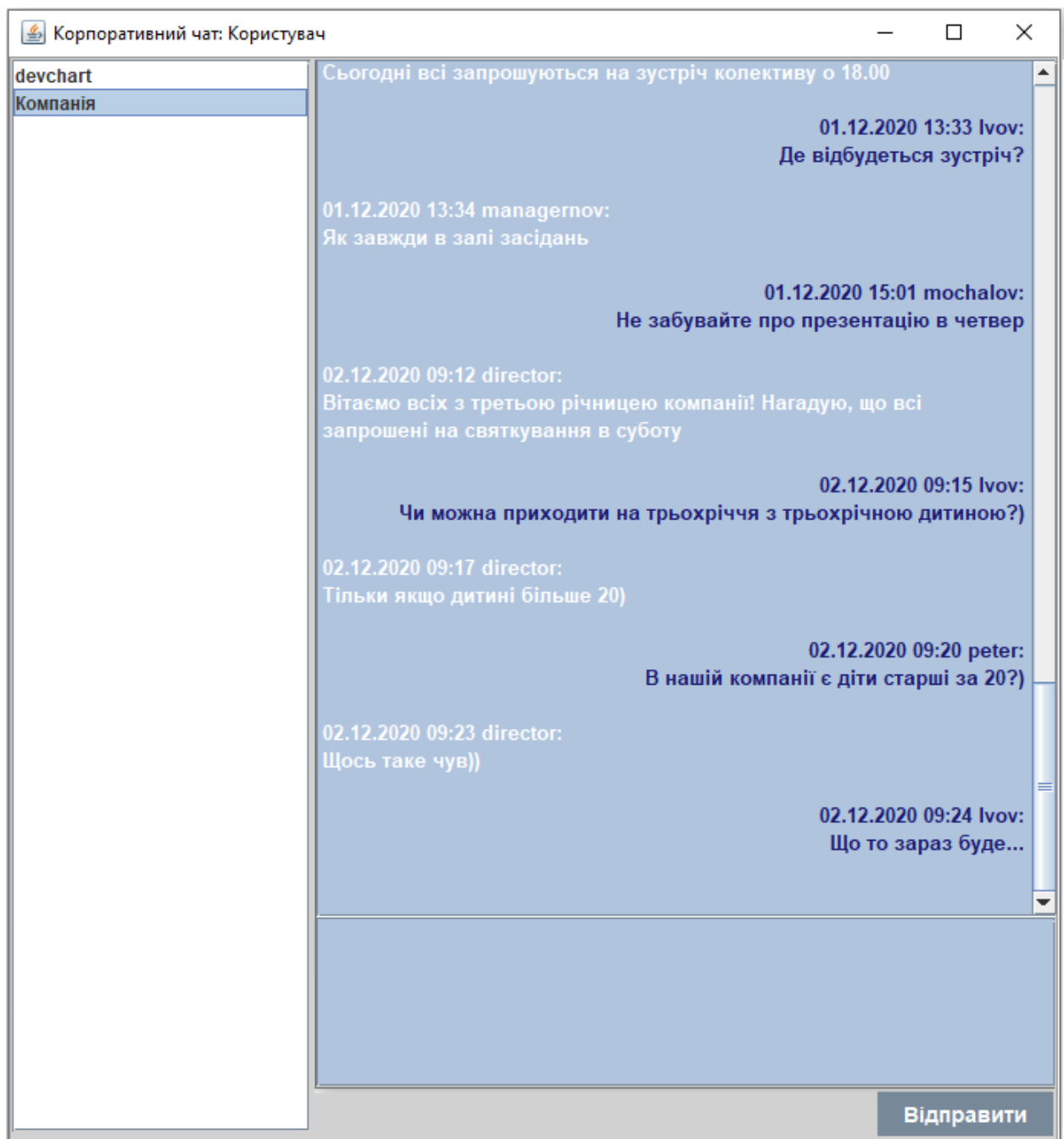


Рисунок 5.4 – Спілкування в групі

В груповому спілкуванні користувачу доступний перелік всіх діалогів у групі, він може обрати будь-який з них, але при цьому не може додати себе до якогось іншого або вилучити з існуючого. Керування групами відбувається тільки адміністратором. Користувач може в заданих для нього групах переглядати отримані повідомлення, а також надсилати власні.

Повідомлення розташовуються одне за одним за принципом чергування: одне за вирівнюванням по правій частині, наступне – по лівій.

Якщо запустити клієнтський застосунок адміністратора, то спочатку порібно ввести дані акаунту адміністратора, а тоді буде відкрито вікно доступу до основних функцій (рис. 5.5).

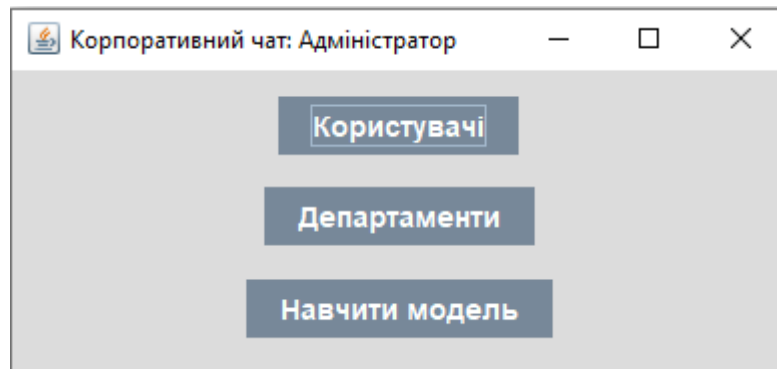


Рисунок 5.5 – Основне меню адміністратора

Якщо натиснути на кнопку «Департаменти», то відкриється вікно керування департаментами для групового спілкування (рис. 5.6).

У переліку зліва перелічено всі доступні департаменти. Якщо обрати один з них (не обов’язково реалізує департаменти, може бути просто групою користувачів), то правіше в переліку «Учасники» з’явиться набір учасників, яких було додано до цього департаменту (групи). Можна обрати один з записів та натиснути на кнопку «Вилучити» для вилучення користувача з цієї групи. Можна ввести в полі нижче логін акаунта та натиснути на кнопку «Додати» для додавання введеного акаунту до спілкування в даному департаменті (групі).

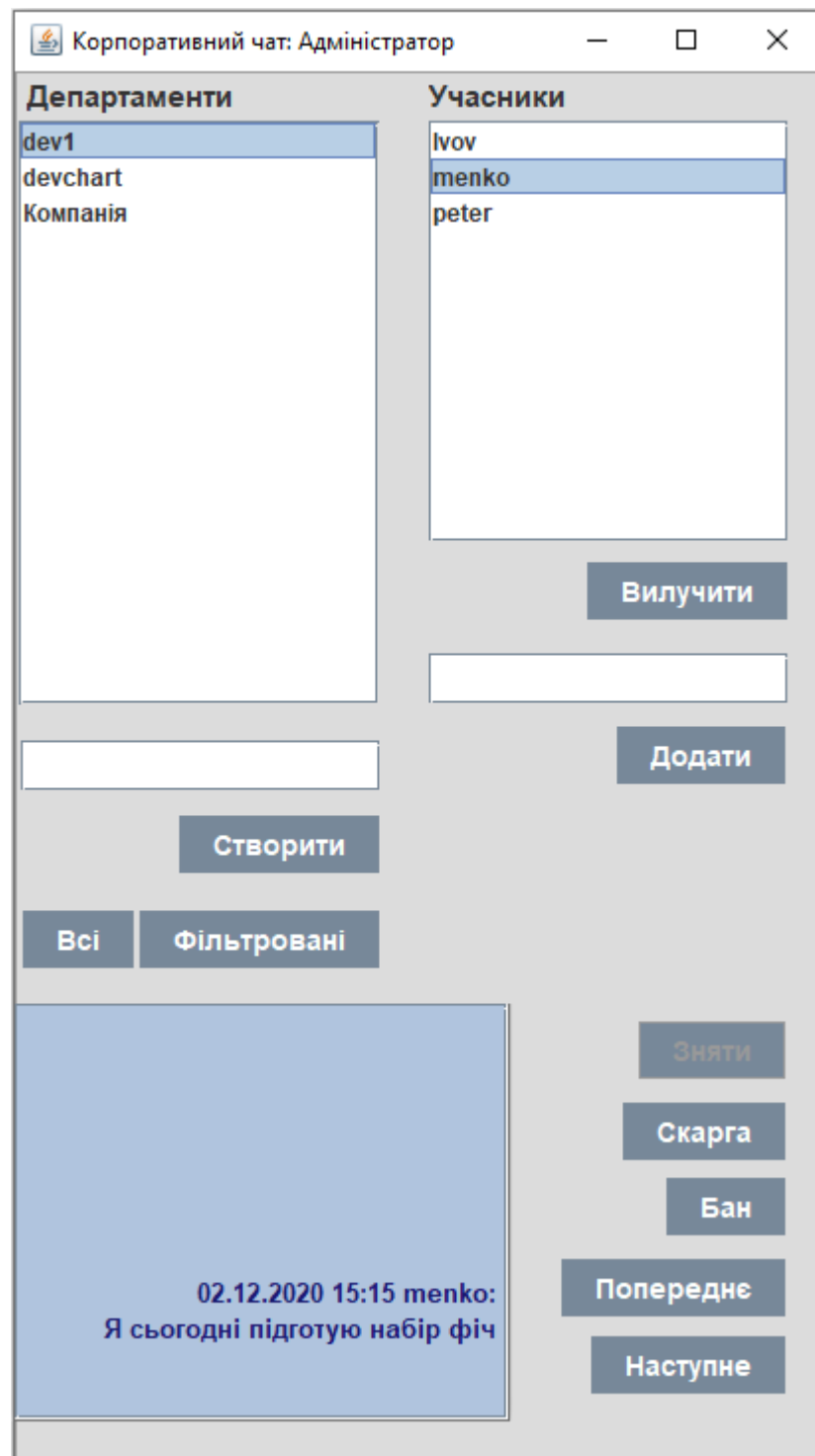


Рисунок 5.6 – Керування департаментами адміністратором

Якщо в полі нижче переліку «Департаменти» ввести назву департаменту (групи) і натиснути на кнопку «Створити», то буде створено нову групу і додано її до переліку «Департаменти».

Якщо обрати одного з користувачів з переліку «Учасники», то в синьому полі буде відображено повідомлення даного користувача з обраного

групового спілкування. За допомогою кнопок «Попереднє» і «Наступне» можна переміщуватися між повідомленнями.

За допомогою кнопки «Бан» можна заборонити доступ до створення нових повідомлень обраному в даний момент користувачу. Заборона доступу спочатку фіксується на день, на другий раз – на три дні, третій – тиждень, четвертий – місяць. За допомогою кнопки «Скарга» відправляється стандартне повідомлення користувачу разом з текстом його власного повідомлення. Таке повідомлення не заслуговує на заборону доступу, але є некоректним на думку адміністратора.

За допомогою кнопки «Зняти» можна зняти заборону доступу або скасувати фільтр з повідомлення. Кнопка неактивна, якщо повідомлення чисте. Повідомлення адміністратору відображаються без фільтрів.

На рис. 5.7 приведено вигляд вікна програми, якщо адміністратор натиснув на кнопку «Користувачі» на основній власній формі.

Для обрання потрібного акаунту можна виконати пошук, для цього потрібно в текстове поле ввести логін та натиснути на кнопку «Шукати». У переліку нижче виведено результати такого пошуку або всі акаунти, зареєстровані в системі. За допомогою кнопки «Нагляд» можна перейти до відображення тільки тих акаунтів, які були за допомогою розробленого методу класифіковані як ненадійні.

Якщо обрати користувача з переліку, то у переліку «Заборона доступу» буде відображено перелік всіх виконаних відповідних дій. За допомогою кнопки «Зняти» можна скасувати активну заборону.

Для обраного користувача можна внести зміни в дані його профілю, після чого натиснути на кнопку «Редагувати» для збереження.

Внизу виводяться повідомлення обраного користувача. За допомогою кнопки «Фільтровані» можна перейти до перегляду тих повідомлень, які було фільтровані системою. За допомогою кнопок «Попереднє» та «Наступне» можна переходити між спілкуваннями (в перелік будуть входити і приватні, і групові спілкування).

Корпоративний чат: Адміністратор

Шукати

Всі

Нагляд

dekster
director
goncharenko
lvov
managernov
menko
mochalov
peter

Додати

Всі

Фільтровані

Бан

Скарга

Попереднє

Наступнє

Заборона доступу

25.12.2020 17:17 - 26.12.2020 17:17

Зняти

Логін

menko

Прізвище, ім'я

Менко Сергій Іванович

Телефон

+380671011001

Посада

менеджер з продажів

Тип

користувач

Редагувати

01.12.2020 12:43 peter:
Не встигаю завершити вчасно

01.12.2020 12:44 menko:
Строки невірно оцінені?

01.12.2020 12:45 peter:
Та ні, функціонал довелось розширити

01.12.2020 12:45 menko:
Як завжди з ними

Рисунок 5.7 – Керування користувачами адміністратором

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

У роботі виконується дослідження та програмна реалізація методів керування спілкуванням користувачів.

Розробка програмного додатку має призвести до створення системи, в межах якої можна буде керувати взаємодією користувачів. Користувачами будуть виступати працівники підприємства. Це дозволить керувати їх робочою взаємодією, вилучаючи зокрема інформацію зі спілкування, що не повинна розповсюджуватися, а також виконувати за необхідності фільтрацію (цензурування).

Очікується, що підприємства будуть сплачувати за кожний встановлений екземпляр програми, а ніяких інших додаткових сплат за купівлю не буде. Заплановано здійснити продаж 5000 застосунків, кожен екземпляр буде коштувати 25 грн. Це за рахунок невеликої суми для підприємств має дозволити продати таку кількість екземплярів.

6.1 Планування розробки програмного виробу

Весь комплекс розробки програмного забезпечення можна поділити на етапи, для кожного етапу вказується трудомісткість, кількість виконавців та тривалість робіт [21]. У розробці приймають участь програміст впродовж 2.5 місяців і консультант впродовж 0.5 місяця. Розробка програми починається 14 вересня 2020 року та має бути завершена до 27 листопада 2020 року.

Отримана інформація зведена в табл. 6.1, за даними якої складається зведений стрічковий графік планування розробки програмного продукту, який представляє собою таблицю, в першому стовпці якої розміщені в порядку збільшення термінів початку виконання всі види роботи, а навпроти – календарний період їх виконання.

Тривалість робіт розраховується за формулою (6.1):

$$T_{\text{ц}} = \frac{Q}{R}, \quad (6.1)$$

де $T_{\text{ц}}$ – довжина циклу, днів;

Q – трудомісткість, людино-днів;

R – кількість виконавців, люд.

Таблиця 6.1 – Характеристика робіт по розробці програми

Найменування роботи	Трудомісткість		Виконавці		Тривалість, днів
	людино-днів	% до підсумку	спеціальність	кількість	
1. Дослідження вимог до проєктованого продукту	10	14.7	програміст	2	5
			консультант		
2. Вибір інструментів проєктування, дослідження існуючих рішень	4	5.8	програміст	1	4
3. Проєктування загальної структури програми	10	14.7	програміст	2	5
			консультант		
4. Розробка схеми функціонування програми	2	2.9	програміст	1	2
5. Створення програмного коду	23	33.8	програміст	1	23
6. Тестування та налагодження програмного продукту	13	19.1	програміст	1	13
7. Складання програмної документації	6	8.8	програміст	1	3
			консультант		
Всього	68	100.0			55

Необхідні початкові дані:

– час на розробку – 2.5 місяці;

6.2 Визначення витрат на розробку програми

6.2.1 Розрахунок основної заробітної плати

Результати виконаних розрахунків основної заробітної плати зведені в табл. 6.3.

Розрахунок заробітної плати проводиться у відповідності з формулою (6.2):

$$ЗП_{\text{осн}} = \sum_1^i (ЗП_{\text{мес}_i} \cdot T_{p_i}), \quad (6.2)$$

де $ЗП_{\text{мес}_i}$ – щомісячна заробітна плата i -того працівника, грн.;

T_{p_i} – час, витрачений на розробку програми i -тим працівником, міс. [21].

Таблиця 6.3 – Розрахунок основної заробітної плати

Посада виконавця	Кількість, осіб	Місячний оклад	Кількість, міс. роботи	Премії та доплати, грн.	Середня місячна зарплата, грн.	Сумарна зарплата, грн.
Програміст	1	5700	2.5	–	5700	14250
Консультант	1	8000	0.5	–	8000	4000
Всього	2					18250

6.2.2 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10 % від основної заробітної плати [21]. Розрахунок основної та додаткової заробітної плати наведений в табл. 6.4.

Таблиця 6.4 – Розрахунок додаткової заробітної плати

Посада виконавця	Додаткова заробітна плата, грн.	Сума основної та додаткової заробітної плати, грн.
Програміст	1425	15675
Консультант	400	4400
Всього	1825	20075

6.2.3 Відрахування на єдиний соціальний внесок

Відрахування на єдиний соціальний внесок визначають у відсотковому відношенні від суми основної і додаткової зарплат, з урахуванням премій і доплат: премії і доплати дорівнюють 0, відрахування на соціальне страхування складають 22% [21].

Розрахунок виконується за формулою (6.3):

$$\text{ВІД} = (\text{ЗП}_{\text{осн}} + \text{ЗП}_{\text{дод}}) \cdot 0.22, \quad (6.3)$$

де $\text{ЗП}_{\text{осн}}$ – основна заробітна плата, грн.;

$\text{ЗП}_{\text{дод}}$ – додаткова заробітна плата, грн. [21].

Отже, відрахування на єдиний соціальний внесок дорівнюють:

$$(18250 + 1825) \cdot 0.22 = 4416.5 \text{ грн.}$$

6.2.4 Визначення витрат на матеріали

У даному підрозділі виконується визначення витрат на матеріали.

Матеріали, які знадобилися для розробки даної програми, представлені в табл. 6.5.

Таблиця 6.5 – Витратні матеріали

Матеріал	Кількість	Вартість, грн.
Папір формату А4	3 пачки	80
Олівці	50 шт.	5
CD-RW диск	5 шт.	30
Тонер для принтера	1 шт.	160

Витрати на матеріали розраховуються за формулою (6.4):

$$M = \sum_{i=1}^n (C_i \cdot N_i \cdot (1 + K_{т.з}) - C_{io} \cdot N_{io}), \quad (6.4)$$

де M – витрати на матеріали, покупні напівфабрикати і комплектуючі вироби, грн.;

$K_{т.з.}$ – коефіцієнт, що враховує транспортно-заготівельні витрати (0,03 – 0,1);

C_i – ціна i -го найменування матеріалу, напівфабрикату, комплектуючого, грн.;

N_i – потреба в i -му матеріалі, напівфабрикаті, комплектуючому;

C_{io} – ціна поворотних відходів i -го найменування матеріалу, грн.;

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів, комплектуючих [21].

Розрахуємо витрати на матеріали:

$$M = 3 \cdot (1 + 0.1) \cdot 80 + 50 \cdot (1 + 0.1) \cdot 5 + 30 \cdot (1 + 0.1) \cdot 5 + (1 + 0.1) \cdot 160 = 880 \text{ грн.}$$

Разом витрати на матеріали склали 880 грн.

6.2.5 Витрати на спеціальне устаткування

Витрати на оплату машинного часу електронно-обчислювальної машини (ЕОМ) для написання програми визначають по формулі (6.5):

$$C_{\text{МО}} = P_{\text{екс}} \cdot t_{\text{МО}}, \quad (6.5)$$

де $C_{\text{МО}}$ – витрати на оплату машинного часу, грн.;

$P_{\text{екс}}$ – експлуатаційні витрати на одну годину машинного часу, грн.;

$t_{\text{МО}}$ – машинний час ЕОМ для написання і відладки цього програмного продукту, год. [21].

Амортизаційні відрахування визначають по формулі (6.6):

$$A = \Phi_6 \cdot \frac{N_a}{100}, \quad (6.6)$$

де Φ_6 – балансова вартість обчислювальної техніки, грн.;

N_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, % [21].

Балансова вартість обчислювальної техніки становить $30000 \cdot (1 + 0,05) = 31500$ грн., а норма амортизаційних відрахувань – 25%.

Таким чином амортизаційні відрахування становлять:

$$A = 31500 \cdot 0.25 = 7875 \text{ грн.}$$

Витрати на електроенергію розраховуються за формулою (6.7):

$$C_B = N_H \cdot \Phi_{\text{еф}} \cdot K_{\text{зч}} \cdot K_{\text{зп}} \cdot C_{\text{ел.в}}, \quad (6.7)$$

де N_H – номінальна потужність ЕОМ, кВт;

$\Phi_{\text{еф}}$ – річний ефективний фонд часу роботи ЕОМ, год.;

$K_{зч}$ – середній коефіцієнт завантаження за часом;

$K_{зп}$ – коефіцієнт завантаження за потужністю;

$\Pi_{ел.в}$ – ціна однієї кВт·год. електроенергії, грн. [21].

Номінальна потужність ЕОМ дорівнює 0,3 кВт. $\Phi_{ef} = 1800$ год.
Середній коефіцієнт завантаження за часом – 0,45. Коефіцієнт завантаження за потужністю – 0,7. Ціна однієї кВт·год електроенергії – 2,11 грн.

Таким чином:

$$C_e = 0,33 \cdot 1800 \cdot 0,45 \cdot 0,7 \cdot 2,11 = 722,17 \text{ грн.}$$

Зарплата обслуговуючого персоналу розраховується за формулою (6.8):

$$ЗП_{обсл} = \Phi ЗП_p \cdot (1 + K_{відр}) \cdot \frac{t_{обсл}}{\Phi_{ef.обсл}}, \quad (6.8)$$

де $\Phi ЗП_p$ – річний фонд зарплати обслуговуючих працівників, грн.;

$K_{відр}$ – коефіцієнт відрахувань на соціальне страхування і до інших фондів, %;

$t_{обсл}$ – час впродовж року, необхідний на техобслуговування ЕОМ, год./рік;

$\Phi_{ef.обсл}$ – річний ефективний фонд часу обслуговуючого персоналу, год./рік [21].

Машину обслуговує один адміністратор з основною заробітною платою 4000 грн./міс. Річний фонд заробітної плати становить відповідно 48000 грн. На одну ЕОМ обслуговуючий витрачає по 1 годині на місяць, що в рік становить 12 годин. Час, необхідний на технічне обслуговування ЕОМ протягом року, дорівнює 1750 год.

Таким чином:

$$ЗП_{обсл} = 48000 \cdot (1 + 0,22) \cdot 12 / 1750 = 401,55 \text{ грн.}$$

Сума витрат по статті "Поточний ремонт устаткування" дорівнює 3% від балансової вартості устаткування, що дорівнює $7875 \cdot 0.03 = 236.25$ грн.

Інші витрати складають 4% від суми попередніх:

$$(7875 + 722.17 + 401.55 + 236.25) \cdot 0.04 = 369.39 \text{ грн.}$$

Зведемо отримані дані щодо кошторису витрат на зміст і експлуатацію устаткування в табл. 6.6.

Таблиця 6.6 – Кошторис витрат на зміст і експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація устаткування	7875
Експлуатація устаткування	722.17
Зарплата обслуговуючого персоналу з відрахуваннями	401.55
Поточний ремонт устаткування	236.25
Інші витрати	369.39
Всього	9604.35

Експлуатаційні витрати на одну годину машинного періоду складають за формулою (6.9):

$$P_{\text{екс}} = \frac{S_{\text{ктс}}}{\text{Феф.обсл.}}. \quad (6.9)$$

Розрахуємо:

$$P_{\text{екс}} = 9604.37 / 1800 = 5.33 \text{ грн./год.}$$

Розробка і відладка програмного продукту займає 288 годин машинного часу (36 дні по 8 робочих годин), тобто $t_{\text{мо}} = 288$ год. Витрати на

оплату машинного часу склали:

$$C_{\text{мо}} = 5.33 \cdot 288 = 1535.04 \text{ грн.}$$

6.2.6 Накладні витрати

Накладні витрати складають 20 % до основної заробітної плати розробників – 3650 грн. Результати калькуляції кошторисної вартості робіт по розробці програми і ціна програмного продукту наведені у табл. 6.7.

Таблиця 6.7 – Калькуляція кошторисної вартості робіт по розробці програми і ціна програмного продукту

Найменування статей витрат	Сума, грн.	Питома вага до підсумку, %
Основна заробітна плата	18250	59.72
Додаткова заробітна плата	1825	5.97
Соціальні відрахування	4416.5	14.45
Витрати на матеріали	880	2.88
Витрати на оплату машинного часу	1535.04	5.02
Накладні витрати	3650	11.95
Разом витрати	30556.54	100
Прибуток (30%)	9166.96	—
Всього	39723.50	—
Податок на додану вартість (20%)	7944.70	—
Підсумок	47668.20	—

6.3 Оцінка економічної ефективності

Витрати на розробку програми складають 47668.20 гривень. Очікується продати 5000 копій за мінімальною ціною 25 гривень за копію. З

урахуванням ризику 50 % надходження від продажів складуть 62500 гривень. Це значно перевищує ціну розробки (на 14831.80 грн.).

Таким чином, можна зробити висновок про економічну доцільність розробки даного програмного забезпечення.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

У ході виконання дипломної кваліфікаційної роботи магістра досліджуються та програмно реалізуються методи керування спілкуванням користувачів, що зумовлює необхідність наявності робочого місця з персональним комп'ютером (ПК) та додатковим електронним обладнанням у приміщенні обчислювального центру (ОЦ).

На операторів ЕОМ, програмістів та інших робітників, що пов'язані з обладнанням ОЦ, у процесі роботи, за кваліфікацією ГОСТ 12.0.003-74 «Система стандартів безпеки праці. Небезпечні та шкідливі виробничі фактори. Класифікація» [22], впливають такі небезпечні та шкідливі виробничі чинники як:

- ураження електричним струмом у результаті порушення правил електробезпеки, несправності електрообладнання, відсутності захисного заземлення;
- вплив електромагнітних полів від монітору комп'ютера, що призводить до підвищеного стомлення, зниження працездатності та профзахворювання;
- розумове та емоційне перевантаження внаслідок недотримання змін режимів праці та відпочинку, що визиває зниження працездатності та стомлення [22].

7.2 Заходи з забезпечення безпеки

При роботі у ОЦ головним небезпечним чинником є вплив на персонал електричного струму.

За ступенем небезпеки ураження електричним струмом, відповідно до «Правил улаштування електроустановок» приміщення ОЦ відноситься до категорії приміщень без підвищеної небезпеки [23].

Заходи по забезпеченню електробезпеки розроблено у відповідності до ГОСТ 12.1.019-79 Система стандартів безпеки праці «Електробезпека. Загальні вимоги і номенклатура видів захисту» [24] і ГОСТ 12.1.030-81 Система стандартів безпеки праці «Електробезпека. Захисне заземлення, занулення» [25]: за способом захисту людини від ураження електричним струмом – відеотермінали, ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ відповідають першому класу захисту та заземлені відповідно до Державного нормативного акту про охорону праці 0.00-1.32-01 «Правила улаштування електроустановок. Електрообладнання спеціальних установок» [23]-[26].

Для забезпечення електробезпеки у ОЦ використано захисне відключення за ДСТУ 12.2.007- 75 «Захисне заземлення» [27].

В якості заходів захисту запропоновано:

- кілька разів на протязі робочого дня мити руки і обличчя водою, а після закінчення роботи вимити руки й лице з милом;
- щоденно протирати екран монітора, клавіатуру, пристрій «миша» та, якщо є, приєднаний фільтр антистатичною серветкою [28]-[29].

7.3 Заходи з виробничої санітарії і гігієни праці

Так як робота з обчислювальною технікою є малорухомою і створює велике навантаження на очі та кисті рук, операторам для підтримки працездатності необхідно керуватися Державними санітарними правилами і нормами 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», при роботі за дисплеєм у режимі праці і відпочинку передбачаються через кожні 40-45 хвилин 3-5 хвилинні перерви на відпочинок та гімнастичні вправи [30].

Виробниче середовище в приміщеннях ОЦ сполучають в собі раціональні архітектурно-планувальні рішення, найбільш оптимальні

санітарно-гігієнічні умови, науково обґрунтоване кольорове фарбування, у приміщенні ОЦ на робочих місцях з комп'ютерною технікою (КТ) нормалізація параметрів повітряного середовища забезпечується згідно до вимог Державних санітарних норм 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» і ГОСТ 12.1.005-88 «Система стандартів безпеки праці [31]. Загальні санітарно-гігієнічні вимоги до повітря робочої зони», що наведено в табл. 7.1.

Таблиця 7.1 Норми мікроклімату для приміщень з КТ

Пора року	Категорія робіт	Температура повітря, °С, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодна	легка -1а	22-24	40-60	0.1
	легка -1б	21-23	40-60	0.1
Тепла	легка -1а	23-25	40-60	0.1
	легка -1б	22-24	40-60	0.2

Рівні позитивних і негативних іонів у повітрі приміщення ОЦ відповідають санітарно-гігієнічним нормам 2152-80 «Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень», що наведені в табл. 7.2 [28].

Таблиця 7.2 – Рівні іонізації повітря приміщень з КТ

Рівні іонізації повітря	Кількість іонів в 1 см ³ повітря	
	n+	n-
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально допустимі	50000	50000

Природне освітлення здійснюється через віконні прорізи, при цьому коефіцієнт природної освітленості не нижче 1.5% за Державними будівельними нормами В.2.5-28-2006, робочі місця у приміщенні розташовані подалі від вікон таким чином, щоб сонячне світло не потрапляло на екран монітору, вікна оснащені сонцезахисною плівкою з металізованим покриттям [32].

Розрахуємо систему загального рівномірного штучного освітлення в адміністративному приміщенні [33].

Приймаємо розміри приміщення: довжина $A = 11$ м, ширина $B = 7$ м, висота $H = 5$ м. Висота робочої поверхні складає $h = 0.7$ м.

Коефіцієнти відбиття поверхонь приміщення: колір стелі, стін - сірий, незначне пиловиділення, $c=50\%$; $st=30\%$; $p=10\%$. Розряд зорової роботи – III б.

Згідно з Державними будівельними нормами В.2.5-28-2006 «Природне і штучне освітлення. Норми проектування» рівень нормованого загального освітлення в приміщенні нормуються як $E_n = 200$ Лк.

Обираємо світильник типу ЛПП, призначений для приміщень висотою до 6 м. $IP=65$ – ступінь захисту світильника, $[L/h] = 1.4$ – числове значення коефіцієнта світильника.

Коефіцієнт запасу k_z враховує зниження рівня освітленості з часом в результаті забруднення та старіння ламп, світильників і поверхонь приміщення, приймається в залежності від виробничих умов. Приймаємо $k_z = 1.6$.

Коефіцієнт нерівномірності (мінімального) освітлення z (відношення середньої освітленості до мінімальної освітленості), для люмінесцентних ламп низького тиску, як правило дорівнює $z = 1.1$.

Розрахуємо кількість рядів світильників у приміщенні за формулою (7.1) та отримаємо результат:

$$N_p = \frac{B}{(H-h_p) \cdot [L/h]} = \frac{7}{(5-0.7) \cdot 1.4} = 1.16 \approx 2 \text{ ряди світильників.} \quad (7.1)$$

Визначимо максимально припустиму відстань між рядами світильників за формулою (7.2):

$$L_{\max} = \frac{B}{N_p}, \quad (7.2)$$

де B – ширина приміщення;

N_p – кількість рядів.

Розрахуємо:

$$L_{\max} = \frac{7}{2} = 3.5 \text{ м.}$$

Висота підвісу світильника над робочою поверхнею обчислюється за формулою (7.3):

$$h = \frac{L_{\max}}{[L/h]}. \quad (7.3)$$

Розрахуємо:

$$h = \frac{3.5}{1.4} = 2.5 \text{ м.}$$

Знайдемо висоту звисання світильника від стелі за формулою (7.4):

$$h_3 = H - h_p - h, \quad (7.4)$$

де H – висота приміщення;

h_p – висота робочої поверхні;

h – висота підвісу світильника над робочою поверхнею.

Розрахуємо:

$$h_3 = 5 - 0.7 - 2.5 = 1.8 \text{ м.}$$

Чисельне значення індексу приміщення визначаємо за рівнянням (7.5):

$$i = \frac{A \cdot B}{h \cdot (A+B)}, \quad (7.5)$$

де A – довжина приміщення, м;

B – ширина приміщення, м;

h – висота розміщення світильників над робочою поверхнею, м.

Розрахуємо:

$$i = \frac{11 \cdot 7}{2.5 \cdot (11+7)} = 1.71.$$

Визначаємо сумарний світловий потік освітлювальної установки у даному виробничому приміщенні за формулою (7.6):

$$\Phi_{\Sigma} = \frac{E_n \cdot S \cdot k_3 \cdot z}{\eta}, \quad (7.6)$$

де Φ_{Σ} – розрахункове значення сумарного світлового потоку у приміщенні, лм;

E_n – нормоване значення освітленості, лк;

S – площа освітлюваної поверхні, м²;

k_3 – коефіцієнт запасу;

z – коефіцієнт нерівномірності (мінімальної) освітленості;

η – коефіцієнт використання світлового потоку.

Розрахуємо:

$$\Phi_{\Sigma} = \frac{200 \cdot 11 \cdot 7 \cdot 1.6 \cdot 1.1}{0.54} = 50193 \text{ лм.}$$

Визначаємо умовну загальну кількість світильників у приміщенні за формулою (7.7):

$$N^* = A \cdot \frac{B}{L_{\max}^2}. \quad (7.7)$$

Розрахуємо:

$$N^* = \frac{11 \cdot 7}{3.5 \cdot 3.5} = 6.3 \approx 8 \text{ шт.}$$

Розрахуємо світловий потік умовного джерела світла за формулою (7.8):

$$\Phi_{\text{л}}^* = \frac{\Phi_{\Sigma}}{N_{\text{л}}}, \text{ лм,} \quad (7.8)$$

де Φ_{Σ} – розрахункове значення сумарного світлового потоку у приміщенні, лм;

$N_{\text{л}}$ – загальна кількість ламп у приміщенні, шт.

Загальна кількість ламп у приміщенні визначається за формулою (7.9):

$$N_{\text{л}} = N^* \cdot n, \quad (7.9)$$

де n – кількість ламп у світильнику (в ЛПП світильнику знаходиться 2 лампи).

Розрахуємо:

$$N_{\text{л}} = 8 \cdot 2 = 16 \text{ шт.}$$

Тоді,

$$\Phi_{\text{л}} = \frac{50193}{16} = 3137 \text{ лм.}$$

Обираємо лампу ЛБ із $P_{\text{л}}=40$ Вт, $\Phi_{\text{л}} = 3200$ лм. Розрахунки виконаємо за формулою (7.10):

$$m = \frac{\Phi_{\text{л}}^*}{\Phi_{\text{л}}}. \quad (7.10)$$

Відповідно:

$$m = \frac{3137}{3200} = 0.98.$$

Визначаємо оптимальну (фактичну) кількість світильників у приміщенні за формулою (7.8).

Фактична кількість ламп у приміщенні розраховується за формулою (7.11):

$$N_{\text{фл}} = N_{\text{ф}} \cdot n. \quad (7.11)$$

Розрахуємо:

$$N_{\text{фл}} = 8 \cdot 2 = 16 \text{ шт.}$$

Визначаємо загальну розрахункову освітленість E_p у приміщенні, що створюється при застосуванні стандартних ламп, за формулою (7.12):

$$E_p = \frac{\Phi_{\text{л}} \cdot N_{\text{фл}} \cdot \eta}{S \cdot k_3 \cdot z}. \quad (7.12)$$

Розрахуємо:

$$E_p = \frac{3200 \cdot 16 \cdot 0.54}{11 \cdot 7 \cdot 1.6 \cdot 1.1} = 204 \text{ лк.}$$

Тобто виконується умова $E_p = (-10\% \dots +20\%) \cdot E_n$, лк.

Розрахуємо загальну потужність освітлювальної установки за формулою (7.13):

$$P_{\Sigma} = N_{\text{фл}} \cdot P_{\text{л}}, \quad (7.13)$$

де $P_{\text{л}}$ – потужність вибраної стандартної лампи.

Розрахуємо:

$$P_{\Sigma} = 16 \cdot 40 = 640 \text{ Вт.}$$

Розробляємо схему розташування світильників за умови $A = 11$ м, $B = 7$ м, $N_{\text{св}}=8$ шт., $L_{\text{max}} = 3.5$ м, довжина світильника $L_{\text{св}}= 1.27$ м.

Для захисту від впливу шуму використано організаційні та технічні заходи відповідно до Державних санітарних норм 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку [34].

Джерелами шуму при роботі з ПК є:

- жорсткий диск;
- вентилятор блока живлення мережі;
- вентилятор, розташований на процесорі;
- швидкісні CD-ROM;
- механічні сканери;
- пересувні механічні частини принтера [28].

Для зниження негативного впливу електромагнітного випромінювання згідно Державних санітарних норм і правил № 239-96

«Державні санітарні норми і правила захисту населення від впливу електромагнітних випромінювань» здійснено:

- використання на робочому місці відеотерміналів, що відповідають сучасним вимогам захисту від випромінювань;
- не переобтяжувати приміщення значною кількістю робочих місць з відеотерміналами;
- захист працюючих віддаленням робочого місця від джерела опромінення [35].

7.4 Заходи з пожежної безпеки

В приміщенні ОЦ джерелом загоряння можуть бути електронні схеми ЕОМ, прилади для технічного обслуговування, електроустаткування, кондиціонери повітря, де за рахунок різних порушень відбувається перегрів тих чи інших елементів, електричні іскри, дуга, що можуть визвати загоряння горючих матеріалів, дерев'яні та пластикові стільці та столи, папір, перевантаження мережі живлення, людські чинники тощо [28].

Прийнято наступні рішення:

- клас пожежі для приміщень ОЦ встановлюється згідно Державних будівельних норм В.1.1.7-2002 «Захист від пожежі. Пожежна безпека об'єктів будівництва» – пожежі твердих речовин, переважно органічного походження, горіння яких супроводжується тлінням (деревина, пластмаси, папір) – і визначається як клас А;
- категорія приміщення ОЦ за вибухопожежною небезпекою визначається згідно Нормативно-правових актів з пожежної безпеки Б.03.002-2007 «Норми визначення категорії приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою» – і визначається як пожежонебезпечність адмінприміщень [36]-[37].

7.5 Заходи з забезпечення безпеки у надзвичайних ситуаціях

Для проведення рятувальних та невідкладних аварійних робіт необхідно:

- завчасно спланувати дії формувань цивільної оборони як при загрозі нападу так і під час проведення рятувальних та невідкладних аварійних робіт;
- перевірити та уточнити плани цивільної оборони на навчаннях, що проводяться на об'єктах;
- вивести в найкоротші строки формування цивільної оборони, створені в містах, в заміську зону, розмістити їх в завчасно намічених районах й привести в готовність до проведення рятувальних робіт;
- організувати управління та керівництво формуваннями цивільної оборони при проведенні рятувальних робіт [34].

ВИСНОВКИ

Розглянуто проблеми сучасних засобів комунікації відносно корпоративної сфери, правила поведінки у корпоративному месенджері.

Виконано огляд програмних засобів, які можуть використовуватися для підтримки комунікації користувачів у корпоративній мережі, зокрема Slack та MyChat. Slack є глобальним рішенням, призначеним для корпоративної мережі, а MyChat локальним.

Проаналізовано методи інтелектуального аналізу даних, які можуть використовуватися для фільтрації та обробки тексту. Проаналізовано метод опорних векторів. Розроблено власний метод керування спілкуванням користувачів.

Наукова новизна роботи в розробленому методі, який базується на застосуванні фільтрації повідомлень шляхом визначеного фільтру слів, словосполучень та конструкцій речень та автоматичному аналізі цензурованих повідомлень, обчисленні статистики кожного користувача в кожен період часу, винесенні рекомендованого рішення щодо користувача на основі застосування методу опорних векторів для класифікації користувачів в поточний момент.

Визначено функції, які повинна виконувати програма. Створено діаграми прецедентів для моделювання функціональних вимог для роботи користувача та адміністратора з програмою. Обрано мову програмування Java для реалізації програми. Системою керування базами даних обрано MySQL. Визначено набір таблиць, їх структуру та побудовано структурну схему бази даних.

Реалізовано програмне забезпечення, яке дозволяє виконувати приватне і відкрите спілкування, створювати групи для спілкування, обмежувати доступ до спілкування та автоматизовано рекомендувати рішення щодо цього.

Усі завдання, поставлені на роботу, виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Корпоративний месенджер – види, можливості і сфери застосування [Електрон. ресурс]. – Режим доступу : <https://dlg.im/ru/blog/korporativnyi-miessindzhier-vidy-vozmozhnosti-i-sfiery-primieneniia/>.
2. Кожемякина, Е. Почему работодателям пора регулировать общение сотрудников в мессенджерах [Електрон. ресурс] / Е. Кожемякина. – Режим доступу : <https://www.vedomosti.ru/management/blogs/2019/05/05/800701-pochemu>.
3. Обзор профессионального корпоративного чата MyChat [Електрон. ресурс]. – Режим доступу : <https://www.ixbt.com/soft/mychat.shtml>.
4. 12 правил делового общения в мессенджерах [Електрон. ресурс]. – Режим доступу : <https://dlg.im/ru/blog/12-pravil-delovogo-obshenia-v-messengerah/>.
5. Онлайн-этикет [Електрон. ресурс] : как общаться в рабочих чатах, чтобы не испортить карьеру. – Режим доступу : <https://www.marieclaire.ru/karera/onlayn-etiket-kak-obschatsya-v-rabochih-chatah-chtobyi-ne-isportit-kareru/>.
6. Welcome to your new HQ | Slack [Electronic resource]. – Access mode : <https://slack.com/intl/en-ua/>.
7. Белоногова, Н.Н. Корпоративный мессенджер Slack [Електрон. ресурс] : обзор «убийцы Skype и внутрикорпоративной электронной почты» / Н.Н. Белоногова. – Режим доступу : <https://www.kp.ru/guide/slack.html>.
8. Slack – корпоративный мессенджер [Електрон. ресурс]. – Режим доступу : <https://trello-gid.ru/slack-korporativnyj-messendzher/>.
9. Обзор профессионального корпоративного чата MyChat [Електрон. ресурс]. – Режим доступу : <https://www.ixbt.com/soft/mychat.shtml>.
10. Как использовать интеллектуальный анализ текста [Електрон. ресурс]. – Режим доступу : <https://www.waveaccess.ru/blog/2020/january/9/>

интеллектуальный-анализ-текста-как-использовать.aspx.

11. Oard, D.W. The State of the Art in Text Filtering [Text] / D.W. Oard // User Modeling and User-Adapted Interaction. – 1997. – № 7. – Pp. 141-178.
12. Жерон, О. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow [Текст] : концепции, инструменты и техники для создания интеллектуальных систем / О. Жерон. – М. : Вильямс, 2020. – 688 с.
13. Машинное обучение [Электрон. ресурс] : как метод опорных векторов может быть использован в трейдинге. – Режим доступа : <https://www.mql5.com/ru/articles/584>.
14. Методы классификации и прогнозирования. Метод опорных векторов. Метод "ближайшего соседа". Байесовская классификация [Электрон. ресурс]. – Режим доступа : <https://intuit.ru/studies/courses/6/6/lecture/176>.
15. Вьюгин, В. Математические основы теории машинного обучения и прогнозирования [Текст] / В. Вьюгин. – М. : МЦМНО, 2013. – 390 с.
16. Албахари, Дж. С# 6.0. Справочник. Полное описание языка [Текст] / Дж. Албахари, Б. Албахари. – М. : Вильямс, 2018. – 1040 с.
17. Скит, Дж. С# для профессионалов [Текст] : тонкости программирования / Дж. Скит. – М. : Вильямс, 2014. – 608 с.
18. Simran, K.A. C# vs Java [Electronic resource] : Differences you should Know / K.A. Simran. – Access mode : <https://hackr.io/blog/c-sharp-vs-java>.
19. Васвани, В. MySQL [Текст] : использование и администрирование / В. Васвани. – М. : Питер, 2011. – 368 с.
20. MySQL Workbench Tutorial & MySQL Introduction [Electronic resource]. – Access mode : <https://www.guru99.com/introduction-to-mysql-workbench.html>.
21. Методичні вказівки по економічному обґрунтуванню дипломних проектів для студентів спеціальностей 8.05010301 «Програмне забезпечення автоматизованих систем» та 8.05010201 «Інформаційні технології проектування» для усіх форм навчання [Текст] / В.В. Остапенко,

Є.М. Сердюк. – Запоріжжя : ЗНТУ, 2015. – 22 с.

22. ГОСТ 12.0.003-74 Система стандартов безопасности труда. Опасные и вредные производственные факторы. Классификация [Текст]. – М. : ИПК Издательство стандартов, 1974. – 8 с.

23. Правила улаштування електроустановок [Текст] : ПУЕ-2017. – На заміну ПУЕ-86 ; чинний з 2017-08-21. – К. : Міненерговугілля України, 2017. – 617 с.

24. ГОСТ 12.1.019-79 Система стандартов безопасности труда. Электробезопасность. Общие требования и номенклатура видов защиты [Текст]. – М. : ИПК Издательство стандартов, 1979. – 7 с.

25. ГОСТ 12.1.030-81 Электробезопасность. Защитное заземление. Зануление [Текст]. – М. : ИПК Издательство стандартов, 1981. – 8 с.

26. ДСТУ Б В.2.5-82:2016 Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом [Текст]. – Київ : УкрНДНЦ, 2016. – 109 с.

27. Батлук, В. А. Охорона праці. Навчальний посібник. Третє видання, доповнене [Текст] / В. А. Батлук, М. П. Кулик, Р. А. Яцюк. – Львів : Видавництво Львівської політехніки, 2011. – 388 с.

28. Жидецький, В. Ц. Основи охорони праці [Текст] : підручник / В. Ц. Жидецький. – 5-те вид., доп. – К. : Знання, 2014. – 373 с.

29. Основи охорони праці [Текст] : підручник / В. В. Березуцький, Т. С. Бондаренко, Г. Г. Валенко та ін.; за ред. проф. В. В. Березуцького. – Х. : Факт, 2005. – 327 с.

30. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. НПАОП 0.00-7.15-18. Державні санітарні правила і норми [Текст]. – Харків : Форт, 2018. – 48 с.

31. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень [Текст]. – Харків : Форт, 2000. – 24 с.

32. ДБН В.2.5-28:2018. Природне та штучне освітлення. Посібник [Текст] : проблеми природного і штучного освітлення. – К. : Етін, 2019. –

180 с.

33. Методичні вказівки до лабораторного заняття «Розрахунок загального рівномірного штучного освітлення виробничих приміщень» з дисципліни «Цивільний захист і охорона праці в галузі» [Текст] : для студентів усіх спеціальностей та форм навчання / В.І. Шмирко, О.В. Коробко, Ю.І. Троян. – Запоріжжя : каф. ОПіНС. НУ «Запорізька політехніка», 2020. – 36 с.

34. ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» [Текст]. – К. : МОЗ України, 1999. – 24 с.

35. Березуцький, В.В. Небезпечні виробничі ризики та надійність [Текст] : навч. посібник / В.В. Березуцький, М.І. Адаменко ; Харківський політехнічний ін-т, нац. техн. ун-т. – Харків : ФОП Панов А.М., 2016. – 385 с.

36. Пожежна безпека об'єктів будівництва. Загальні вимоги [Текст] : ДБН В.1.1-7:2016. – На заміну ДБН В.1.1.7-2002 ; чинний від 2017-06-01. – К. : Мінрегіон України, 2017. – 47 с. – (Державні будівельні норми).

37. Стеблюк, М. І. Цивільна оборона та цивільний захист [Текст] : навч. посіб. для вузів / М. І. Стеблюк. – К. : Знання, 2013. – 487 с.

ДОДАТОК А
Технічне завдання

Вступ

Виконання роботи присвячено розробленню програмного забезпечення керування спілкуванням користувачів для забезпечення дисциплінування спілкування працівників у робочий час.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу магістра за темою “Дослідження та програмна реалізація методів керування спілкуванням користувачів. Research and Software Implementation of Methods for Managing User Communication”, затверджене наказом № 301 від 3 листопада 2020 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Розробка призначена для реалізації робочого спілкування в корпоративній мережі з забезпеченням можливості фільтрування спілкування для уникнення заборонених тем, а також образливих, лайливих слів.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Для забезпечення встановленої цілі програма повинна надавати можливість виконувати такі функції:

- створення акаунтів працівників;
- пошук акаунтів;
- авторизація акаунтів;
- відображення переліку приватних комунікацій;

- відкриття приватного спілкування;
- вилучення приватного спілкування;
- надсилання повідомлення в приватну комунікацію;
- додавання приватного спілкування;
- створення відділу;
- редагування даних акаунту;
- включення учасників до відділу;
- вилучення учасників з відділу;
- відображення переліку доступних комунікацій відділів;
- надсилання повідомлення в комунікацію всередині відділу;
- висловлення скарги про повідомлення;
- прогнозування статусу користувача;
- обмеження на доступ користувача;
- надсилання попередження користувачу.

A.3.2 Вимоги до надійності

Для забезпечення надійного функціонування, по-перше, має використовуватися сервер, який здатен обробити відповідну кількість звернень клієнтів, окрім того його робота має обов'язково підтримуватися протягом життєвого циклу, а по-друге, має забезпечуватися надійне мережне з'єднання.

A.3.3 Умови експлуатації

Для забезпечення експлуатації програми ніяких особливих вмінь не вимагається. І адміністратор, і користувачі повинні знати мінімальні правила роботи з комп'ютером.

А.3.4 Вимоги до складу та параметрів технічних засобів

Для роботи сервера мають забезпечуватися наступні мінімальні вимоги:

- процесор з 4 ядрами з частотою 2 ГГц і більше;
- обсяг оперативного запам'ятовуючого пристрою – не менше 8 Гб;
- вільний дисковий простір – 300 Гб;
- пропускна здатність мережного інтерфейсу – 1 Гбіт/с.

За такої мінімальної конфігурації передбачається можливість доступу щонайменше 50 користувачів.

А.3.5 Вимоги до транспортування та збереження

Збереження програми має виконуватися на сервері з постійним забезпеченням резервування. Окрім того остання версія програми має зберігатися на переносних носіях для забезпечення накопичення версій.

А.3.6 Вимоги до програмної документації

Програмна документація розміщується в додатках пояснювальної записки на роботу і повинна включати технічне завдання та текст програми. В тексті програми приводиться фрагмент виконаної програмної реалізації.

А.3.7 Вимоги до маркування та упакування

Якщо програма зберігається на переносному носії, то він має обов'язково зберігатися в чохлі, на якому має бути зафіксовано назву роботи.

A.4 Порядок контролю та приймання

Дипломна кваліфікаційна робота магістра після виконання має бути прийнята керівником та консультантами усіх відповідних розділів, зазначених у завданні на роботу, після чого подається до захисту після виконання всіх відповідних процедур.

Програма, що розробляється, повинна реалізовувати функції, визначені технічним завданням. Основні результати мають бути представлені на презентації.

ДОДАТОК Б
Текст програми

Б.1 Текст файла Waiter.java

```

package waiter;

import java.io.IOException;
import java.util.List;
import jadex.brnumge.IInternalAccess;
import java.net.ServerSocket;
import jadex.bdiv3.features.IBDIAgentFeature;
import java.net.Socket;
import participant.DialogServer;
import PDO.Communic;

public class Waiter extends Thread
{
    private IInternalAccess access = null;
    private int pnum;
    private List<Connection> connections;

    public List<Connection> getConnections ()
    {
        return connections;
    }

    public Waiter (int pnum, List<Connection> connections,
IInternalAccess access)
    {
        this.pnum = pnum;
        this.connections = connections;
        this.access = access;
    }

    @Override
    public void run ()
    {
        try {
            ServerSocket sS = new ServerSocket (pnum);
            while (true)
            {
                Socket cS = sS.accept ();

```

```

        Connection connection = new Connection
(this, cS);

        connection.start ();
    }
    } catch (IOException e) {
        System.err.println ("Не удалось запустить
сервер");
    }
}

    public void addCommunic (Connection connection, String
text, String addr, boolean flag)
    {
        Communic allinfo = new Communic (connection.getall
(),text, addr, flag);
        access.getComponentFeature
(IBDI-AgentFeature.class).dispatchTopLevelGoal (new
DialogServer.NewCommunic (allinfo));
    }

    public void addGroupCommunic (Connection connection,
String text, List<String> addrL, boolean flag)
    {
        GroupCommunic allinfo = new GroupCommunic
(connection.getall (),text, addrL, flag);
        access.getComponentFeature
(IBDI-AgentFeature.class).dispatchTopLevelGoal (new
DialogServer.NewGroupCommunic (allinfo));
    }

    public void addConnection (Connection connection)
    {
        access.getComponentFeature
(IBDI-AgentFeature.class).dispatchTopLevelGoal (new
DialogServer.AddAccount (connection));
    }

    public void deleteConnection (Connection connection)
    {
        connections.remove (connection);
    }
}

```

Б.2 Текст файла Connection.java

```
package connection;

import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.net.Socket;
import java.io.IOException;
import java.sql.SQLException;
import java.io.PrintWriter;

import participant.DialogServer;
import libs.ActionCreator;
import PDO.Account;

public class Connection extends Thread
{
    private Socket cS;
    private Waiter waiter;
    private int num = -1;
    private Account acc;
    public PrintWriter pw;
    public BufferedReader br;

    public Account getAcc ()
    {
        return acc;
    }

    public void setAcc (Account ac)
    {
        this.acc = ac;
    }

    public int getPNum ()
    {
        return num;
    }

    Connection (Waiter waiter, Socket cS)
    {
        this.cS = cS;
```

```

        this.waiter = waiter;
    }

    @Override
    public void run ()
    {
        try {
            CSobtainer ();
        } catch (IOException | SQLException sinfo) {
            try {
                if (this.acc!= null) {
                    noconnect ();
                }
                System.out.println (sinfo.getCommunic ());
            } catch (IOException e1) {
                System.out.println (cS.isConnected ());
            }
        }
    }

    private void CSobtainer () throws IOException,
SQLException
    {
        String row;
        String act1;
        pw = new PrintWriter (cS.getOutputStream (), true);
        br = new BufferedReader (new InputStreamReader
(cS.getInputStream ()));
        act1= br.readLine ();
        oneAction (act1);
        while ((row = br.readLine ()) && cS.isConnected)
        {
            oneAction (row);
        }
        if (pw)
        {
            pw.close ();
        }
        if (br)
        {
            br.close ();
        }
        if (cS)
        {

```

```

        cS.close ();
    }
}

private void oneAction (String act) throws SQLException
{
    String[] parts = act.split ("-", 2);
    System.out.println (act);
    if (parts.length != 2)
    {
        System.err.println ("Невірно задано
налаштування");
        return;
    }

    String a = parts [0];
    String[] param = parts[1].split ("=");

    for (int i = 0; i < param.length; i++)
    {
        param [i] = param [i].trim ();
    }

    switch (a) {
        case "auth":
            processAuth (param);
            break;

        case "communic":
            processCommunic (param);
            break;

        case "grcomm":
            processGroupCommunic (param);
            break;

        case "signup":
            processSignup (param);
            break;

        default:
            System.err.println ("Невідома команда");
            break;
    }
}

```

```

    }

    private void processAuth (String[] param) throws
SQLException {
        if (param.length != 2) {
            post ("Невірно задано налаштування
аутентифікації", 1);
            return;
        }
        String name = param [0];
        String passwd = param [1];
        this.num = Account.auth (name, passwd);
        waiter.addConnection (this);
    }

    private void processNoConn () throws IOException
    {
        waiter.deleteConnection (this);
        for (Connection connection : waiter.getConnections
    ())

        {
            if (connection.getNum () == this.getNum ())
            {
                cS.close ();
                return;
            }
        }
        post ("noconn", 2, this.acc.getNUM () + "");
        cS.close ();
    }

    private void processCommunic (String[] param)
    {
        if (param.length != 3)
        {
            System.out.println ("Невірно задано параметри
передачі повідомлення");
            post ("problem", 1);
            return;
        }
        String text = param[2];
        String addr = param [1];
        waiter.addCommunic (this, text, addr, false);
    }

```

```

private void processGroupCommunic (String[] param)
{
    if (param.length != 3)
    {
        System.out.println ("Невірно задано параметри
передачі повідомлення");
        post ("problem", 1);
        return;
    }
    String text = param[2];
    String addr = param [1];
    waiter.addGroupCommunic (this, text, addr, false);
}

private void processSignup (String[] param) throws
SQLException
{
    if (param.length != 1)
    {
        System.out.println ("Невірно задано параметри
реєстрації");
        post ("problem", 1);

        return;
    }

    if (!this.getAcc().IsReady ())
    {
        String problem = "Авторизуйтеся";
        waiter.addCommunic (this, problem, this.getAcc
().getAuth (), true);
        return;
    }
    String name = param[0];
    Account uacc = Account.getAccOnName (name);
    if (uacc)
    {
        String problem = "Невірно визначено акаунт";
        waiter.addCommunic (this, problem, this.getAcc
().getLogin (), true);
        return;
    }
}

```

```

        if (uacc.IsReady ())
        {
            String problem = "Акаунт вже авторизовано";
            waiter.addCommunic (this, problem, this.getAcc
().getLogin (), true);
            return;
        }

        uacc.signup ();
        String okinfo = "Акаунт зареєстровано";
        waiter.addCommunic (this, okinfo, this.getAcc
().getLogin (), true);
        waiter.addCommunic (this, okinfo, name, true);
    }

    public void Answer (String text)
    {
        pw.println (text);
    }

    public void post (String info, int mode, String param)
    {
        String act = ActionCreator.create (info, param);
        if (mode == 1)
        {
            Answer (text);
            return;
        }

        for (Connection connection : waiter.getConnections
())
        {
            if (mode == 5 && connection.getAcc ().getNum ()
== this.getAcc ().getNum ())
            {
                continue;
            } else
            {
                connection.Answer (text);
            }
        }
    }
}

```

```

        public void post (String act, String addr, String src,
String param)
        {
            String ans = ActionCreator.create (act, param);
            String servad = "";
            if (addr.equals (DialogServer.ServerAddress))
            {
                servad = src;
            } else if (src.equals (DialogServer.ServerAddress))
            {
                servad = addr;
            }

            if (servad)
            {
                for (Connection connection :
waiter.getConnections ())
                {
                    if (connection.getAcc ().getLogin
().equals (servad))
                    {
                        connection.Answer (ans);
                    }
                }

                return;
            }

            for (Connection connection : waiter.getConnections
())
            {
                if ( (connection.getAcc ().getLogin ().equals
(addr) || connection.getAcc ().getLogin ().equals (src)) &&
!src.equals (addr))
                {
                    connection.Answer (ans);
                }
            }
        }
    }

```

Б.3 Текст файлу Account.java

```

package PDO;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.*;

import participant.DialogServer;
import waiter.Waiter;
import PDO.Ban;
import PDO.Department;

public class Account {

    private int id;
    private String login;
    private String passwd;
    private String surname;
    private String phone;
    private String position;
    private float level;
    private int type;
    private ArrayList<Ban> bans = new ArrayList<>();;
    private ArrayList<Department> departments = new
ArrayList<>();;
    private ArrayList<PrivateMessage> pm = new
ArrayList<>();;
    private ArrayList<DepartmentMessage> dm = new
ArrayList<>();;
    private int authd = 0;

    public int getId() {
        return id;
    }

    public String getLogin() {
        return login;
    }

    public String getSurname() {

```

```

        return surname;
    }

    public String getPhone() {
        return phone;
    }

    public String getPosition() {
        return position;
    }

    public float getLevel() {
        return level;
    }

    public int getType() {
        return type;
    }

    public ArrayList<Ban> getBans() {
        return bans;
    }

    public ArrayList<Department> getDepartments() {
        return departments;
    }

    public ArrayList<PrivateMessage> getPMessages() {
        return pm;
    }

    public ArrayList<DepartmentMessage> getDMessages() {
        return dm;
    }

    public int getBansNum() {
        return bans.size();
    }

    public <Ban> getLastBan() {
        Collections.sort(bans,
Comparator.comparing(Ban::tomoment));
        return bans[0];
    }

```

```

        public long getLastBanTime() {
            Collections.sort(bans,
Comparator.comparing(Ban::tomoment));
            return bans[0].tomoment;
        }

        public long getLastBanPeriod () {
            long nowt = System.currentTimeMillis();
            Collections.sort(bans,
Comparator.comparing(Ban::tomoment));
            long per = nowt - bans[0].tomoment;
            return Math.round(per / 6e4f);
        }

        public int getMessagesNum() {
            return pm.size()+dm.size();
        }

        public int getCensoredMessagesNum() {
            int num = 0;
            for (int i = 0; i < pm.size(); i++) {
                if (pm.get(i).type==2)
                    num = num + 1;
            }
            for (int i = 0; i < dm.size(); i++) {
                if (dm.get(i).type==2)
                    num = num + 1;
            }
            return num;
        }

        public double getCensoredMessagesPerc() {
            return this.getCensoredMessagesNum()/this.
getMessagesNum*100;
        }

        public boolean IsReady () {
            return this.authd;
        }

        public boolean isProblemed() {
            return this.type == 2;
        }

```

```

        public boolean isBanned() {
            Collections.sort(bans,
Comparator.comparing(Ban::tomoment));
            long p1 = bans[bans.size()-1].frommoment;
            long p2 = bans[bans.size()-1].tomoment;
            long p3 = System.currentTimeMillis();
            if (p3>p1 && p3<p2)
                return true;
            else return false;
        }

        public void ban(long frommoment, long tomoment) {
            Ban b = new Ban();
            b.user = this.id;
            b.frommoment = frommoment;
            b.tomoment = tomoment;
            b.type = 2;
            bans.add(b);
        }

        public void unban() {
            Collections.sort(bans,
Comparator.comparing(Ban::tomoment));
            bans[bans.size()-1].tomoment =
System.currentTimeMillis();
            bans[bans.size()-1].type = 3;
        }

        public String getBanTime() {
            long p1 = bans[bans.size()-1].tomoment;
            long p2 = System.currentTimeMillis();
            long per = p1 - p2;
            int d1 = per / 86400;
            long t1 = per % 86400;
            int d2 = t1 / 3600;
            long t2 = t1 % 3600;
            int d3 = t2 / 60;
            int d4 = t2 % 60;
            return String.format("[%02d:%02d:%02d:%02d]", d1,
d2, d3, d4);
        }

        public String getDeps() {
            String d = "";

```

```

        for (int i = 0; i < departments.size(); i++) {
            d = d + departments.name + ", ";
        }
        return d;
    }

    public Account (int id, String login, String passwd,
String surname, String phone, String position, float level, int
type) {
        this.id = id;
        this.login = login;
        this.passwd = passwd;
        this.surname = surname;
        this.phone = phone;
        this.position = position;
        this.level = level;
        this.type = type;
    }

    public void setLogin(String l) {
        return this.login = l;
    }

    public void setSurname(String s) {
        return this.surname = s;
    }

    public void setPhone(String p) {
        return this.phone = p;
    }

    public void setPosition(String p) {
        return this.position = p;
    }

    public void setLevel(float l) {
        return this.level = l;
    }

    public void setType(int t) {
        return this.type = t;
    }

    public void setBans(ArrayList<Ban> b) {

```

```

        return this.bans = b;
    }

    public void setDepartments(ArrayList<Department> d) {
        return this.departments = d;
    }

    public void setPrivateMessages(ArrayList<PrivateMessage>
d) {
        return this.pm = d;
    }

    public void setDepartmentMessages
(ArrayList<DepartmentMessage> d) {
        return this.dm = d;
    }

    public void setPrivateMessages () {
        ResultSet alldata;

        try {
            alldata = db.query("SELECT * FROM
PrivateMessage WHERE from = " + this.getId());
            this.pm = new ListArray<PrivateMessage>();
            while (alldata.next()) {
                PrivateMessage p = new PrivateMessage();
                p.id = alldata.getInt("id");
                p.from = alldata.getInt("from");
                p.to = alldata.getInt("to");
                p.moment = alldata.getTimestamp
("moment");

                p.messtext =
alldata.getString("messtext");
                p.type = alldata.getInt("type");
            }
            this.pm.add(p);
        } catch (SQLException e) {
            sql.printStackTrace();
        }

    }

    public void setDepartmentMessages () {
        ResultSet alldata;

```

```

        try {
            alldata = db.query("SELECT * FROM
DepartmentMessage JOIN DepartmentAccounts WHERE
DepartmentAccounts.user = DepartmentMessage.ifrom AND
DepartmentMessage.user = " + this.getId());
            this.departments = new ListArray<Department>();
            while (alldata.next()) {
                PrivateMessage p = new PrivateMessage();
                p.id = alldata.getInt("id");
                p.from = alldata.getInt("from");
                p.moment = alldata.getTimestamp
("moment");

                p.messtext =
alldata.getString("messtext");
                p.type = alldata.getInt("type");
            }
            this.departments.add(p);
        } catch (SQLException e) {
            sql.printStackTrace();
        }
    }

    public void setBans () {
        ResultSet alldata;

        try {
            alldata = db.query("SELECT * FROM Ban WHERE
user = " + this.getId());
            this.bans = new ListArray<Ban>();
            while (alldata.next()) {
                Ban p = new Ban();
                p.id = alldata.getInt("id");
                p.user = alldata.getInt("user");
                p.frommoment = alldata.getTimestamp
("frommoment");

                p.tomoment = alldata.getTimestamp
("tomoment");

                p.type = alldata.getInt("type");
            }
            this.bans.add(p);
        } catch (SQLException e) {
            sql.printStackTrace();
        }
    }

```

```

    }

    public void setDepartments () {
        ResultSet alldata;

        try {
            alldata = db.query("SELECT * FROM Department
JOIN DepartmentAccounts WHERE DepartmentAccounts.dep =
Department.id AND DepartmentAccounts.user = " + this.getId());
            this.departments = new ListArray<Department>();
            while (alldata.next()) {
                Department p = new Department ();
                p.id = alldata.getInt("id");
                p.name = alldata.getString("name");
                p.company = alldata.getString ("company");
                p.address = alldata.getString ("address");
                p.phone = alldata.getString ("phone");
            }
            this.departments.add(p);
        } catch (SQLException e) {
            sql.printStackTrace();
        }
    }

    public Account (String info) {
        this.deserialize(info);
    }

    public void save() {
        String comm = "UPDATE Account SET surname = ? ,
phone = ? , position = ? , level = ? , type = ? WHERE id = ?";

        Connection dbc = db.connect();
        PreparedStatement pcomm;

        try {
            comm = dbc.prepareStatement(comm);
            comm.setString (1, this.surname);
            comm.setString (2, this.phone);
            comm.setString (3, this.position);
            comm.setFloat(4, this.level);
            comm.setInt(5, this.type);
            comm.executeUpdate();
        } catch (SQLException sql) {

```

```

        sql.printStackTrace();
    }

}

public void update() {
    ResultSet alldata;

    try {
        alldata = db.query("SELECT * FROM Account WHERE
id = " + this.getId());

        this.id = alldata.getInt("id");
        this.login = alldata.getString("login");
        this.passwd = alldata.getString ("password");
        this.surname = alldata.getString ("surname");
        this.phone = alldata.getString ("phone");
        this.position = alldata.getString ("position");
        this.level = alldata.getFloat("level");
        this.type = alldata.getInt("type");
    } catch (SQLException e) {
        sql.printStackTrace();
    }

}

public String serialize() {
    return String.join("=", String.valueOf(this.ID),
this.login, String.valueOf(this.passwd),
String.valueOf(this.surname), String.valueOf(this.phone),
String.valueOf(this.position), String.valueOf(this.level),
String.valueOf(this.type));
}

public void deserialize(String info) {
    String[] splitted = info.split("=");

    if(splitted.length != 7) return;
    this.id = Integer.parseInt(splitted [0]);
    this.login = splitted [1];
    this.passwd = splitted [2];
    this.surname = splitted [3];
    this.phone = splitted [4];
    this.position = splitted [5];
}

```

```

        this.level = Float.parseFloat(splitted [6]);
        this.type = Integer.parseInt(splitted [7]);
    }

    public static int auth(String name, String passwd) throws
SQLException {
        ResultSet alldata = db.query("SELECT COUNT(*) AS
sumn, id FROM Account WHERE login = \"" + name + "\" AND password =
\"" + passwd + "\"");

        if(alldata.getInt("sumn") == 1) {
            return alldata.getInt("id");
        }

        return -1;
    }

    public static Account getAccOnId (int id) throws
SQLException {
        ResultSet alldata = db.query("SELECT COUNT(*) AS
sumn, * FROM Account WHERE id = " + id);

        if(rs.getInt("sumn ") == 1) {
            return Account.create(alldata);
        }

        return null;
    }

    public static Account getAccOnName(String name) throws
SQLException {
        ResultSet alldata = db.query("SELECT COUNT(*) AS
sumn, * FROM Account WHERE login = \"" + name + "\"");

        if(alldata.getInt("sumn") == 1) {
            return Account.new(alldata);
        }

        return null;
    }

    public static Account newAcc(ResultSet alldata) throws
SQLException {
        int id = alldata.getInt("id");

```

```

        String login = alldata.getString("login");
        String passwd = alldata.getString("passwd");
        String surname = alldata.getString("surname");
        String phone = alldata.getString("phone");
        String position = alldata.getString("position");
        float level = alldata.getFloat("level");
        int type = alldata.getInt ("type");

        return new Account (ID, login, passwd, surname,
phone, position, level, type);
    }

    public static int signUp(String name, String passwd)
throws SQLException {
        if(Account.getAccOnName (name)) {
            return -1;
        }

        String comm = "INSERT INTO Account (name, passwd)
VALUES(?, ?)";

        Connection cdb = db.connect();
        PreparedStatement pcomm;

        try {
            pcomm = conn.prepareStatement(cdb);

            pcomm.setString(1, name);
            pcomm.setString(2, passwd);
            pcomm.executeUpdate();

            return Account.getAccOnName (name).getID();
        } catch (SQLException sql) {
            sql.printStackTrace();
        }

        return -1;
    }
}

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота магістра

**ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
МЕТОДІВ КЕРУВАННЯ СПІЛКУВАННЯМ
КОРИСТУВАЧІВ**

Виконав

студент групи КНТ-119м

Р. В. Лобода

Керівник

к.т.н.

Є. О. Гофман

2020

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес керування спілкуванням користувачів.

Предмет дослідження – програмне забезпечення корпоративної комунікації користувачів.

Мета роботи – розроблення програмного забезпечення керування спілкуванням користувачів для забезпечення дисциплінування спілкування працівників у робочий час.

Рисунок В.2 – Слайд 2

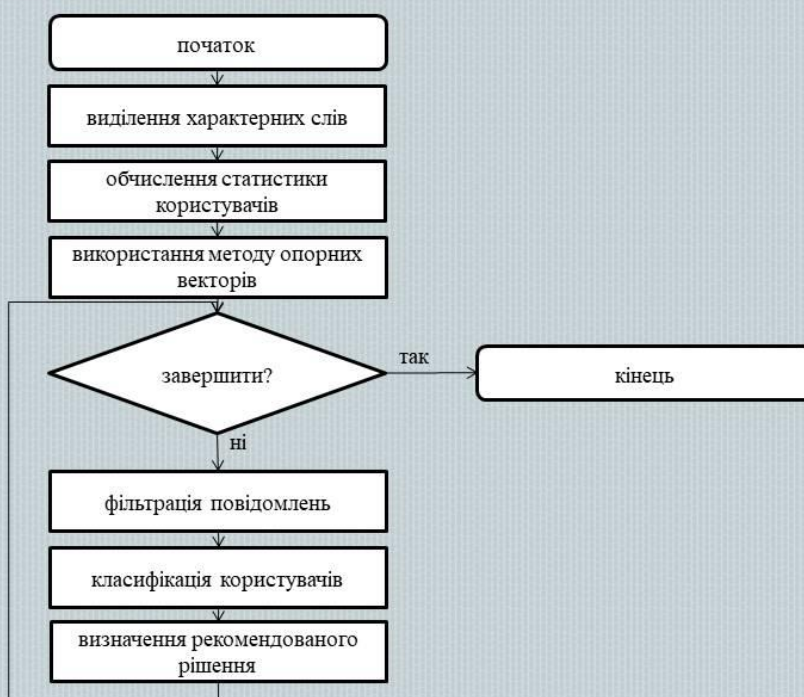
Завдання

- ✓ дослідження методів керування спілкуванням користувачів;
- ✓ проєктування структури програмного забезпечення;
- ✓ розробка програмного забезпечення;
- ✓ тестування та відлагодження програми;
- ✓ економічне обґрунтування доцільності розробки та забезпечення засходів охорони праці та безпеки в надзвичайних ситуаціях

3

Рисунок В.3 – Слайд 3

Метод керування спілкуванням користувачів



4

Рисунок В.4 – Слайд 4

Діаграми прецедентів



5

Рисунок В.5 – Слайд 5

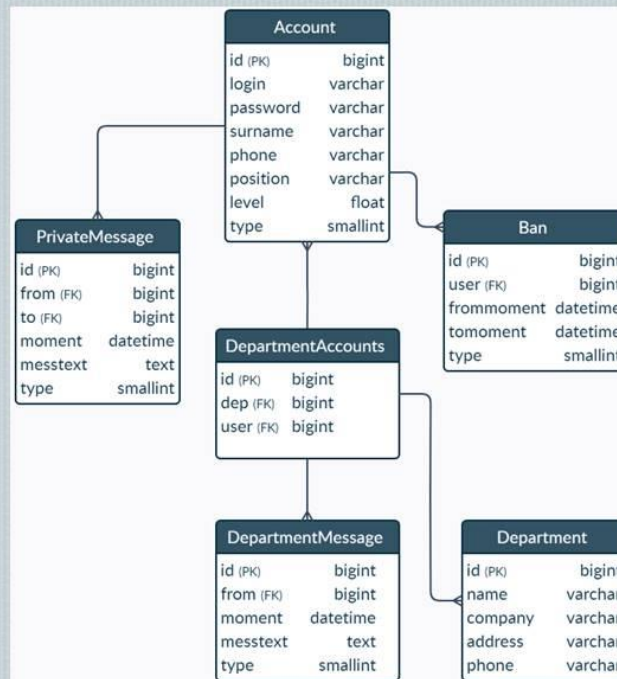
Порівняння мов програмування

Критерій	C#	Java
Об'єктно-орієнтована парадигма	+	+
Безкоштовне рішення	-	+
Створення графічного інтерфейсу	+	+
Засоби роботи в мережі	+	+
Робота з вільними системами керування базами даних	+	+
Кросплатформність	-	+

6

Рисунок В.6 – Слайд 6

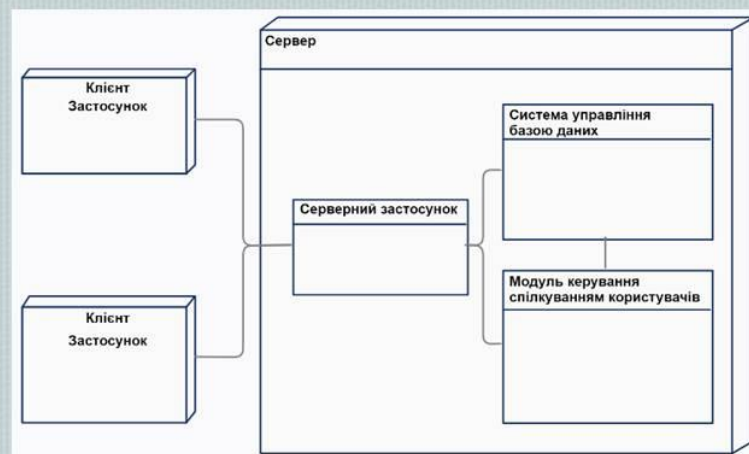
Структура бази даних



7

Рисунок В.7 – Слайд 7

Діаграма розгортання



8

Рисунок В.8 – Слайд 8

Схема взаємодії між сервером та клієнтами

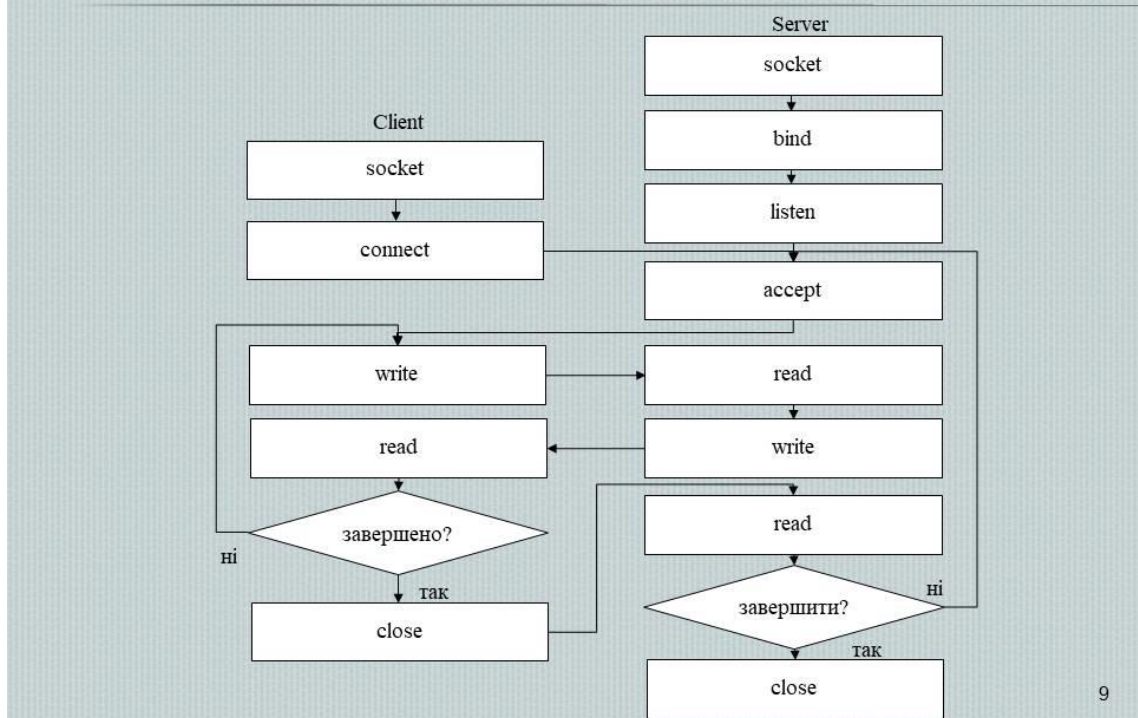


Рисунок В.9 – Слайд 9

Приватне та групове спілкування в корпоративному чаті програми

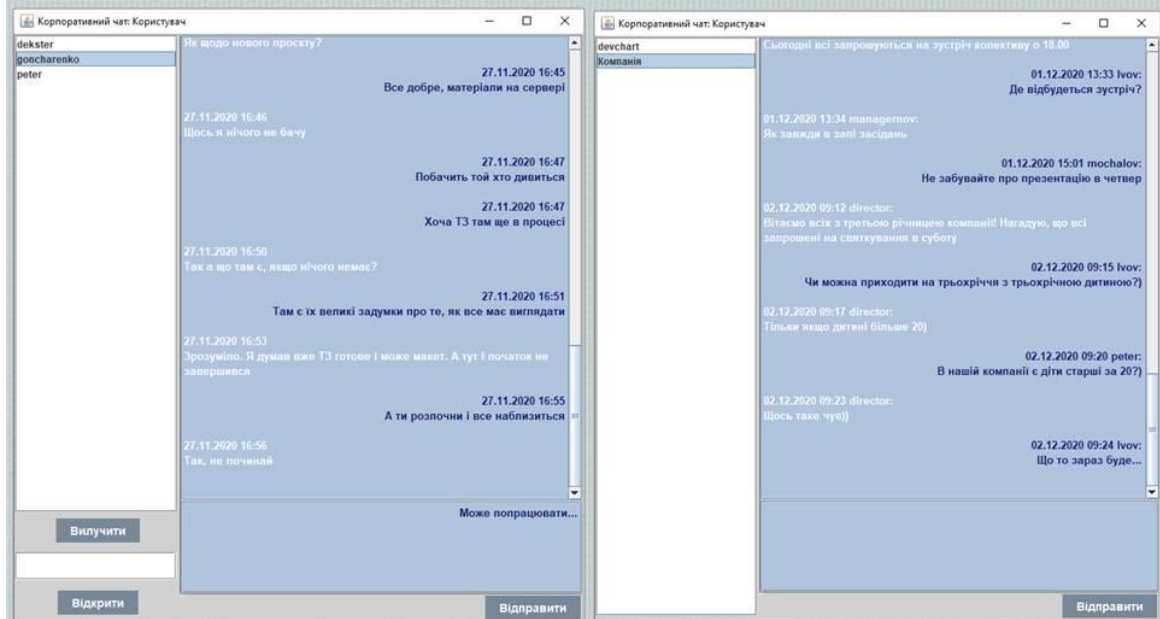
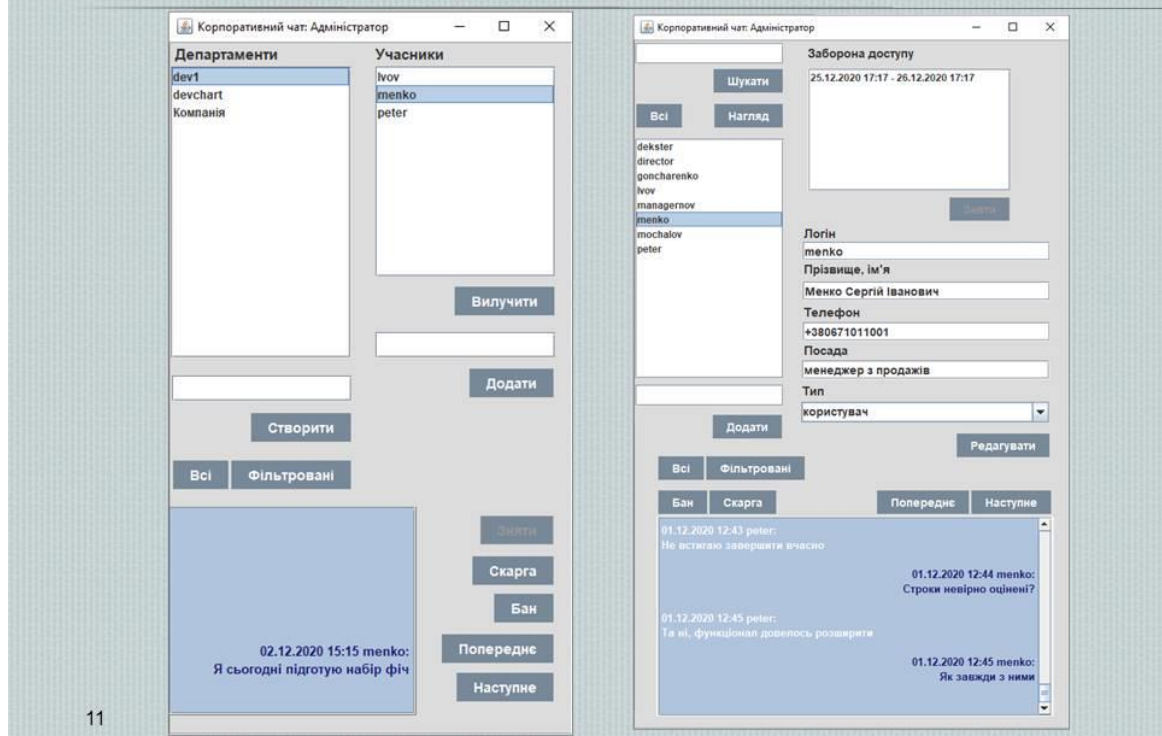


Рисунок В.10 – Слайд 10

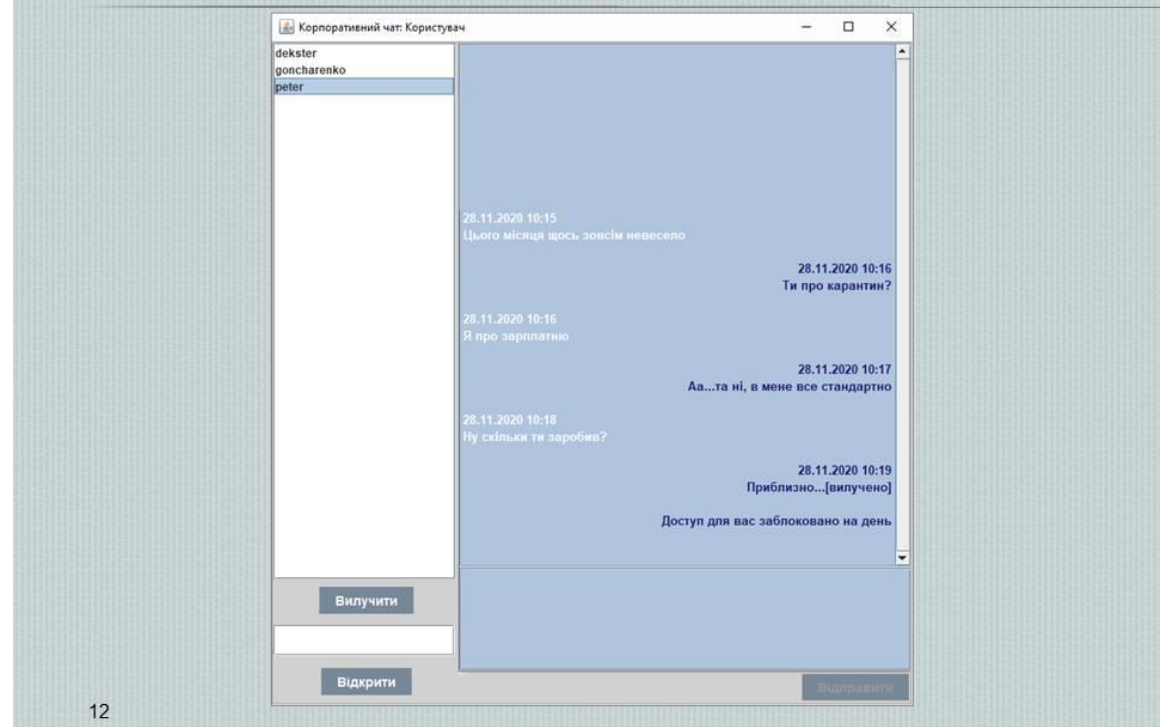
Взаємодія адміністратора з програмою



11

Рисунок В.11 – Слайд 11

Фільтрування повідомлень та заборона доступу



12

Рисунок В.12 – Слайд 12

Висновки

Розглянуто проблеми сучасних засобів комунікації відносно корпоративної сфери, правила поведінки у корпоративному месенджері.

Виконано огляд програмних засобів, які можуть використовуватися для підтримки комунікації користувачів у корпоративній мережі, зокрема Slack та MyChat. Slack є глобальним рішенням, призначеним для корпоративної мережі, а MyChat локальним.

Проаналізовано методи інтелектуального аналізу даних, які можуть використовуватися для фільтрації та обробки тексту. Проаналізовано метод опорних векторів. Розроблено власний метод керування спілкуванням користувачів.

Визначено функції, які повинна виконувати програма. Створено діаграми прецедентів для моделювання функціональних вимог для роботи користувача та адміністратора з програмою. Обрано мову програмування Java для реалізації програми. Системою керування базами даних обрано MySQL. Визначено набір таблиць, їх структуру та побудовано структурну схему бази даних.

13

Рисунок В.13 – Слайд 13

Висновки

Реалізовано програмне забезпечення, яке дозволяє виконувати приватне і відкрите спілкування, створювати групи для спілкування, обмежувати доступ до спілкування та автоматизовано рекомендувати рішення щодо цього.

Наукова новизна роботи в розробленому методі, який базується на застосуванні фільтрації повідомлень шляхом визначеного фільтру слів, словосполучень та конструкцій речень та автоматичному аналізу цензурованих повідомлень, обчисленні статистики кожного користувача в кожен період часу, винесенні рекомендованого рішення щодо користувача на основі застосування методу опорних векторів для класифікації користувачів в поточний момент.

Усі завдання, поставлені на роботу, виконано.

14

Рисунок В.14 – Слайд 14