МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____Є. О. Давиденко

«___»_____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ОБМІНУ ПОВІДОМЛЕННЯМИ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409. 21810909

Студент _____ О. В. Гранченко

«___»_____2022 р.

Керівник д-р. техн. наук, доцент _____ А. В. Швед

«___»_____2022 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва

«___»_____2022 р.

Миколаїв 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Є. О. Давиденко

«___» _____ 2021__ р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук

_Гранченко Олександр Віталійович_____

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Програмне забезпечення обміну повідомленнями в режимі реального часу_____

Затверджена наказом по ЧНУ ім. П. Могили від «01» __грудня_ 2021 р. № 314

2. Строк представлення кваліфікаційної роботи «__» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Початкові дані: функціональні вимоги до застосунку.

Очікуваним результатом роботи є програмне забезпечення обміну повідомленнями в режимі реального часу для операційної системи Windows_____

4. Перелік питань, що підлягають розробці: аналіз предметної області та існуючих аналогів; моделювання ПЗ застосунку, проєктування архітектури

5. системи та інтерфейсу користувача; розробка програмного забезпечення месенджеру; тестування та аналіз якості отриманих результатів

5. Перелік графічних матеріалів

Презентація_____

_____

6. Завдання до спеціальної частини

Розглянути особливості гігієни праці для фахівців іт-індустрії та визначити умови, при яких фахівець буде здоровим та максимально продуктивним під час роботи в ІТ сфері_____

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
А. О. Алексєєва	Кафедра екології	Охорона праці

Керівник роботи _доцент кафедри ІПЗ, д-р техн. наук Швед Альона Володимирівна_____

(посада, прізвище, ім'я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

_____Гранченко Олександр Віталійович_____

(прізвище, ім'я, по батькові студента)

_____

(підпис)

Дата видачі завдання «____» _____ 202_р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: «Програмне забезпечення обміну повідомленнями в режимі реального часу»

№	Найменування роботи	Початок	Закінчення	Примітка
1.	Аналіз предметної області	20.11.21	24.11.21	Виконано
2.	Огляд літератури за темою роботи	25.11.21	03.12.21	Виконано
3.	Аналіз мессенджерів після огляду літератури	06.12.21	08.12.21	Виконано
4.	Складання календарного плану	09.12.21	10.12.21	Виконано
5.	Моделювання та конструювання ПЗ	13.12.21	24.12.21	Виконано
6.	Основна розробка, кодування, створення користувацького інтерфейсу	10.01.22	01.04.22	Виконано
7.	Тестування, аналіз результату тестування	04.04.22	02.05.22	Виконано
8.	Оформлення КРБ та презентації	03.05.22	27.05.22	Виконано
9.	Попередній захист	30.05.22	04.06.22	Виконано
10.	Завершення оформлення КРБ та презентації	06.06.22	18.06.22	Виконано
11.	Захист КРБ	28.06.22	28.06.22	Виконано

Розробив студент _Гранченко Олександр Віталійович_____ _____

(*прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 20__ р.

Керівник роботи _д-р. техн. наук, доцент Швед А. В._____ _____

(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 20__ р.

АНОТАЦІЯ

До кваліфікаційної роботи бакалавра

«Програмне забезпечення обміну повідомленнями в режимі реального часу»

Студент 409 гр.: Гранченко Олександр Віталійович

Керівник: доцент кафедри ІПЗ, д-р техн. наук Швед А. В.

Кваліфікаційна робота бакалавра спрямована на розробку мессенджеру обміну повідомленнями в режимі реального часу, основною перевагою якої є розширений функціонал та оптимізація роботи з текстовими повідомленнями. Метою кваліфікаційної роботи бакалавра є підвищення зручності процесу комунікації учасників інтерактивного спілкування.

Об’єктом роботи є процес комунікації між учасниками інтерактивного спілкування, шляхом обміну повідомленнями в режимі реального часу.

Предметом роботи є інформаційні технології та інформаційні системи обміну повідомленнями в режимі реального часу.

Структурно робота складається із п’яти розділів.

У першому розділі виконано аналіз предметної сфери, проаналізовано існуючі сучасні рішення, визначено їх переваги а недоліки. Сформульовані вимоги до програмного застосунку, виконана постановка задачі розробки.

У другому розділі наведені рішення з моделювання програмного забезпечення застосунку.

У третьому розділі наведені проектні рішення та рішення з програмної реалізації застосунку. Обґрунтовано вибір засобів розробки.

У четвертому розділі описано процес розробки застосунку та проведено тестування.

П’ятий розділ присвячено вирішенню питань з охорони праці. Розглядались питання особливості гігієни праці для фахівців іт-індустрії.

Результатом роботи є мессенджер для операційної системи Windows, який включає в собі додатковий функціонал, що дозволяє звичайному користувачеві оптимізувати процес комунікації з іншими користувачами. Дане програмне забезпечення призначене для використання в повсякденних цілях, зокрема для звичайного та зручного спілкування з рідними та близькими.

КРБ викладена на 55 сторінки, вона містить 4 розділи, 20 ілюстрацій, 2 таблиці, 20 джерел в переліку посилань.

Ключові слова: мессенджер, комунікація, Windows.

ABSTRACT

Of the Bachelor’s Thesis

«Real-time messenger application»

Student of group 409: Hranchenko Oleksandr

Supervisor: Associate Professor at the Department of SE, Dr.Sc., Professor Shved A. V.

The thesis is aimed at developing a messenger of real-time messaging, the main advantage of which is the expanded functionality and optimization of work with text messages. The purpose of the thesis is to increase the convenience of the communication process of the participants of interactive communication.

The object of the thesis is the process of communication between the participants of interactive communication, through the exchange of messages in real time.

The subject of the thesis is information technologies and information systems of real-time messaging.

Structurally, the thesis consists of five sections.

The first section analyzes the subject area, analyzes the existing modern solutions, identifies their advantages and disadvantages. The requirements to the software application are formulated, the development task is set.

The second section provides solutions for modeling application software.

The third section presents design solutions and solutions for software implementation. The choice of development tools is substantiated.

The fourth section describes the application development and testing process.

The fifth section deals with occupational safety. The issues of occupational health for specialists in the IT industry were considered.

The result is a messenger for the Windows operating system, which includes additional functionality that allows the average user to optimize the process of communicating with other users. This software is designed for everyday use, in particular for normal and convenient communication with family and friends.

Thesis is set out on 55 pages, it contains 4 sections, 20 illustrations, 2 tables, 20 sources in the list of links.

Keywords: messenger, communication, Windows.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

СМС — служба коротких повідомлень;
ПК — персональний комп’ютер
ОС — операційна система
VPN — віртуальна приватна мережа
2FA — двофакторна аутентифікація
СКБД — система керування базами даних

ВСТУП

Актуальність теми зумовлена недосконалістю сучасних застосунків обміну миттєвими повідомленнями. Основою сучасного суспільства є інформаційний обмін. Інформація є однією з найбільших цінностей. Розвивається безліч видів представлення даних та її представлення користувачам. Інформація швидко застаріває. Вважається, що обмін інформацією став причиною появи різного роду мереж, технічних інформаційних систем, в тому числі комп'ютерних мереж. Спочатку мережі об'єднували окремі комп'ютери в межах будівлі. Далі з'явилися мережі, які об'єднували комп'ютери різних міст і країн. І сьогодні ми можемо спостерігати глобальну мережу, яка насправді є об'єднанням безлічі різних систем більш низького рівня. Завдяки мережі Інтернет людство отримало в свої руки унікальну систему, всі можливості якої до кінця не досліджені й на сьогоднішній день.

Говорячи про обмін інформацією або в кінцевому підсумку даними, найважливішим фактором є не тільки швидкість обміну, а й можливість взаємодії з джерелом інформації, можливість задавати питання і отримувати відповідь в реальному часі, отримувати свіжі новини і повідомляти їх іншим.

Це важливо тоді, коли ми контактуємо з реальною людиною, яка може бути на іншому кінці світу, а може й бути в сусідній з вами кімнаті і, незалежно від цього, ви можете в одну мить передати йому необхідну інформацію і настільки ж швидко отримати відповідь.

Відправлення звичайного повідомлення є нині не поганою “розумовою гімнастикою” через те, що велика кількість людей навколо використовують найрізноманітніші застосунки для спілкування on-line, адже єдиного мессенджера, що міг би влаштувати потребності всіх людей – не існує.

Розробка мессенджерів буде актуальною завжди, адже люди використовують такі застосунки як Viber та Telegram кожного дня, проте їх функціонал та захист не є ідеальним і деякі люди використовують такі

мессенджери через відсутність варіантів, які б задовольнили їх на сто відсотків. Зважаючи на те, що об’єднати усі мессенджери в один є неможливим для виконання завданням, найбільш вдалим рішенням буде створення застосунку, що буде задовольняти усі потреби звичайного користувача у обміні текстовими повідомленнями.

Великий вибір серед мессенджерів створює здорову конкуренцію у цій сфері, що дуже добре впливає на розвиток таких застосунків, проте єдиного варіанту не існує, що створює певні труднощі у комунікації між звичайними людьми. Сучасні месенджери створюються відповідно до нових актуальних потреб користувачів і забезпечують той функціонал, який найбільше потрібен сьогодні. Вони постійно вдосконалюються та доповнюються на відміну від давно існуючих аналогів, у яких оновлення з'являються не так часто.У нинішніх умовах багато хто перейшов на віддалену роботу і мессенджер із засобу спілкування перетворився на робочий інструмент.

Об’єктом роботи є процес комунікації між учасниками інтерактивного спілкування, шляхом обміну повідомленнями в режимі реального часу.

Предметом роботи є інформаційні технології та інформаційні системи обміну повідомленнями в режимі реального часу.

Метою роботи є підвищення зручності процесу комунікації учасників інтерактивного спілкування за рахунок створення системи обміну повідомленнями в режимі реального часу, основною перевагою якої буде розширений функціонал та оптимізація роботи з текстовими повідомленнями.

Для досягнення визначеної мети необхідно вирішити такі **завдання**:

1. проаналізувати існуючі аналоги. Визначивши їх переваги та недоліки;
2. на основі аналізу визначити вимоги до майбутнього застосунку;
3. змоделювати та зпроектувати застосунок;
4. розробити клієнтську та серверну частини ПЗ, реалізувати базу даних;
5. провести тестування ПЗ

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ АВТОМАТИЗАЦІЇ ІНФОРМАЦІЙНО-ТЕЛЕКОМУНІКАЦІЙНИХ ПРОЦЕСІВ В СИСТЕМАХ ОБМІНУ ДАНИМИ

1.1 Опис предметної сфери

Слово "messenger" означає «висланець». Це система швидкого обміну повідомленнями в вигляді тексту, аудіо, відео і файлів різного типу.

Мессенджер як застосунок постає перед нами чудовим прикладом величезного покращення та зростання популярності як засобу інтернет-комунікації. Якщо порівнювати з іншим online засобом зв’язку, таким як електронна пошта, де ви повинні дочекатися доки отримувач перевірить свою пошту, оформить лист та відправить вам у відповідь, обмін миттєвими повідомленнями відбувається в режимі реального часу, коли одержувач знаходиться в мережі і є змога обмінюватися короткими повідомленнями.

Головною перевагою мессенджеру в порівнянні з SMS є використання мережі Internet, а в сучасному світі це дешевий ресурс. Більшість сучасних тарифів дозволяють користуватися обміном інформації без ліміту, як і в разі WI-FI передачі.

Хоч і в електронної пошти, і в SMS також є миттєвість передачі та отримання повідомлення, однак перевагою саме мессенджеру є те, що відображається статус користувача, щоб зрозуміти, прочитає він відразу текст чи ні.

Функціонал популярних мессенджерів:

— швидкий обмін інформацією;

— відображення статусу співрозмовника;

— створення групових чатів;

— можливість ведення розмови через аудіо та відео;

— витрачання трафіку інтернету (нині зазвичай вже безліміт);

— можливість обміну фото, відео, файлів різного типу.

У кожного мессенджера є сервер, де йде обробка даних та їх зберігання. Часто вони несумісні один з одним, так як працюють за власними протоколами передачі. У серверній частині зберігаються паролі, логіни і забезпечується безпека.

Розробники намагаються не поєднувати протоколи, тому клієнти встановлюють кілька додатків на комп'ютер і телефон для спілкування, якщо є необхідність [2].

Спілкування за допомогою мессенджерів давно стала для нас невід'ємною частиною повсякденності. Мессенджери дозволяють швидко і легко взаємодіяти один з одним з найрізноманітніших питань: робота, навчання, розваги, особисті справи. Функції сучасних мессенджерів давно вийшли за рамки звичного обміну повідомленнями, фотографіями, відео. Ми можемо здійснювати аудіо- і відеодзвінки, робити покупки, оплачувати комунальні послуги, отримувати довідкову інформацію, замовляти продукти харчування, таксі, квитки і багато іншого.

Розглянемо топ-6 популярних мессенджерів на січень 2021 року згідно statista.com (активних користувачів в мільйонах). Дивіться (рис. 1.1).

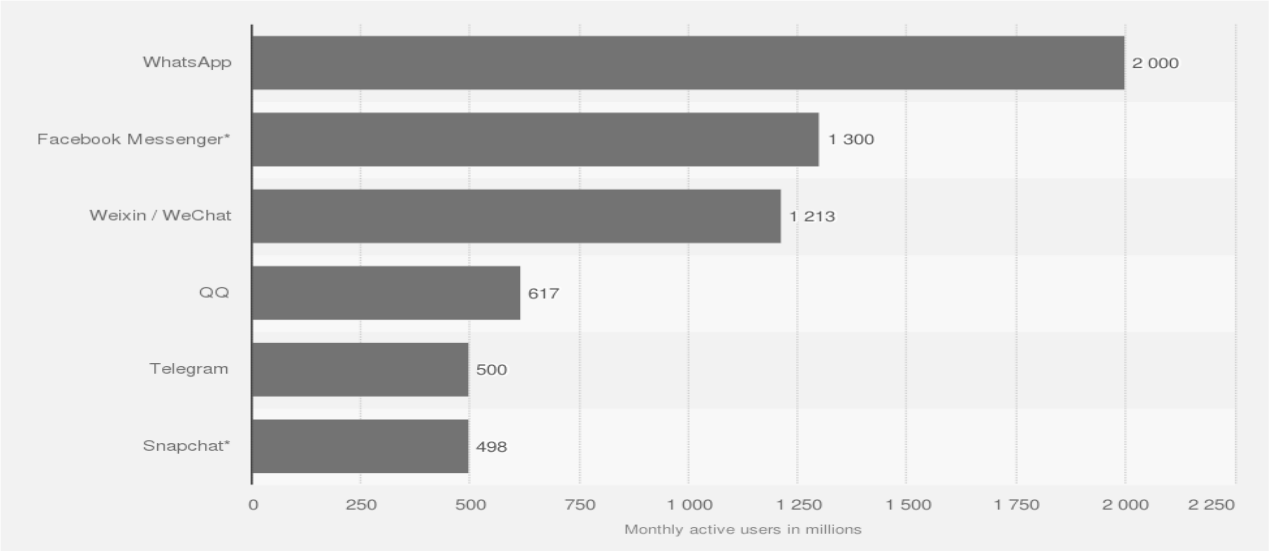


Рисунок 1.1 – Топ-6 популярних мессенджерів на січень 2021 року згідно statista.com (активних користувачів в мільйонах)

Розвиток хмарних технологій також посилило вплив мессенджерів на наше життя: це дійсно зручно, коли під рукою важливі файли, листування, контакти, якими ми можемо поділитися в кілька натискань на смартфоні, клацанням тачпада або миші [1].

1.2 Аналіз сучасних програмних систем обміну повідомленнями в режимі реального часу

Розглянемо найбільш використовувані мессенджери згідно statista.com.

WhatsApp

Рисунок 1.2 – WhatsApp в «Play Market»

Акаунт в застосунку відповідає номеру телефону. Такий сервіс доступний як на ПК, так і на телефонах будь-якої платформи. Можна здійснювати аудіо- і відеодзвінки, формувати загальні чати.

Переваги:

— простота використання;

— висока швидкість;

— групові чати;

— дзвінки;

— власна система шифрування даних;

— офлайн-повідомлення.

— Недоліки:

— зниження якості переданих файлів;

— обмеження по розміру відправляються файлів;

— оповіщення приходять із запізненням [2].

Telegram

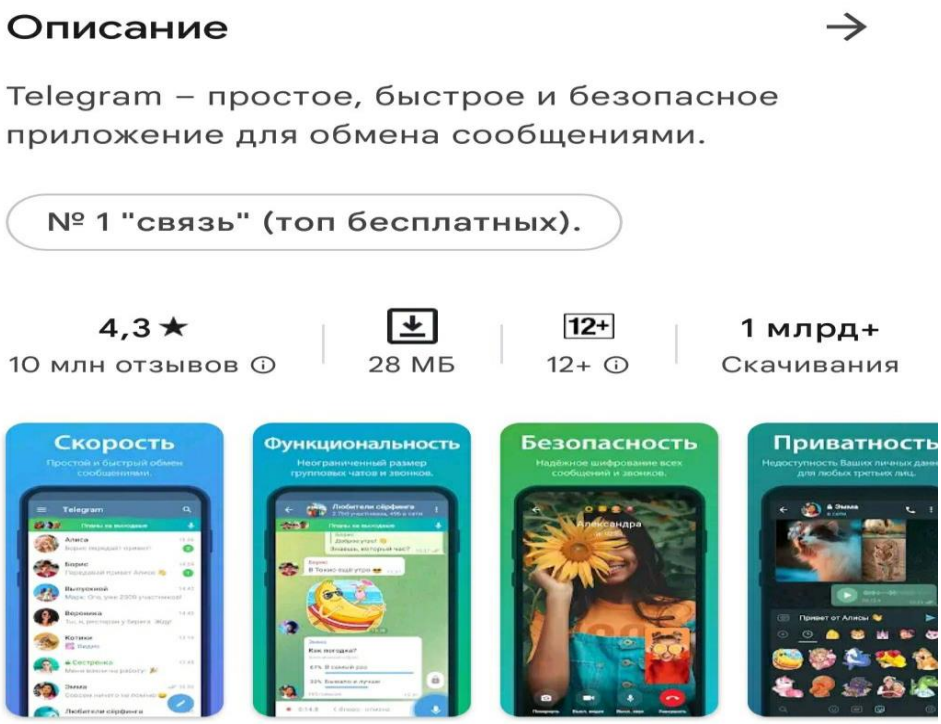


Рисунок 1.3 – Telegram в «Play Market»

Такий мессенджер працює за схожим алгоритмом з WhatsApp. Головна особливість - безпека шифрування даних, більш надійний захист.

Переваги:

— ведення каналів (такий формат зручний для новин, розсилки інформації без зворотного зв'язку);

— створення і підключення ботів (хороший інструмент для ведення бізнесу);

— пошук користувачів по username;

— вбудований редактор фотографій;

— секретний чат, самовидаляючі повідомлення;

— безкоштовні стікери, можна створити свій набір.

— Недоліки:

— немає дзвінків;

— відсутність секретних чатів на ПК [3].

Facebook Messenger

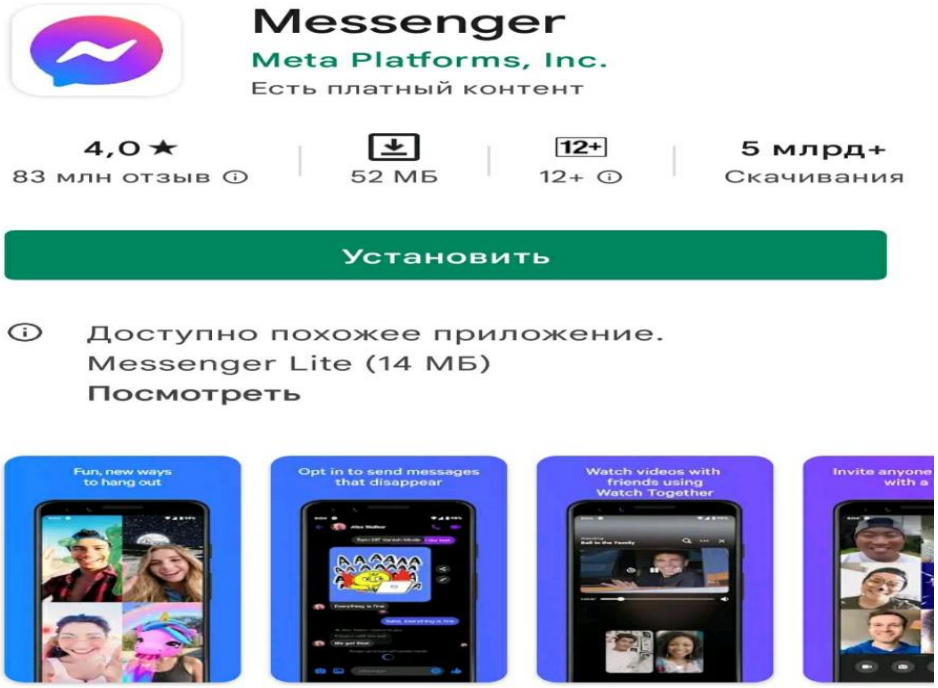


Рисунок 1.4 – Facebook Messenger в «Play Market»

Facebook Messenger – це чат для внутрішнього спілкування всередині соціальної мережі Facebook. Такий мессенджер повідомляє про нові коментарі, зустрічі. Можливий обмін текстовими повідомленнями, фото, відео, є можливість дзвінків (в тому числі групових). Підтримує функцію секретних чатів. Є стікери і Emoji, а також можна здійснювати платежі (при наявності картки американського банку).

Переваги:

— підтримка ботів;

— відключення повідомлень;

— секретний чат;

— швидке вітання через спеціальну клавішу в діалозі;

— можливість залишення своїх координат в діалозі.

— Недоліки:

— тільки для користувачів Facebook;

— повідомлення не можна видаляти або редагувати;

— різний функціонал web-версії і додатки [4].

Viber

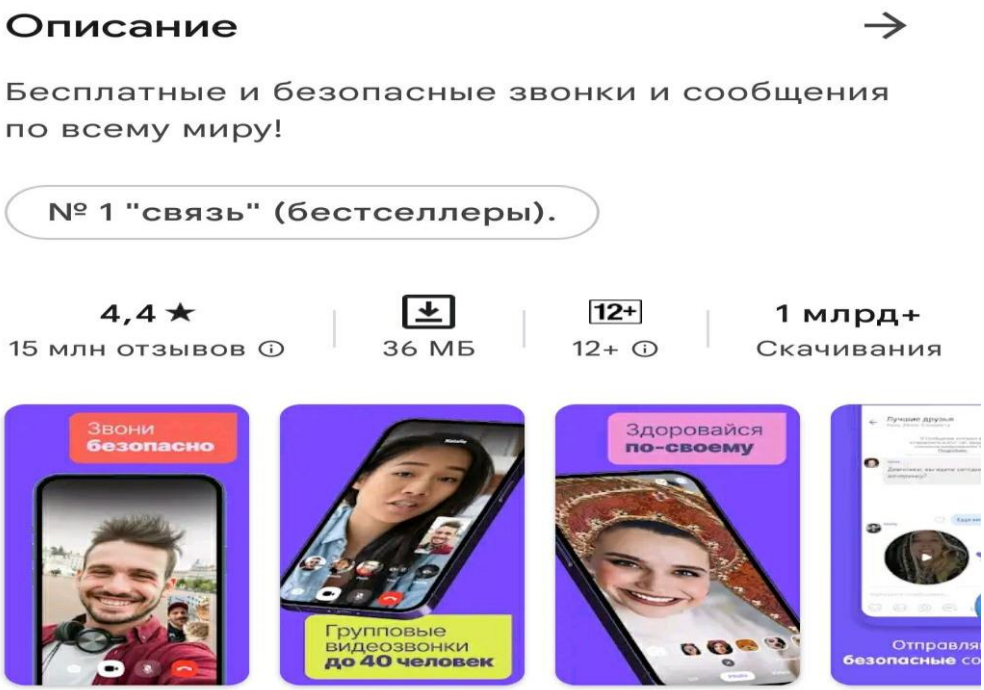


Рисунок 1.5 – Viber в «Play Market»

Viber – це спеціальний застосунок для обміну текстовою інформацією, фото, відео, файлами. Цей сервіс підтримує й дзвінки.

Переваги:

— ігрова платформа;

— відкриті чати;

— великий вибір стікерів, Emoji, фонів.

— push-повідомлення в офлайн режимі;

— синхронізація дзвінків на всіх використовуваних пристроях;

— верифікація контактів вручну;

— виклики на стаціонарні та звичайні номери (потребує коштів).

Недоліки:

— багато спаму;

— погано захищений.

Skype

Рисунок 1.6 – Skype в «Play Market»

Skype – це усім відома програма для спілкування. Працює на Windows, Linux, Mac OS X, Pocket PC платформах + є мобільний додаток. Зручно для створення конференцій з друзями, родичами або колегами. Ідентифікатор – електронна пошта [3].

Переваги:

— безкоштовні групові чати і відеодзвінки;

— офлайн-спілкування;

— можлива демонстрація під час відеоконференції;

— запис відеорозмови;

— необмежений час розмов по відео;

— підтримка ботів.

недоліки:

— неякісна синхронізація з різними пристроями;

— відсутність шифрування;

— високі потреби до ресурсів пристрою;

— не синхронізується зі списком контактів;

— на звичайні номери дзвінки платні.

Skype прийнято вважати одним з лідерів свого сегменту на PC, в разі мобільного застосування менш популярний. Якщо Ви часто використовуєте такий додаток на комп'ютері, то мобільна версія також буде корисною, хоча вона не створює гідну конкуренцію аналогам на цій платформі.

Slack

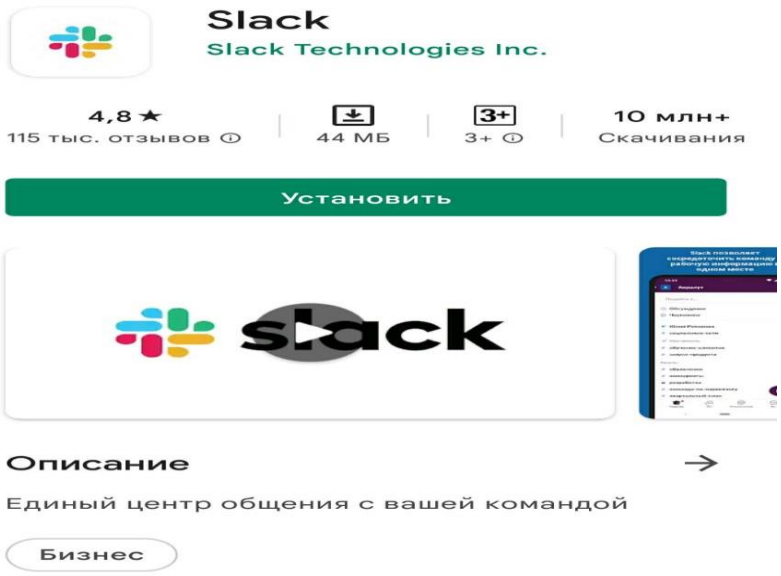


Рисунок 1.7 – Slack в «Play Market»

Slack – це переважно корпоративний мессенджер, який використовується для спілкування співробітників, схожий по інтерфейсу і функціоналу зі Skype. Є створення тематичних каналів, відео- і аудіозвінків, конференцій; існує архівування даних, проведення опитувань.

Переваги:

— швидка синхронізація;

— отримання оповіщення у вигляді спливаючого вікна або листи на електронну пошту;

— інтегрування зі сторонніми сервісами (GoogleDrive, Бітрікс24, іншими мессенджерами);

— зрозуміле оформлення.

— Недоліки:

— інтерфейс лише англійською мовою;

— повний функціонал за окрему плату;

— завантаженість оперативної пам'яті [5].

1.3 Захист інформації в системах обміну даними

Щоб оцінити безпеку даних користувача в тому чи іншому мессенджері, позначимо ключові критерії безпеки, приватності та анонімності. Після цього зможемо звести дані в таблицю, проаналізувати і зробити висновки.

Так як сучасні додатки запускаються в незахищених середовищах і для них забезпечується інший контроль поведінки, все зводиться до того, як саме організована логіка роботи програми на тій чи іншій платформі. Наприклад, процес біометричної аутентифікації в мессенджерах може бути реалізований без використання системних апаратних механізмів. При певних умовах (root-доступ, jailbreak) це дозволяє зловмисникові підмінити значення результату роботи локальної функції, що відповідає за аутентифікацію в мессенджері, і отримати доступ до застосунку.

Apple iOS ретельніше організовує шифрування особистих файлів, які зберігаються на пристрої, але багато що залежить від реалізації опцій DataProtection в коді програми. Аналогічна ситуація з Google Android, який також в фінальних версіях значно просунувся в захисті призначених для користувача даних. У разі формування оцінки критеріїв безпеки з позиції ОС, відкритий вихідний код має перевагу перед закритим – так можна провести аудит під обидві платформи.

А ось підтримка наскрізного (end-to-end) шифрування гарантує, що тільки ви і адресат зможете розшифрувати і прочитати інформацію. E2E вважається основним атрибутом будь-якого мессенджера, який позиціонує себе як безпечний. Наприклад в iMessage до недавнього часу шифрування доводилося активувати в налаштуваннях самостійно (сині повідомлення означають, що опція наскрізного шифрування активована, зелені – немає). Тут же важливо розуміти, які криптографічні алгоритми застосовуються для організації шифрування. Де генерується закритий ключ? Чи відбувається змішування

метаданих? Організовано чи ротація ключів через певний часовий інтервал? Збір даних і метаданих, які кожен з нас генерує своїми діями в мережі, схожі з цифровим відбитком особистості. Мессенджери також збирають метадані, які можуть описувати нашу особистість досить докладно. По суті це – всі дані, крім змісту безпосередньо повідомлення: наприклад, з ким з нашого списку контактів ми розмовляємо, як довго і як часто (відправник, отримувач, час відправлення, час прочитання). Це – запис нашої активності. Також може збиратися інформація про пристрій, IP-адресу, номер мобільного і т.д. А також збір даних про користувачів. Як мінімум це інформація про користувача при реєстрації. У деяких випадках складно визначити, які саме дані збираються, тому як мессенджери бувають інтегровані в екосистему корпорації-виробника (GoogleMessages, AppleiMessage). Компанії можуть бути відомі ідентифікатор користувача, телефон, вміст листування, історія пошуків, переглядів, інформація про покупки, місце розташування, контакти і багато іншого.

Відкритий вихідний код програми для обміну миттєвими повідомленнями дозволяє здійснювати комплексний аудит безпеки. Аматори, ентузіасти, експерти можуть зробити збірку додатку, досліджувати його роботу і привернути увагу до слабких місць, до вразливостей як в серверній, так і в клієнтської частинах коду. З іншого боку, вільний доступ до коду підвищує ризик того, що інформація про виявлену вразливість може використовуватися зі злим наміром, поки не буде закрита або хтось інший із товариства не зверне увагу на слабке місце.

Відкритість коду не може гарантувати безпеку для користувача даних, але є важливим атрибутом її побудови. Переважна більшість незалежних аудиторів зацікавлені в еволюції надійності і безпеки коду мессенджера, а цього можна досягти тільки спільними зусиллями.

Адміністрація деяких мессенджерів активно співпрацює з третіми особами, а інші принципово відмовляються передавати особисті дані. Зловмисник може представитися ким завгодно, в тому числі і співробітником спецслужби, отримавши в підсумку необхідні цінні відомості. При виборі

безпечного додатка це необхідно враховувати, тому як конфіденційні дані можуть потрапити не в ті руки, навіть якщо ви законослухняний громадянин.

Не всі мессенджери застосовують шифрування для зберігання листування і файлів в хмарному середовищі. Успішна атака зловмисника на хмарну інфраструктуру може привести до витоку конфіденційної інформації. Так само як і у випадку зі збором даних, інформація про те, чи дійсно резервні копії даних шифруються, є у відкритому доступі далеко не по всіх мессенджерах.

Однорангове, або пірінгове (peer-to-peer), з'єднання виключає участь третьої сторони. Відправлені повідомлення надходять безпосередньо на пристрій адресата. Важливо зауважити, що таке з'єднання вільно дозволяє побачити, з ким і як довго воно встановлено, що, природно, впливає на анонімність і знижує рівень конфіденційності. Підвищити конфіденційність можна додатковим захистом IP-адреси: використовувати VPN.

При створенні акаунта в мессенджері, наприклад, часто потрібно вказати номер мобільного телефону, який вкрай тісно пов'язаний з нашою реальною особою. Безпека даних може бути не порушена, але анонімність значно знижується. Чим більше даних потрібно при реєстрації, тим нижче анонімність. Це може бути вимога адреси електронної пошти, додаток може запитати доступ до контактів або до вхідних SMS-повідомлень для верифікації. Підтвердження реєстрації може бути реалізовано через дзвінок на номер.

Підтримка мессенджером двофакторної аутентифікації – важливий додатковий елемент безпеки. Другий рівень захисту на основі 2FA (Two-FactorAuthentication) може ефективно зупинити зловмисника. Деякі додатки пропонують користувачеві активувати 2FA шляхом подання відповідного повідомлення. Звідси і наявність опції, що дозволяє встановити код або пароль для доступу до важливих налаштувань безпеки або листування, до чатів; захист інформації, що відображається на екрані інформації (наприклад, коли користувач намагається зробити скріншот секретного листування, другий співрозмовник отримує повідомлення про це); автоматичне блокування екрану,

коли користувач відійшов від пристрою; видалення попереднього прив'язаного пристрої з аккаунта і т. д.

В таблиці 1.1 описано відповідність мессенджерів критеріям безпеки. Чисельний вимір критерію від -100 балів до +100 балів.

Таблиця 1.1 – Підтримка функцій безпеки і приватності популярних мессенджерів

Критерії/ Мессенджери	Signal	Telegram	Viber	WhatsApp	Skype
Загальний бал безпеки та приватності	46	14	2	-10	-40
Критерій 1-4	+56	+40	+36	+30	+16
Критерій 5-8	-10	-26	-34	-40	-56

Як видно з порівняльної таблиці, жоден мессенджер не є ідеальним в плані безпеки. Важливо розуміти, що розробники можуть нехтувати безпекой застосунку заради зручності використання, що є більш привабливим для звичайного користувача.

Через це нам іноді доводиться використовувати не один мессенджер, щоб підтримувати зв'язок з нашими близькими, друзями та колегами по роботі. В залежності від сфери використання, вимоги до безпеки таких застосунків можуть бути знижені, проте не слід забувати про базові потреби користувача в конфіденційності особистих даних, котрі він, наприклад, вводить при реєстрації свого акаунта. Тож, щоб знизити ризики того, що особисті дані користувача можуть витекти в мережу, доцільним буде вимикати функцію резервного копіювання даних в мессенджері та регулярно оновлювати особисті паролі, використовуючи великі та маленькі літери, цифри і спеціальні символи. Також не варто нехтувати довжиною (від восьми символів).

Якщо обирати мессенджер з точки зору зручності, то можна зупинитися на Skype, WhatsApp та Facebook Messenger, адже ці застосунки не відзначаються сильною безпекой, проте є зручними у використанні.

1.4 Специфікація вимог до програмного забезпечення обміну повідомленнями в режимі реального часу

Задля зручності, специфікацію вимог до програмного забезпечення обміну повідомленнями в режимі реального часу представлено таблицей.

Таблиця 1.2 – Специфікація вимог

Назва	Функціональність вимога	Не функціональна вимог	Опис	Пріоритет	Актор
Виділити повідомлення	Так		Можливість виділити обране повідомлення певним кольором	Середній	Користувач
Надійність	Так		Захист застосунку	Середній	Адміністратор
Змінити аватар	Так		Можливість змінити аватар на обраний	Низький	Користувач
Реєстрація користувача	Так		Можливість створити акаунт	Високий	Користувач
Вхід в систему	Так		Можливість увійти в систему	Високий	Користувач
Вихід із системи	Так		Можливість вийти з системи	Високий	Користувач
Додати друга	Так		Можливість додати іншого користувача в список друзів	Високий	Користувач
Список друзів	Так		Можливість переглядати список друзів	Високий	Користувач

Продовження таблиці 1.2

Видалити друга	Так		Можливість видаляти друга зі списку друзів	Високий	Користувач
Знайти друга	Так		Можливість знаходити ім'я друга зі списку за допомогою пошукача	Середній	Користувач
Заблокувати друга	Так		Можливість додати друга в «чорний» список	Середній	Користувач
Додати в «Найкращі друзі»	Так		Можливість додати друга в «найкращі» друзі	Середній	Користувач
Відправити повідомлення	Так		Можливість відправляти повідомлення іншому користувачеві	Високий	Користувач
Статус повідомлення	Так		Відображення поточного статусу повідомлення, чи відправлено/прочитано	Високий	Користувач
Видалити повідомлення	Так		Можливість видалення обраного повідолмення	Середній	Користувач
Залишити фідбек	Так		Можливість залишити відгук про застосунок	Низький	Користувач

Реалізувавши усі ці вимоги, можна створити конкурентний застосунок.

Висновки до розділу 1

Під час роботи було проведено аналіз існуючих мессенджерів, виявлено їх переваги та недоліки. Застосунки для обміну повідомленнями зараз мають більше глобальних користувачів, ніж традиційні соціальні мережі, а це означає, що вони відіграватимуть дедалі важливішу роль у поширенні інформації в майбутньому. Хоча чатові платформи спочатку набули популярності, пропонуючи недорогу веб-альтернативу SMS, з часом вони перетворилися на мультимедійні центри, які підтримують фотографії, відео, платежі тощо. При виборі мессенджера для повсякденного спілкування пересічному користувачеві буде оптимально поєднувати розумний баланс між зручністю і безпекою. Швидкість доставки повідомлення в більш безпечному мессенджері значно знижується. Доводиться приносити в жертву зручність користування і час заради підвищення пріоритетів безпеки.

Було досліджено різні підходи до захисту обміну даними та складено таблицю порівнянь популярних мессенджерів у цьому питанні.

Також було написано специфікацію вимог до програмного забезпечення, що дозволяє виділити задачі та розставити приорітети при розробці застосунку.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОБМІНУ ПОВІДОМЛЕННЯМИ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

2.1 Діаграма прецедентів програмного забезпечення обміну повідомленнями в режимі реального часу

Діаграма варіантів використання або діграма прецедентів (use-case diagram) – це спосіб узагальнити деталі системи та користувачів у обраній системі. Загалом це графічне зображення взаємодій між різними елементами в системі. Діаграми варіантів використання вказують події в системі та те, як ці події протікають, однак діаграма варіантів використання не описує, як ці події реалізуються.

Така методологія використовується в системному аналізі для визначення, роз’яснення та організації системних вимог. У цьому контексті термін «система» відноситься до чогось, що розробляється або функціонує, наприклад, веб-сайт продажу та обслуговування продуктів поштою. Діаграми варіантів використання використовуються в UML (Unified Modeling Language), стандартній нотації для моделювання реальних об’єктів і систем. Наявність діаграми варіантів використання має ряд переваг перед подібними діаграмами, такими як блок-схеми.

Діаграма варіантів використання включає декілька ключових компонентів:

Актори – індивідууми, які мають відношення в межах системи. Вони взаємодіють з кожним кроком процесу. Це, можливо, включає користувачів, клієнти, клієнти або Актори працівників, можливо, взаємодіють з системою внутрішньо або зовні.

Варіанти використання, що часто представляються овалом, описують системну функцію, яку актор може почати або використовувати.

Зв'язувальні елементи можуть представити взаємодію актора з випадком використання або користуватися взаємодіями випадків.

Узагальнення представляють собою стосунки між акторами, які залежать від функцій одне одного, щоб завершити їх роль в межах системи.

Розширення представляють необов'язкові функції в межах системи. Ці зв'язки сполучають акторів, щоб користуватися варіантами або багаторазовими пов'язаними варіантами використання для функціональності.

Включення представляють додаткові необхідні функції, щоб завершити систему і формують з'єднання між двома варіантами або варіантами та акторами.

Метою діаграми варіантів використання є пропонування ясної візуалізації функціональних вимог системи. Ця діаграма допомагає зробити важливі альтернативи дизайну. Також ідентифікує зовнішні і внутрішні чинники, які можуть впливати на взаємодії в системі [6].

Нижче наведено діаграму варіантів використання програмного забезпечення обміну повідомленнями в режимі реального часу.

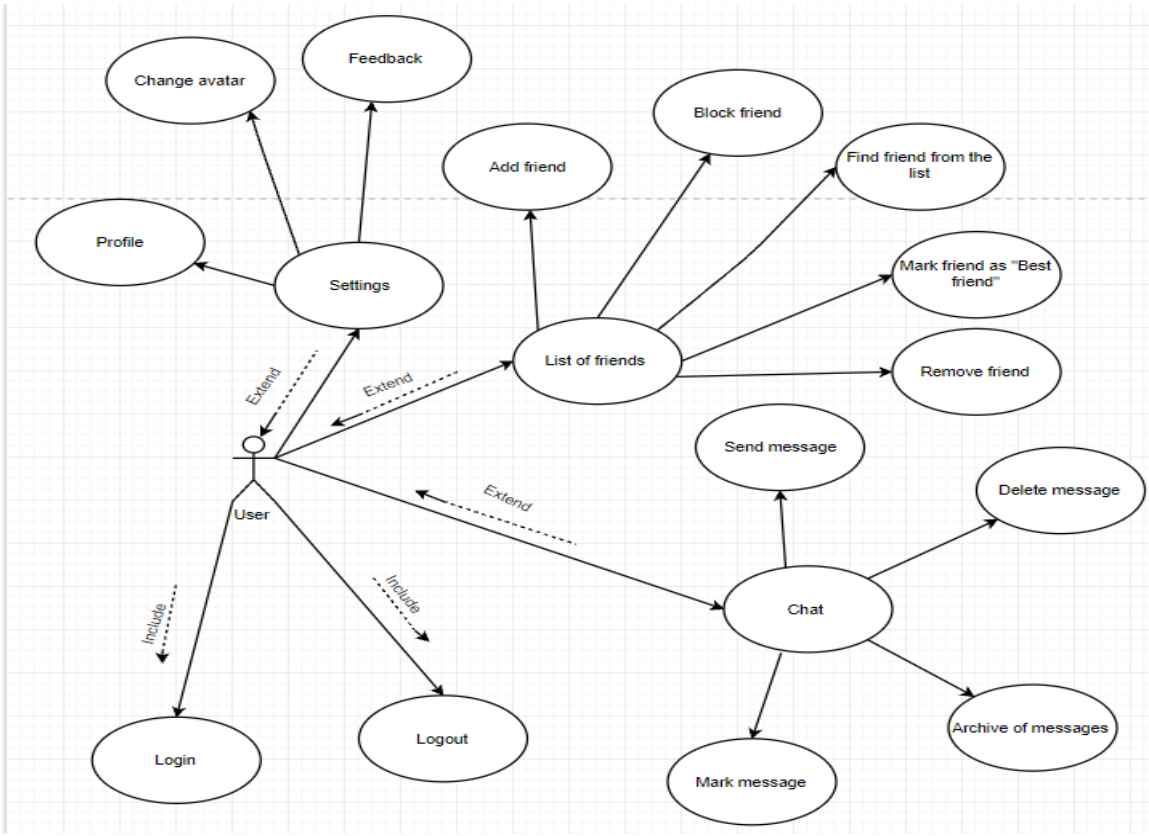


Рисунок 2.1 – Діаграма варіантів використання програмного забезпечення обміну повідомленнями

2.2 Проектування інтерфейсу системи обміну повідомленнями

Макет (mockup) показує як на що буде схожий застосунок. На відміну від прототипу, макет статичний і не включає ніяких взаємодій [7]. При розробці, макет може бути масштабним або full-size фізичною моделлю продукту. У цифровому управлінні продукту, макет буде детальним описом додатка.

Необхідно зауважити, що макети не включають функціональність продукту. Вони не дозволяють користувачеві "робити" що-небудь. Вони проектуються, щоб розділити бачення команди для продукту з іншими посередниками компанії і клієнтами.

Наприклад, макет покаже заплановані кольори, ікони, текстові шрифти, і інші візуальні елементи, оскільки вони з'явилися б в завершальний продукт. Отже, якщо команда дизайну створює успішний макет, він нагадуватиме "робочу" версію додатка хоча це не матиме ніякої реальної функціональності [8]. Макети служать переправою між раннім ескізом і кінцевим продуктом команди. Крім того, цей крок дає можливість ознайомитися з бажаннями замовника та його баченням продукту. В результаті, з'являється змога дешевше вносити зміни, адже команда ще не дійшла до створення дизайну.

Для свого мессенджеру я створив декілька макетів, щоб відобразити загальний вигляд застосунку. Нижче зображено макет логін сторінки мессенджеру.

Рисунок 2.2 – Макет логін сторінки

Візуально вже можна зрозуміти які позиції займають такі елементи сторінки, як поля вводу імені користувача та паролю, функція «Forgot Password», назва застосунку та кнопки «Login» і «Create account». Також з приведеного вище макету видно, що основним кольором усього застосунку буде зелений. Нище зображено макет створення акаунту.

Рисунок 2.3 – Макет сторінки реєстрації нового користувача

З приведеного вище макету зрозуміло які дані будуть потрібні від користувача, щоб створити новий акаунт в системі. Потрібно вказати ім’я користувача, пароль (та ввести пароль ще раз для підтвердження), поштову адресу, стать та дату народження. Також з’явилася кнопка «Back», що дозволить повернутися на попередню сторінку. Далі зображено макет головної сторінки мессенджеру, відразу після логіну.

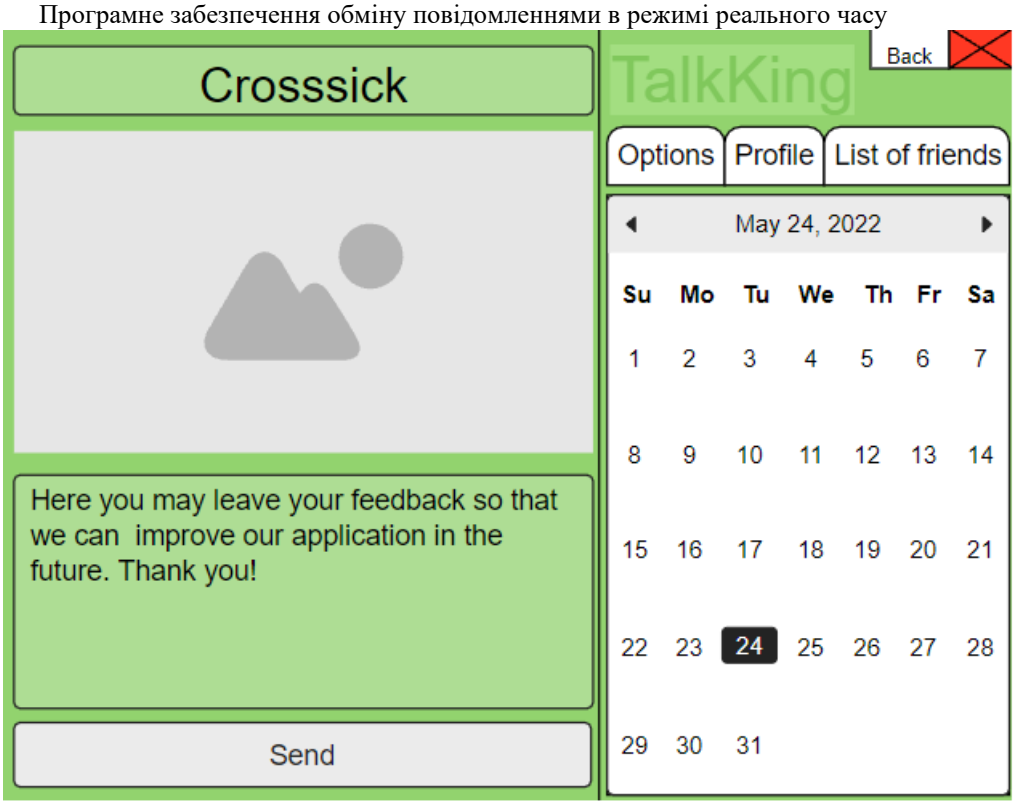


Рисунок 2.4 – Макет головної сторінки

На цьому макеті зображена головна сторінка, яку користувач бачить одразу після входу в систему. Зліва зверху виводиться ім’я користувача, його аватар, знизу зліва – поле для фідбеку, який можна відіслати розробнику на його акаунт. У правій частині є кнопки «Options», «Profile» та «List of friends». Також відображається календар з поточною датою.

На наступному макеті зображено сторінку «List of friends».

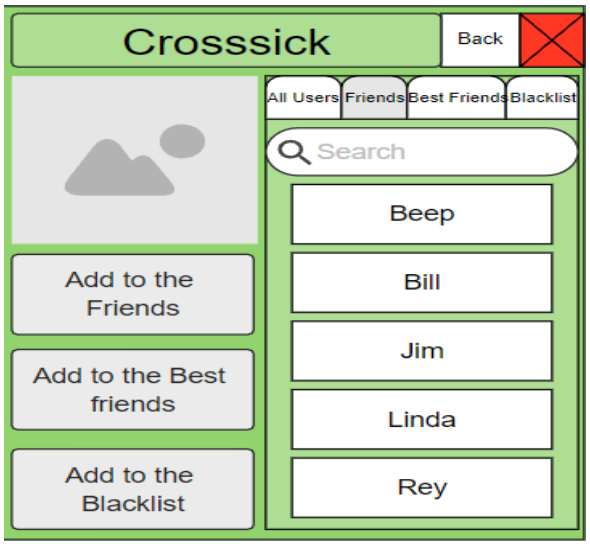


Рисунок 2.5 – Макет сторінки зі списком друзів

На макеті зображено список друзів, кнопки «Add to the Friends», «Add to the Best friends» та «Add to the Blacklist». Також на макеті видно, що можна перемикатися між «All Users», «Friends», «Best Friends» та «Blacklist» списками. Обравши одного з друзів, можна розпочати чатуватися з ним, що зображено на наступному макеті.

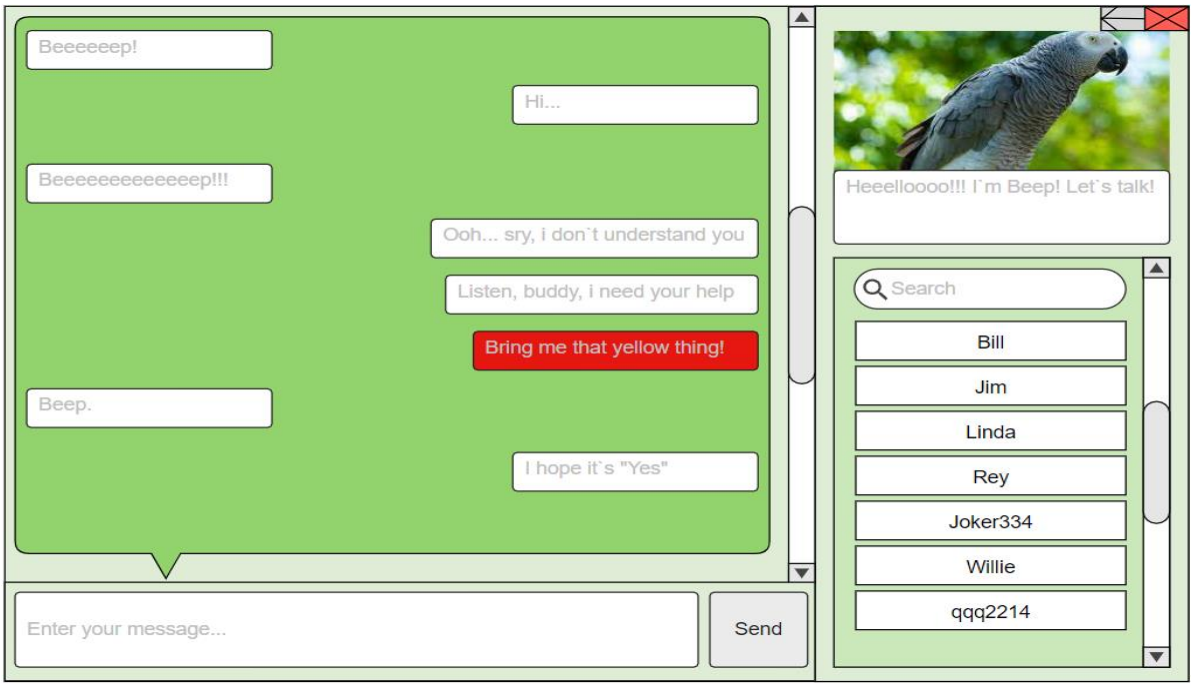


Рисунок 2.6 – Макет чату з другом

На цьому макеті зображено приклад чатування з другом. Повідомлення на червоному фоні означає, що відправник помітив його, щоб звернути особливу увагу на нього.

2.3 Проектування бази даних для програмного забезпечення обміну повідомленнями

База даних – організована колекція даних, яка є доступною та легко управляється. В ній можна організовувати дані в таблицях, рядках, колонках, і індексувати дані, щоб зробити легшим пошук необхідної інформації.

Головна мета бази даних – взаємодіяти з великою кількістю інформації, зберігаючи та оновлюючи її [9]. База даних відноситься до набору логічно пов’язаної інформації, організованої так, щоб її можна було легко отримати,

керувати та оновлювати. Доступ до баз даних зазвичай здійснюється в електронному вигляді з комп’ютерної системи і зазвичай контролюється системою управління базами даних (СКБД).

Підприємства зазвичай використовують як внутрішні бази даних, так і зовнішні бази даних. Внутрішні бази даних зазвичай включають оперативні бази даних і сховища даних. Зовнішні бази даних відносяться до баз даних, зовнішніх для організації, і, як правило, доступ до них здійснюється через Інтернет і належать іншим організаціям [10]. Система управління базами даних – це добре відомий термін в аналізі даних. Він відноситься до набору програм, які дозволяють користувачам маніпулювати, підтримувати, звітувати та зв’язувати дані. СКБД часто використовується для зменшення надлишковості даних, контрольованого обміну даними та зменшення проблем із цілісністю даних. СКБД не є інформаційною системою, а є просто програмним забезпеченням. Реляційна модель, яка зберігає дані в табличному форматі, є найбільш широко використовуваною СКБД. Реляційна СКБД організовує інформацію в рядки, стовпці та таблиці, що полегшує пошук відповідної інформації. Реляційні бази даних популярні, оскільки їх легко розширити, а нові категорії даних можна додавати після створення вихідної бази даних без великих змін. Основними компонентами бази даних є:

— **Апаратне забезпечення**, що складається з набору фізичних електронних пристроїв, таких як пристрої введення/виводу, пристрої зберігання даних та багато іншого. Він також забезпечує інтерфейс між комп’ютерами та реальними системами.
— **Програмне забезпечення**, що являє собою набір програм, які використовуються для контролю та керування загальною базою даних. Він також включає саме програмне забезпечення СКБД. Операційна система, мережеве програмне забезпечення, яке використовується для обміну даними між користувачами, прикладні програми, що використовуються для доступу до даних в СКБД.

— **Дані**. Система управління базою даних збирає, зберігає, обробляє та отримує доступ до даних. База даних містить як фактичні або оперативні дані, так і метадані.

Створення мессенджеру без використання бази даних є неможливим, адже потрібно як мінімум зберігати та систематизовувати інформацію стосовно акаунтів користувачів, їх статус в системі, тощо.

Діаграма зв’язків між об’єктами (ERD) – це знімок структур даних. Діаграма зв'язків сутностей показує сутності (таблиці) в базі даних і зв'язки між таблицями в цій базі даних. Для хорошого дизайну бази даних важливо мати діаграму взаємозв’язків сутностей [11]. Вони широко використовуються для проектування реляційних баз даних. Сутності в схемі ER стають таблицями, атрибутами і перетворюються в схему бази даних. Оскільки їх можна використовувати для візуалізації таблиць бази даних та їх зв’язків, вони також зазвичай використовуються для усунення несправностей бази даних. Діаграми взаємовідносин сутностей використовуються в програмній інженерії на етапах планування проекту програмного забезпечення. Вони допомагають визначити різні елементи системи та їх взаємозв’язки між собою. Часто використовується як основа для діаграм потоків даних.

Як можна створити ER діаграму:

— Потрібно визначити всі сутності в системі. Сутність має з’являтися лише один раз на конкретній діаграмі. Створити прямокутники для всіх об’єктів і правильно назвати їх.
— Визначити зв’язки між сутностями. З’єднати їх за допомогою лінії і додати ромб посередині, що описує відносини.
— Додати атрибути для сутностей. Дати змістовні назви атрибутів, щоб їх було легко зрозуміти.

Хорошою практикою буде надати точне й відповідне ім’я для кожної сутності, атрибута та зв’язку на діаграмі. Прості й знайомі терміни завжди перевершують нечіткі, технічні слова. Називаючи об’єкти, краще

використовувати іменники в однині. Однак прикметники можуть використовуватися для розрізнення суб’єктів, що належать до одного класу. У той же час імена атрибутів повинні бути значущими, унікальними, системно-незалежними та легко зрозумілими. Можна видалити нечіткі, зайві або непотрібні зв’язки між сутностями. Ніколи не зв'язувати зв'язки з іншими зв'язками. Ефективно використовуйте кольори. Ви можете використовувати кольори для класифікації подібних об’єктів або для виділення ключових областей у ваших діаграмах.

Діаграми ER є дуже корисною структурою для створення та маніпулювання базами даних. По-перше, діаграми ER легко зрозуміти і вони не вимагають від людини ретельного вивчення, щоб мати можливість працювати з ними ефективно та точно. Це означає, що дизайнери можуть використовувати діаграми ER для легкого спілкування з розробниками, клієнтами та кінцевими користувачами, незалежно від їхнього рівня знань.

По-друге, діаграми ER легко перекладаються в реляційні таблиці, які можна використовувати для швидкого створення баз даних. Крім того, діаграми ER можуть безпосередньо використовуватися розробниками баз даних як план для впровадження даних у конкретні програмні застосунки. Нарешті, діаграми ER можуть застосовуватися в інших контекстах, наприклад, для опису різних відносин і операцій в організації [12]. Нижче зображено ER діаграму, створену спеціально для розроблюваного застосунку.

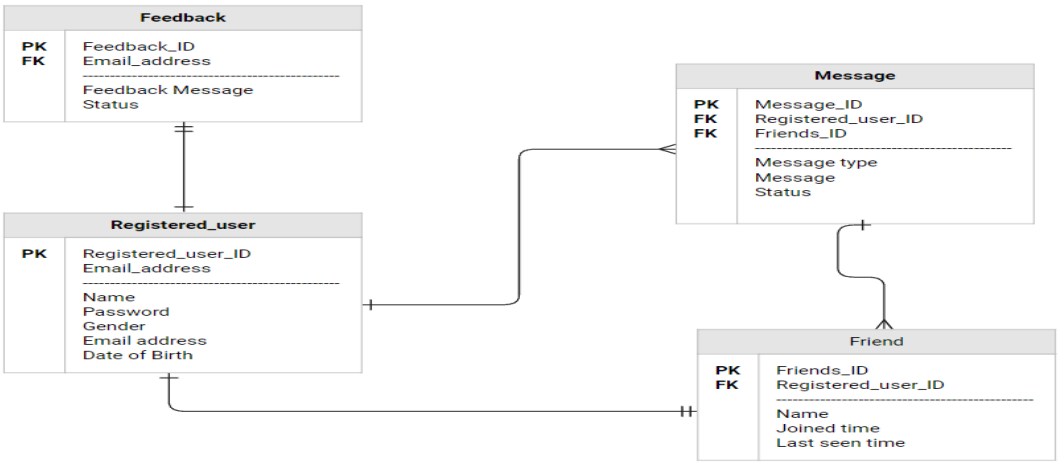


Рисунок 2.7 – ERD діаграма

Висновки до розділу 2

Під час роботи на другим розділом КРБ, було проаналізовано що таке діаграма прецедентів та її ключові компоненти. Створено діаграму прецедентів для програмного забезпечення обміну повідомленнями. Також було пояснено що таке макет, його основні властивості і для чого він потрібен. Для мессенджеру було створено декілька макетів, щоб мати уявленням про основні характеристики майбутнього проекту. У третьому підрозділі другого розділу розглянуто поняття бази даних та ER діаграми. Для подальшої розробки мессенджера було створено відповідну ER діаграму.

3 ВИБІР ЗАСОБІВ РОЗРОБКИ СИСТЕМИ ОБМІНУ ПОВІДОМЛЕННЯМИ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

3.1 Вибір технологій розробки клієнтської частини системи обміну повідомленнями

Варіантів вибору технологій, за допомогою яких можна створити клієнтську частину програмного забезпечення обміну повідомленнями в режимі реального часу досить багато. Перш за все потрібно розуміти різницю між Front-End і Back-End мовами програмування. Front-End мова програмування – це мова, яка використовується для створення клієнта чату. Простіше кажучи, користувальницький інтерфейс, який клієнти використовуватимуть для надсилання повідомлень, побудований на мові програмування інтерфейсу. Back-End мова програмування, з іншого боку, має справу з усім за лаштунками. Вона підтримує код на стороні сервера, який обробляє аутентифікацію, маршрутизацію, API тощо.

Мови програмування інтерфейсу та сервера можуть бути однаковими. Хорошим прикладом цього є JavaScript, який може працювати як у браузері, так і на сервері. Однак у більшості випадків дві мови будуть різними. Мова інтерфейсу значною мірою залежатиме від основної платформи, яку ви виберете для свого мессенджеру, оскільки деякі платформи підтримують лише певні мови. Back-End мова майже завжди не залежить від платформи, і її слід вибирати на основі її функцій оптимізації, а також знайомства ваших розробників.

Тепер, коли описано різницю між Front-End та Back-End мовами, можна описати Front-End мови, які можна використовувати для свого мессенджеру. Як згадувалось вище, Front-End мова, значною мірою залежатиме від того чи платформа або платформи підтримують її. Android, iOS та веб мають свою власну мову програмування. Це означає, що якщо потрібно підтримувати всі три платформи, то є два варіанти. Можна написати та підтримувати три різні кодові бази, або можна вибрати одне з кросплатформних рішень. Якщо

потрібна підтримка кількох платформ, то рекомендуєтья зосередитися на кросплатформних рішеннях. Якщо не має хорошого фінансування, три власні рішення не варті витрат.

Розглянемо Front-End мови для операційної системи Android.

Java – оригінальна мова програмування для створення додатків для Android, Java існує з 1996. Ця мова також користується популярністю за межами розробки додатків. Багато застарілих банківських і розважальних систем побудовано на Java. І, оскільки, це мова об’єктно-орієнтованого програмування, вона часто використовується в університетах для навчання концепціям інкапсуляції, абстракції, успадкування та поліморфізму. Переваги використання Java:

- об'єктно-орієнтована мова програмування, яка дозволяє створювати модульні компоненти та багаторазовий код;
- незалежна від платформи, що означає, що може працювати на будь-якому типі сервера.
- Мессенджери, створені за допомогою Java:
- android-клієнт Twitter створений за допомогою Java;
- Signal, одна з найпопулярніших програм для зашифрованого чату, використовує Java для свого клієнта Android.

Kotlin – не тільки офіційна мова Android, Kotlin також є улюбленою мовою Google для розробки на Android. в той час як Java є об'єктно-орієнтованою мовою програмування, Kotlin має функції як ООП, так і функціонального програмування. Це робить його чудовою мовою, яку можна спробувати, якщо ви займаєтеся функціональним програмуванням. Ще одна причина, чому Google віддає перевагу Kotlin, полягає в тому, що він статично типізований і більш лаконічний, ніж Java. Це означає, що ви зможете писати менше коду, і ваш код матиме меншу ймовірність помилок [13].

Однією з найкорисніших функцій Kotlin є його 100% взаємодія з Java - Kotlin може викликати функції Java, а Java може викликати функції Kotlin.

Тобто є можливість створити програму, яка буде наполовину з Java, наполовину з Kotlin, якщо потрібно. Переваги використання Kotlin:

– легко читати та розуміти;
– «нуль-безпечний», що означає, що ваша програма не завершується аварійним завершенням через винятки нульового покажчика;
– незмінний, що знижує ймовірність помилок розробника;
– сумісний з Java, що дозволяє створити гнучке середовище розробки.

IOS, як і Android, має власні унікальні мови програмування для своєї платформи. Якщо є необхідність створити вбудований додаток для чату для iOS, доведеться обирати між двома основними мовами: Objective-C і Swift.

Objective-C – це найстаріша мова, мова, яку Apple взяла за основу, перш ніж створити власну мову. Як і C++ та C#, Objective-C заснований на мові програмування C. Він додає об'єктно-орієнтовані можливості до C.

Переваги використання Objective-C:

– виразний синтаксис Objective-C дозволяє легко розуміти код;
– об'єктно-орієнтована мова програмування, яка дозволяє створювати модульні компоненти та багаторазовий код.
– Програми чату, створені за допомогою Objective-C:
– додаток WhatsApp для iOS створено за допомогою Objective-C;
– майже всі програми iOS, створені до 2014 року, тобто до створення Swift, мають принаймні частину своєї кодової бази, створеної за допомогою Objective-C.

Swift – мова програмування, що значніше легша за Objective-C. Swift – чудовий вибір для створення нативних програм iOS.

Переваги використання Swift:

– працює швидше, ніж Objective-C;
– статично типізований, що зменшує кількість помилок розробника;
– запобігає переповнення стеку та керує пам’яттю, автоматично;
– повністю з відкритим кодом;
– сумісний з Objective-C.

За допомогою Swift, Telegram повністю перебудував свій додаток для iOS.

Далі розглянемо веб мови програмування для створення клієнтської частини. Javascript, по суті, єдиний варіант для цього завдання, проте є декілька фреймворків Javascript, що роблять розробку швидшою та легшою.

Звичайний **JavaScript** завжди підходить для створення веб-чату, хоч це і не найкращий варіант. Переваги використання JavaScript:

– відносно простий і легкий для вивчення;
– надзвичайно популярний і поширений;
– дуже універсальний, можна створювати розширені інтерфейси, використовуючи виключно JavaScript.

React – один з найпопулярніших JS-фреймворків. Його зосередженість на створенні модульних компонентів інтерфейсу користувача дозволяє легко створити прототип веб-застосунку. Існує також багато бібліотек з відкритим кодом готових до використання компонентів React. Переваги використання React:

– архітектура компонентів дозволяє повторно використовувати код;
– легко зрозуміти та навчитися, якщо є досвід роботи з JavaScript;
– можна легко перенести на React Native для мобільної розробки.

Angular – це чудовий вибір для створення веб-чату. Він пропонує стандартизовану структуру програми, яка забезпечує узгодженість вашого проекту. Як і React, існує кілька бібліотек готових до використання, добре перевірених компонентів інтерфейсу, які можна використовувати для прискорення розробки. Цікавою особливістю Angular є те, що він створений за допомогою TypeScript, що дуже типізований. Перевага сильно типізованої мови полягає в тому, що вона допомагає розробникам зберігати свій код чистим і читабельним. Легше налагоджувати та знаходити типові помилки. Переваги використання Angular:

– велика спільнота розробників;
– архітектура компонентів дозволяє повторно використовувати код;
– може відкладати завантаження модулів, створити легкий додаток;

– багато попередньо створених бібліотек дизайну для прискорення розробки.

Варто виділити окрему групу мов програмування, які підійдуть для створення клієнтської частини, що буде кросплатформною.

React Native – це фреймворк, створений на основі React, який націлений на API основної платформи. Оскільки React Native обертається навколо рідного інтерфейсу користувача, та сама база коду React компілюється у дві власні програми. Основним недоліком React Native є те, що він призначений для підтримки мобільних додатків і не компілюється у веб-програму. Отже, якщо потрібно підтримувати всі три платформи, доведеться створити та підтримувати кодову базу React Native для мобільних платформ і кодову базу React для веб-платформи. При цьому багато з найпопулярніших програм для чату та соціальних мереж використовують React native для своїх мобільних додатків. Facebook, Pinterest, Instagram і Skype використовують React Native [14]. Переваги використання React Native:

– дозволяє повторно використовувати код на мобільних платформах;
– дуже велика спільнота розробників;
– легко освоїти, знаючи JavaScript;
– модульна архітектура компонентного стилю дозволяє легко повторно використовувати код.

Чати, створені за допомогою React Native:

– мобільні програми Skype використовують React Native;
– Discord використовує React Native для своєї програми iOS;
– Instagram використовує React Native для своїх програм для iOS і Android.

Ionic – це дуже цікавий і універсальний кросплатформний фреймворк. Він не залежить від фреймворку, що означає, що його можна використовувати з React, Angular, Vue або навіть зі звичайним JavaScript. На відміну від React Native, Ionic компілює кодову базу в гібридну програму для магазину застосунків і прогресивну веб-програму для Інтернету. Гібридний додаток – це

веб-додаток, вбудований у мобільний додаток. Він виконує один і той же код, незалежно від платформи. Переваги використання Ionic:

– постачається з набором рідних сумісних інструментів, що надають доступ до функцій мобільного пристрою;
– front-end agnostic, що означає, що ви можете використовувати React Native, React, Angular, Vue або будь-який інший фреймворк;
– велика спільнота розробників.

3.2 Вибір технологій розробки серверної частини системи обміну повідомленнями

Мова на стороні сервера не обмежена клієнтською платформою. Це означає, що теоретично можна використовувати будь-яку внутрішню мову для роботи мессенджера. Проте є деякі важливі функції коду, які роблять певні мови серверної системи кращими, ніж інші. Повідомлення повинні бути для користувача миттєвими, код повинен обробляти багато одночасних з'єднань, більшість виконання коду визначається діями користувачів.

Розглянемо найбільш вдалі мови програмування для написання серверної частини програмного забезпечення обміну повідомленнями в режимі реального часу.

Erlang – це функціональна мова програмування загального призначення, розроблена для паралельності. Він може обробляти багато процесів, що виконуються одночасно. Це робить його чудовою мовою для створення великої, масштабованої програми чату. Завдяки його паралельності команда WhatsApp обрала Erlang для свого одноіменного застосунку. Переваги використання Erlang:

– пропонує паралельність за рахунок використання полегшених потоків процесів;
– масштабується як по вертикалі, додавши більше ядер, так і по горизонталі, додавши більше машин;
– надійний, оскільки може перезапустити процеси, які аварійно завершуються через помилки.

– Додатки для чату, створені за допомогою Erlang:
– WhatsApp використовує Back-End на основі erlang для свого мессенджеру;
– клієнт чату Facebook працює на сервері erlang.

Javascript може працювати як на стороні сервера, так і на стороні клієнта. Для серверної версії Javascript потрібне середовище Node. Javascript чудово підходить для виконання коду, керованого подіями, і шаблонів у реальному часі через такі фреймворки, як WebSocket. Однією з найбільших переваг Javascript є те, що це одна з найпопулярніших мов програмування і постійно оновлюється. Переваги використання Node.js, він же javascript для серверів:

– пропонує неблокуючий ввод-вивод і асинхронну обробку запитів;
– багата екосистема плагінів, фреймворків і відкритого коду;
– дуже швидка обробка завдяки двигуну V8.
– LinkedIn в основному використовує Back-End Node.js для свого мобільного застосунку.

Go, також відомий як Golang, був створений Google. Він використовує простий для читання синтаксис для створення високопродуктивних програм. Переваги використання Go:

– повністю з відкритим кодом
– автоматичний збір сміття робить керування пам’яттю нескладним
– пропонує паралельність використання кількох ядер

Деякі функції серверної частини Slack написані на Go. Список новин Twitter у серверній частині працює на основі Go.

3.3 Вибір засобів реалізації інформаційного забезпечення технологій реалізації бази даних

Базу даних для створення мессенджеру можна обрати майже будь-яку, все залежить від конкретних завдань та специфіки розробки.

MongoDB використовується для багатьох мессенджерів. Основні переваги, пов’язані з моделлю документа, дозволяють плавно розширювати програму під час створення функціональних можливостей. MongoDB став вибором для багатьох популярних застосунків-чатів. MongoDB не має деяких функцій реляційної бази даних, як-от транзакцій або приєднання, але він має можливість легко масштабувати горизонтальну і гнучку схему, якою легко маніпулювати за допомогою формату даних JSON. MongoDB є потужним інструментом, якщо його правильно використовувати, але це не єдине рішення [15].

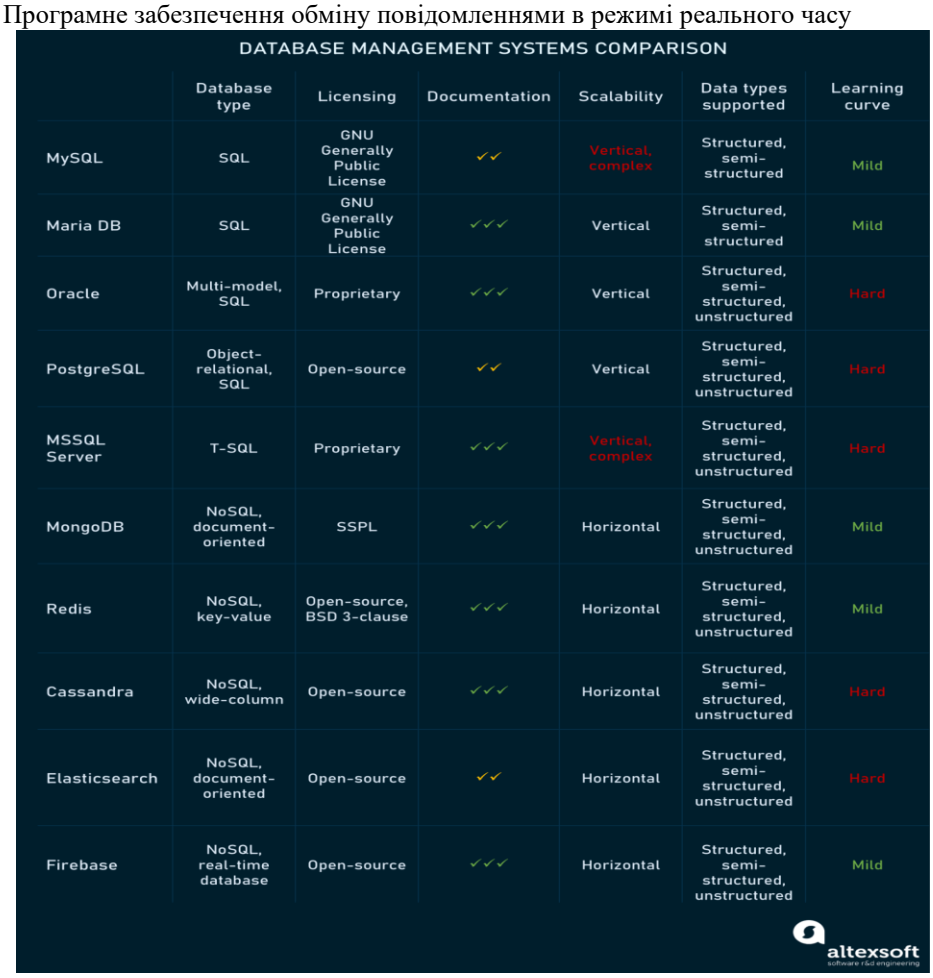
MySQL також хороший варіант для зберігання даних мессенджера на сервері. Швидкість цієї бази даних дуже висока через вбудовану багато потоковість. Код відкритий, тому можна при необхідності можна розширити функціонал за допомогою різноманітних пакетів. MySQL вважається однією з найшвидших доступних баз даних. Крім того, він також забезпечує багатопотоковість для досягнення більш оптимізованої продуктивності. Завдяки підтримці вбудованих програм MySQL підходить для різних випадків використання. Проте слід зазначити, що MySQL страждає від проблем зі стабільністю та має тенденцію до пошкодження в певних випадках використання. Хоча MySQL найкраще підходить для багатьох випадків використання, але для великих підприємств, що мають дуже багато записів – MySQL не підходить. Для такого великого обсягу, MySQL не забезпечує належної підтримки операцій читання/запису [16].

SQLite – це дуже популярна база даних, яка успішно використовується з форматом файлів на диску для настільних застосунків, таких як системи контролю версій, інструменти фінансового аналізу, пакети каталогізації та редагування медіа. SQLite – це дуже легка та зважена база даних, тому її легко використовувати як вбудоване програмне забезпечення [17]. База даних SQLite безпечна за замовчуванням, але це не означає, що вона повністю безпечна. Якщо потрібно зробити базу даних безпечною, слід розглянути можливість її шифрування. Однак неможливо повністю усунути ризик втрати даних, оскільки база даних не може бути повністю захищена від зовнішніх атак. Як результат,

SQLite найкраще підходить для невеликих баз даних з невеликою кількістю користувачів. Його низький рівень складності робить його ідеальним для однокористувацьких додатків, а його високорівневі функції безпеки роблять його кращим вибором для веб-застосунків. Однак його обмежена функціональність робить його непридатним для програм, які потребують детального контролю доступу. Крім того, SQLite не рекомендується використовувати у великомасштабних базах даних, які потребують великої кількості одночасних операцій читання/запису [18].

PostgreSQL – це система управління базами даних корпоративного класу з відкритим вихідним кодом. Він підтримує як SQL, так і JSON для реляційних і нереляційних запитів для розширюваності та відповідності SQL. PostgreSQL підтримує розширені типи даних і функції оптимізації продуктивності, які доступні лише в дорогих комерційних базах даних, таких як Oracle і SQL Server [19]. Ця система управління базами даних поділяє свою популярність з MySQL. Це об’єктно-реляційна СКБД, де визначені користувачем об’єкти та підходи до таблиць поєднуються для побудови більш складних структур даних. Крім того, PostgreSQL схожий з MySQL. Він спрямований на посилення стандартів відповідності та розширюваності. Отже, він може обробляти будь-яке робоче навантаження, як для одномашинних продуктів, так і для більш складних застосунків. Ця СКБД доступна для використання з такими платформними системами, як Microsoft, iOS, Android та багатьма іншими, що робить її досить вдалим рішенням для створення мессенджеру.

Розглянув декілька баз даних, можна сказати, що особливої різниці між вибором бази даних для програмного забезпечення обміну повідомленнями в режимі реального часу немає. Тобто вибір бази даних для цієї задачі обмежується іншими факторами, такими як: зручність, доступність, модель розповсюдження, написана документація та підтримкою тих чи інших типів даних. Нижче приведено порівняння найбільш популярних баз даних.



DATABASE MANAGEMENT SYSTEMS COMPARISON

	Database type	Licensing	Documentation	Scalability	Data types supported	Learning curve
MySQL	SQL	GNU Generally Public License	✓✓	Vertical, complex	Structured, semi-structured	Mild
Maria DB	SQL	GNU Generally Public License	✓✓✓	Vertical	Structured, semi-structured	Mild
Oracle	Multi-model, SQL	Proprietary	✓✓✓	Vertical	Structured, semi-structured, unstructured	Hard
PostgreSQL	Object-relational, SQL	Open-source	✓✓	Vertical	Structured, semi-structured, unstructured	Hard
MSSQL Server	T-SQL	Proprietary	✓✓✓	Vertical, complex	Structured, semi-structured, unstructured	Hard
MongoDB	NoSQL, document-oriented	SSPL	✓✓✓	Horizontal	Structured, semi-structured, unstructured	Mild
Redis	NoSQL, key-value	Open-source, BSD 3-clause	✓✓✓	Horizontal	Structured, semi-structured, unstructured	Mild
Cassandra	NoSQL, wide-column	Open-source	✓✓✓	Horizontal	Structured, semi-structured, unstructured	Hard
Elasticsearch	NoSQL, document-oriented	Open-source	✓✓	Horizontal	Structured, semi-structured, unstructured	Hard
Firebase	NoSQL, real-time database	Open-source	✓✓✓	Horizontal	Structured, semi-structured, unstructured	Mild

altexsoft

Рисунок 3.1 – Порівняння баз даних [20]

Висновки до розділу 3

Під час роботи над третім розділом кваліфікаційної роботи бакалавра було розглянуто Front-End та Back-End мови програмування, які можна використати для створення клієнтської та серверної частин програмного забезпечення обміну повідомленнями в режимі реального часу. Описано переваги та недоліки таких Front-End мов, як: Java, Kotlin, Objective-C, Swift, Javascript, React, Angular, React Native та Ionic. В якості Back-End мов було розглянуто Erlang, Javascript та Go. Також було описано декілька баз даних, що підходять для розробки мессенджеру. До їхнього списку входять: MongoDB, MySQL, SQLite, PostgreSQL. Наведено порівняння найпопулярніших баз даних у різних аспектах.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ОБМІНУ ПОВІДОМЛЕННЯМИ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

4.1 Реалізація програмних компонентів логіки системи обміну повідомленнями

Для реалізації програмного забезпечення обміну повідомленнями в режимі реального часу «TalkKing» було обрано такі технології, як: React, Node.js, mongoDB і Socket.io. Щоб створити новий проект React, потрібно виконати наступну команду:

create-react-app chatapp

Ця команда створить усі необхідні файли, завантажить усі необхідні зовнішні залежності та виконає всю роботу з налаштування за нас.

Node.js сервер – чудовий вибір, адже:

— По-перше, потрібен сервер, на якому можна зберігати та отримувати дані і повідомлення користувача;

— По-друге, потрібно реалізувати обмін повідомленнями в реальному часі, для цього потрібен веб-сокет;

— По-третє, для його реалізації потрібен лише знання Javascript.

Тепер необхідно створити проект Node.js, виконавши команду нижче:

npm init

Ця команда створить файл package.json, зберігаючи інформацію про проект. Тепер давайте подивимося на package.json, який я створив. Node.js буде зберігатися в server.js.

Сервер запускається в три кроки:

— По-перше, програма починається з виконання методу конструктора, коли ми створюємо клас Object of Server;

— По-друге, у метод appConfig() включаються всі конфігурації.;

— Останній крок: потрібно додати Routes і Socket.

І все це відбувається, коли запускається метод aapExecute(), створивши об’єкт сервера. Далі потрібно написати файл конфігурації. Основним файлом

конфігурації є файл app-config.js, куди записуються всі конфігурації, пов'язані з програмою, а також включається файл ExpressConfig. Всередині includeConfig() буде використовуватися bodyparser.

Для зв'язку з базою даних потрібно створити файл db.js у папці /config. Тут використовується модуль MongoDB. Для з'єднання з базою даних, потрібно викликати спеціальне посилання, яке автоматично створюєеться після встановлення mongoDB, рис. 4.1.

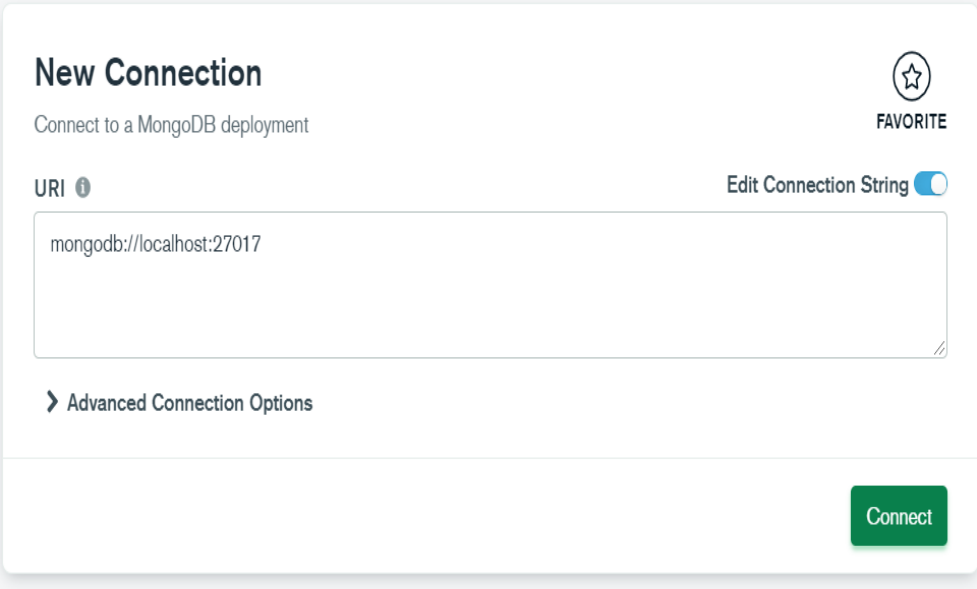


Рисунок 4.1 – URL для підключення mongoDB

Слід створити сторінку аутентифікації. У застосунку є вкладки для входу та реєстрації. Якщо натиснути на вкладку «Login», то з'явиться форма входу, а якщо натиснути сторінку «Create account», з'явиться форма реєстрації, рис. 4.2.

Тут використовую react-bootstrap для реалізації сторінок. У контейнері сторінок буде відображено компоненти входу та реєстрації. Після імпортування Tabs і Tab, було також імпортовано компоненти входу та реєстрації, щоб відобразити їх у контейнері Tab. Метод setRenderLoadingState() буде викликано з дочірнього компонента за допомогою React Props. Як результат, є контейнер з двома сторінками, щоб відображати компоненти входу та реєстрації. Для форми реєстрації можна використати такі компоненти, як Form, Form.Group, From.Controland Button, що є частиною бібліотеки react-bootstrap.

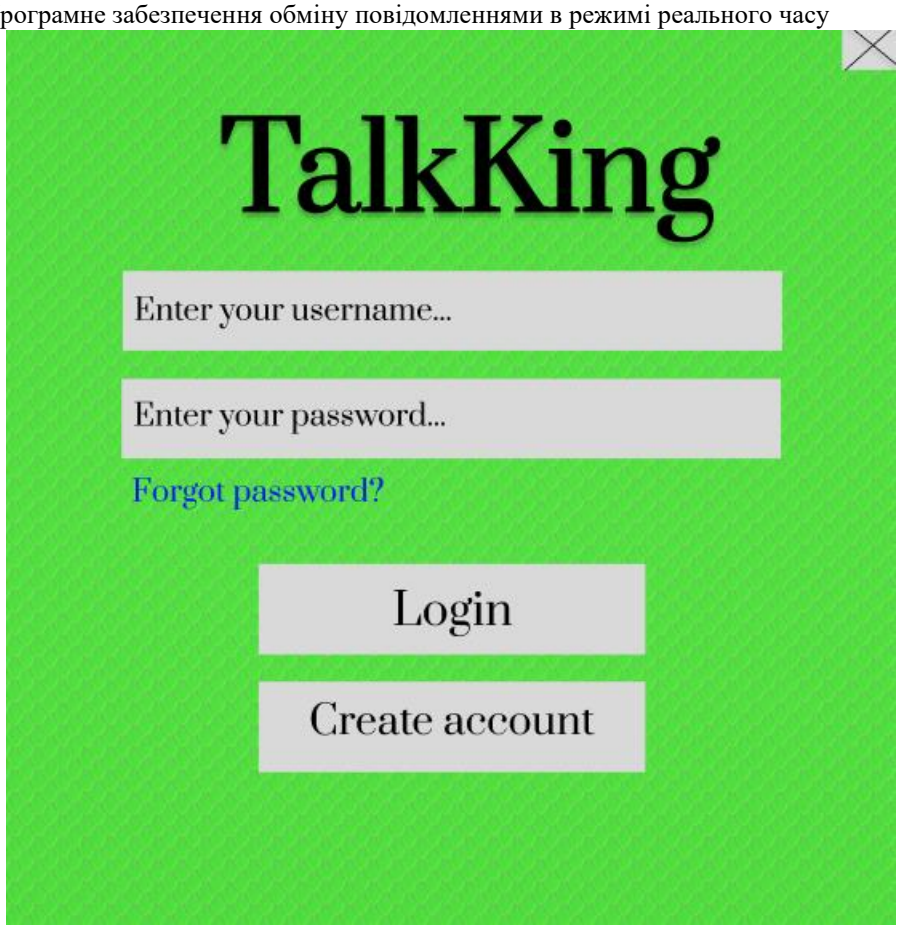


Рисунок 4.2 – Сторінка входу

Методи checkUsernameAvailability() і handleInputChange() оновлять стан з відповідними деталями. handleRegistration() викликає Make an Ajax, щоб зареєструвати нового користувача. Використовуватиметься клас ChatHttpServer для реєстрації нового користувача. Програма перенаправить користувача на домашню сторінку, як тільки користувач завершить процес реєстрації. Імпортовано всі необхідні компоненти із сторонніх бібліотек. Потім імпортовано об’єкт класу ChatHttpServer. В об’єкті state є дві властивості: ім’я користувача та пароль. Потрібно надсилати цей об’єкт стану безпосередньо на сервер під час реєстрації. Після успішної реєстрації користувач перенаправляється на /home сторінку. Крім того, буде зберігатися userId в локальному сховищі, щоб його можна було використовувати пізніше в програмі. Методи checkUsernameAvailability() і handleInputChange() оновлять ключі об’єкта стану відповідно. Якщо щось піде не так, з’явиться сповіщення з повідомленням про помилку.

Після цього потрібно написати кілька рядків коду всередині registerRouteHandler(), щоб просто обробити запит/відповідь і викликати метод, який просто вставляє дані в базу даних. RegisterRouteHandler() подбає про маршрут /register. Якщо з валідацією щось не так, буде надіслано правильний код помилки з повідомленням про помилку. Потім згенерується хеш пароля для пароля, введеного користувачем. І вже в кінці, якщо все вірно або точнішими словами, якщо запит дійсний, то буде зареєстровано користувача за допомогою виклику registerUser().registerUser() визначається всередині query-handler.js. За допомогою нього будуть додаватися записи в базу даних. Наступний фрагмент коду демонструє додавання користувача до бази даних:

```
try {
        const [DB,ObjectID] = await this.Mongodb.onConnect();
        DB.collection('users').insertOne(data, (err, result) =>{
          DB.close();
          if( err ){
            reject(err);
          }
          resolve(result);
        });
      } catch (error) {
        reject(error)
      }
```

Дуже важливо перед реєстрацією перевірити унікальність імені користувача. Очевидно, що не може бути двох користувачів з однаковим іменем користувача. Перш за все, потрібно написати розмітку в Registration.js, де просто використовується індикатор, який показує конкретне ім’я користувача. У наведеній нижче розмітці використано стан React, щоб приховати та показати повідомлення за допомогою властивості usernameAvailablestate класу компонента реєстрації.

```
<Alert className={{
   'username-availability-warning' : true,
   'visibility-hidden': this.state.usernameAvailable
}}  variant="danger">
```

{this.state.username} is already taken, try another username.

</Alert>

Також необхідно перевіряти унікальність імені користувача, зробивши HTTP-запит до сервера Node.js. Потім викликається ChatHttpServer.checkUsernameAvailability(), який в кінцевому підсумку робить HTTP-запит для перевірки унікальності імені користувача. І опираючись на відповіді сервера HTTP, буде встановлено значення властивості usernameAvailable стану. Також є userNameCheck() метод для перевірки кількості користувачів з такими самими іменами. Цей метод, в основному, перевіряє кількість даного імені користувача та поверне цю кількість у викликі.

Реалізація функції входу досить легка і майже ідентична до вже описаних вище функцій. Почати слід з розмітки, потім компонент, і в кінці пишеться частина з NodeJs. Компоненти Form,Form.Group,From.ControlandButton є частиною бібліотеки react-bootstrap. Методи handleInputChange() оновлять стан React з відповідними деталями. Коли імпортовано всі необхідні компоненти із сторонніх бібліотек, потрібно імпортувати об’єкт ChatHttpServerclass. В об’єкті state є дві властивості: ім’я користувача та пароль. Потрібно надіслати цей об’єкт стану безпосередньо на сервер під час входу. Після успішного входу користувач перенаправиться на домашню сторінку. Крім того, треба зберігати userId в локальному сховищі, щоб можна було використовувати його пізніше в програмі. Методи HandleInputChange() оновлять ключі об’єкта стану відповідно.

Щодо частини Nodejs, тут потрібно подбати про три речі. Спочатку додати маршрут. По-друге, додати метод для обробки запиту/відповіді маршруту і в кінці метод для виконання входу. Додамо маршрут:

this.app.post('/login', routeHandler.loginRouteHandler);

Коли /loginroute додано до відповідного файлу, треба написати метод для обробки відповіді на маршрут /login. Спочатку виконується перевірка, якщо все

добре, то фактично отримається результат з бази даних. Після перевірки отримається запис користувача, використовуючи його/її ім’я користувача. Якщо результат отримується із бази даних за допомогою імені користувача, то потрібно порівнювати пароль користувача за допомогою методу compareHash(). Після порівняння пароля, на основі результату складатиметься статус користувача онлайн.

Дуже важливо перевірити сеанс користувача, перш ніж дозволити користувачам спілкуватися в чаті та читати повідомлення. Щоб перевірити сеанс користувача, необхідно зробити HTTP-виклик. Щоразу, коли користувач відвідує домашню сторінку програми, перевіряється сеанс. Тут буде використовуватися метод життєвого циклу React, тобто componentDidMount(). У цьому методі потрібно створити HTTP-виклик, щоб перевірити сеанс користувача. В Home.js це можна описати таким чином:

```
async componentDidMount() {
   try {
      this.setRenderLoadingState(true);
      this.userId = await ChatHttpServer.getUserId();
      const response = await ChatHttpServer.userSessionCheck(this.userId);
      if (response.error) {
         this.props.history.push(`/`)
      } else {
         this.setState({
            username: response.username
         });
      }
      this.setRenderLoadingState(false);
   } catch (error) {
      this.setRenderLoadingState(false);
      this.props.history.push(`/`)
   }
}
```

У цьому методі спочатку викликається метод setRenderLoadingState() для відображення екрана завантаження, поки перевіряється сеанс користувача, зробивши виклик HTTP. Щоб перевірити сеанс користувача, потрібен

ідентифікатор користувача цього користувача. Його можна отримати за допомогою методу getUserId(), який визначено всередині служби ChatHttpServer. Після того, як отримаємо ідентифікатор користувача, можемо викликати метод theuserSessionCheck(), за допомогою якого зробимо HTTP-виклик, щоб перевірити, чи ввійшов користувач чи ні. Якщо у відповіді сказано, що користувач увійшов в систему, встановимо ім’я користувача в стані нашого компонента. Якщо користувач не ввійшов у систему, перенаправимо користувача на домашню сторінку. Тепер додамо маршрут для служби перевірки сеансу у файл routes.js. Тут перевіримо онлайн-статус користувача і на основі онлайн-статусу визначимо сеанс цього користувача.

```
this.app.post('/userSessionCheck',
routeHandler.userSessionCheckRouteHandler);
```

Тепер, щоб перевірити онлайн-статус у базі даних, доведеться написати метод userSessionCheck() у класі QueryHandler. Цей метод в основному перевіряє онлайн-статус, виконавши запит MongoDB.

Щодо підключення React застосунку до Socket серверу, Перше, на що слід звернути увагу, потрібно зчитати параметр userId за допомогою socket.request._query['userId']. По-друге, є метод addSocketId() для оновлення даних у MongoDB. І в кінці треба викликати решту подію сервера сокетів за допомогою методу this.socketEvents(). Створення списку чатів включає в себе наведені нижче пункти:

– створення компонента ChatList;
– написання розмітки для списку чату;
– клас компонента WritingChatList;
– запис події сокет-сервера;
– оновлення онлайн-статусу користувача в ChatList;
– вибір користувача зі списку чату користувача.

Наступний крок – це властивість chatListUsers класу ChatList Component. У властивості chatListUsers будемо зберігати список користувачів. Ця змінна є

не що інше, як масивом, який оновлюється за певних умов. Спочатку оновимо список чатів, а по-друге, передамо дані вибраного користувача зі списку користувачів у компонент Conversation. За допомогою методуgetChatList() оновлюємо список користувачів, визначених у службі ChatSocketService. Наприклад, якщо новий користувач увійде в систему або наявний користувач переходить в автономний режим – в обох випадках, оновлюємо список онлайн-користувачів. Тут передаємо повний список онлайн/офлайн користувачів із сервера сокету nodejs користувачеві, який щойно увійшов у систему. А для решти користувачів ті, хто вже в мережі, отримають інформацію про користувачів, які перейшли в автономний режим або щойно увійшли. Якщо відповідь від події socket містить властивість singleUser, що означає, що встановлено нове з’єднання сокету. Якщо відповідь містить властивість userDisconnected, це означає, що користувач перейшов у режим офлайн і видалив його зі списку чату. І зрештою, в умові else оновимо список користувачів у блоці else.

Все вищеописане буде працювати, якщо є сервер Socket, який відповідає на оновлений список чатів. Тут пишемо socket.io.on listener на стороні сервера, коли клієнт запитує його. Як тільки React запитає список чату на сервері Socket, сокет запитує деталі з бази даних. Після запиту на основі користувача він видає результат. Перш за все, виконується перевірка. Якщо все добре, процес, що йде вперед, видає відповідь з повідомленням про помилку. Всередині блоку try-catch виконаємо всі випадки. Try-catch призначений для обробки невідомих винятків і необроблених відхилень обіцянок. Це хороша практика, якої слід дотримуватися. Потім всередині блоку try-catch спочатку отримуємо інформацію про користувача, викликавши getUserInfo(). Пізніше отримаємо весь getChatList(), який визначено всередині класу QueryHandler. Події з двома сокетами необхідні, щоб не надсилати весь список чату користувачеві, а надсилати лише інформацію про користувача, який перебуває в режимі онлайн/офлайн. На основі параметра socketId проектуємо результат з бази даних. Останній метод, який залишився, – отримати список чатів із бази даних.

Просто отримуємо всіх користувачів із бази даних і відображаємо їх у інтерфейсі на основі статусу онлайн/офлайн. Також необхідно просто виділяти вибраного користувача зі списку чату. Метод this.state.selectedUserId поверне true або false, і за допомогою цього покажемо поточного вибраного користувача. Порівнявши вибраний користувачем userId з ідентифікатором користувача. Метод selectedUser() оновить властивість selectedUserIdstate. Крім того, цей метод видаватиме останній вибраний ідентифікатор користувача. Крім того, цей метод викличе метод props, тобто updateSelectedUser, який надасть поточного вибраного користувача його батьківському компоненту. Компонент ChatList є окремим компонентом, а компонент Conversation буде окремим компонентом. Отже, тут потрібно передавати дані між двома компонентами-побратимами. Тому спочатку створимо компонент Conversation, а потім будемо відображати повідомлення та робити інше.

Створимо компонент Conversation. Додамо файл talk.js у папці /home.

```
import React, { Component } from 'react';
import './Conversation.css';
class Conversation extends Component {
 render() {
   return (
     <>
       Conversation Component works!
     </>
   );
 }
}
export default Conversation;
```

Коли обирається будь-який користувач зі списку чату, треба відобразити чат між користувачами. Потрібно передати дані з компонента Chatlist до компонента Conversation. В ChatList.js є selectedUser().

```
selectedUser = (user) => {
   this.setState({
      selectedUserId: user.id
   });
```

```
    this.props.updateSelectedUser(user)
}
```

У першому рядку призначено ідентифікатор вибраного користувача зі списку чату властивості selectedUserId. Другий рядок надсилає дані іншому компоненту. Тут будемо використовувати параметри та стани для надсилання даних із компонента ChatList та отримання даних у компоненті Conversation відповідно. Використовуємо звичайні параметри і стани для передачі даних між компонентами. Щоб отримати дані від компонента ChatList, спочатку потрібно отримати дані в компоненті Home. Тут створимо стан у компоненті Home, який утримуватиме поточного вибраного користувача. Цей стан буде оновлено функцією, яка буде передана як props у компоненті ChatList. Створимо новий метод під назвою updateSelectedUser() у компоненті Home і додамо до нього наведений нижче код:

```
updateSelectedUser = (user) => {
    this.setState({
        selectedUser: user
    });
}
```

Далі необхідно зробити HTTP-виклик, щоб отримати чат між користувачами із сервера. Тому для цього будемо використовувати метод getMessages(). Цей метод визначено всередині класу ChatHttpServer. Щоразу, коли чат отримає дані нового користувача, він буде викликати службу HTTP, щоб отримати чат між двома користувачами. Для цього будемо використовувати два методи життєвого циклу компонентів getDerivedStateFromProps() і ComponentDidUpdate() відповідно. Тепер, щоб перевірити оновлені дані користувача, використовуємо getDerivedStateFromProps(), який оновлює стан оновленими даними, отже, компонент буде повторно відтворюватися. Після того, як компонент викличе метод render(), відразу після виконання методу ComponentDidUpdate(). У цьому методі зробимо HTTP-виклик, щоб отримати чат між двома користувачами.

```
componentDidUpdate(prevProps) {
  if (prevProps.newSelectedUser === null || (this.props.newSelectedUser.id
!== prevProps.newSelectedUser.id)) {
    this.getMessages();
  }
}
```

Метод ComponentDidUpdate() надасть попередній Props, тобто можна отримати нові props у властивості props виклику. Використовуючи ці два параметри, можемо визначити, чи ввійшов у систему користувач вибрав іншого користувача у списку чату. Після отримання оновлених даних користувача викличемо метод getMessages() класу. У методі getMessages() класу будемо використовувати ChatHttpServerclass. Щойно отримаємо нові повідомлення, метод getMessages() оновить стан і відобразить чат між двома користувачами. Викликаємо getMessages(), який визначено всередині класу ChatHttpServerclass. Як тільки отримаємо відповідь від сервера, то оновимо стан за допомогою властивості conversations. Метод ThescrollMessageContainer прокручує вміст донизу. Перевіривши параметри запиту сервер повертає відповідь. Потім всередині блоку else отримуємо повідомлення з MongoDB, викликаючи метод getMessages(). Якщо все добре, повертаємо відповідь на повідомлення.

Компонент Conversation буде накладений, доки не обереться будь-який користувач зі списку чату. Після того, як буде обрано користувача зі списку чату, покажемо повідомлення про це. Потім на основі об’єкта стану чату, отримаємо відповідний інтерфейс користувача. this.getMessageUI() поверне чат у формі повідомлень між двома користувачами, а this.getInitiateConversationUI() поверне заповнювач, якщо жодного користувача не вибрано зі списку чату. Створимо метод для прослуховування вхідних подій сокета. Після отримання події socket, помістимо нові повідомлення в масив станів Conversation.

4.2 Тестування системи обміну повідомленнями

Перед релізом будь-який застосунок необхідно протестувати на працездатність. При реєстрації користувача, спробуємо вписати неправильний пароль в «Confirm password» поле.

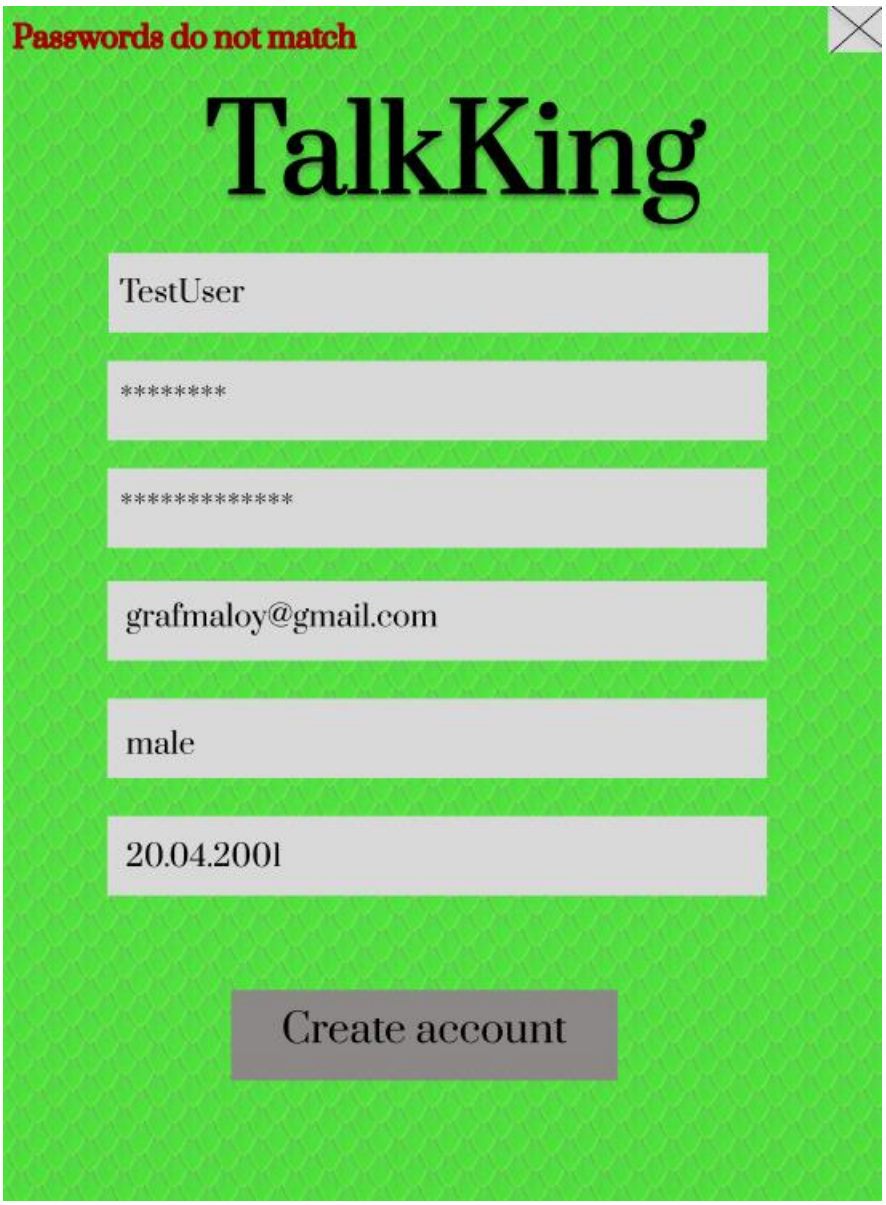


Рисунок 4.3 – Паролі не зівпадають

Як результат, користувач не був зареєстрований, а мессенджер вивів нам повідомлення «Passwords do not match», що вказує на різні паролі.

Спробуємо також, вже на логін сторінці, ввести неправильний пароль.

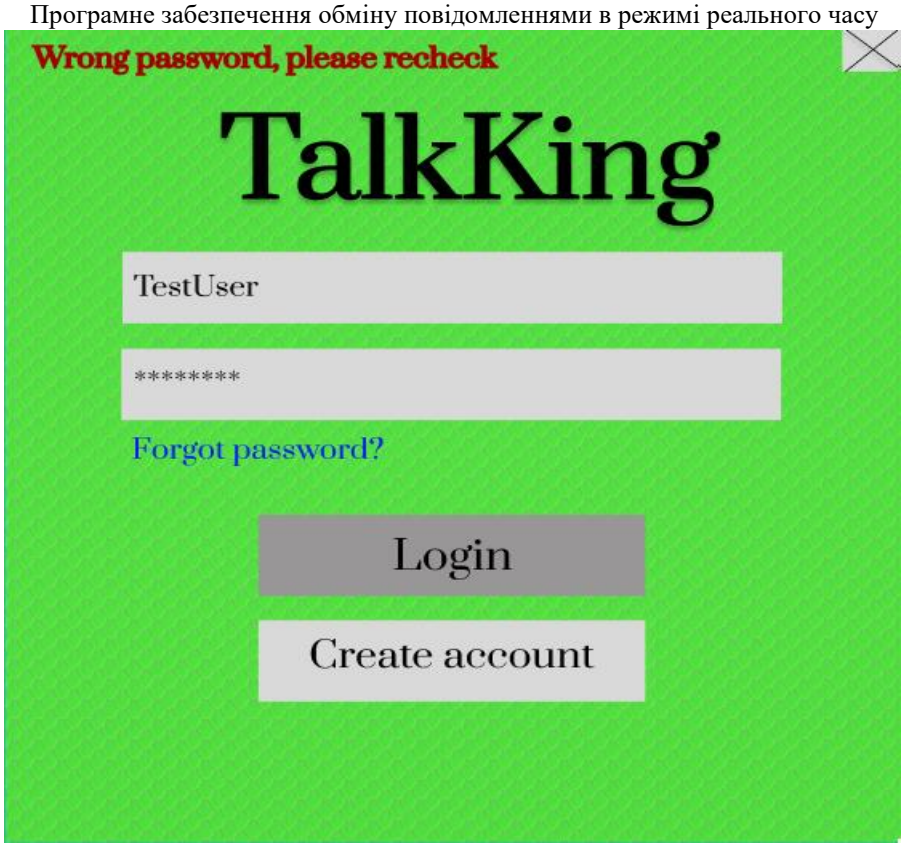


Рисунок 4.4 – Неправильний пароль

Як результат, було отримано повідомлення «Wrong password, please recheck», що вказує на неправильний пароль та застосунок просить користувача перевірити ще раз.

Теперь оберемо TestFriend зі списку друзів та відкриємо з ним чат:

Рисунок 4.5 – Чат з «TestFriend»

Необхідно протестувати найголовнішу функцію мессенджеру – роботу чата. Напишемо повідомлення одному з друзів.

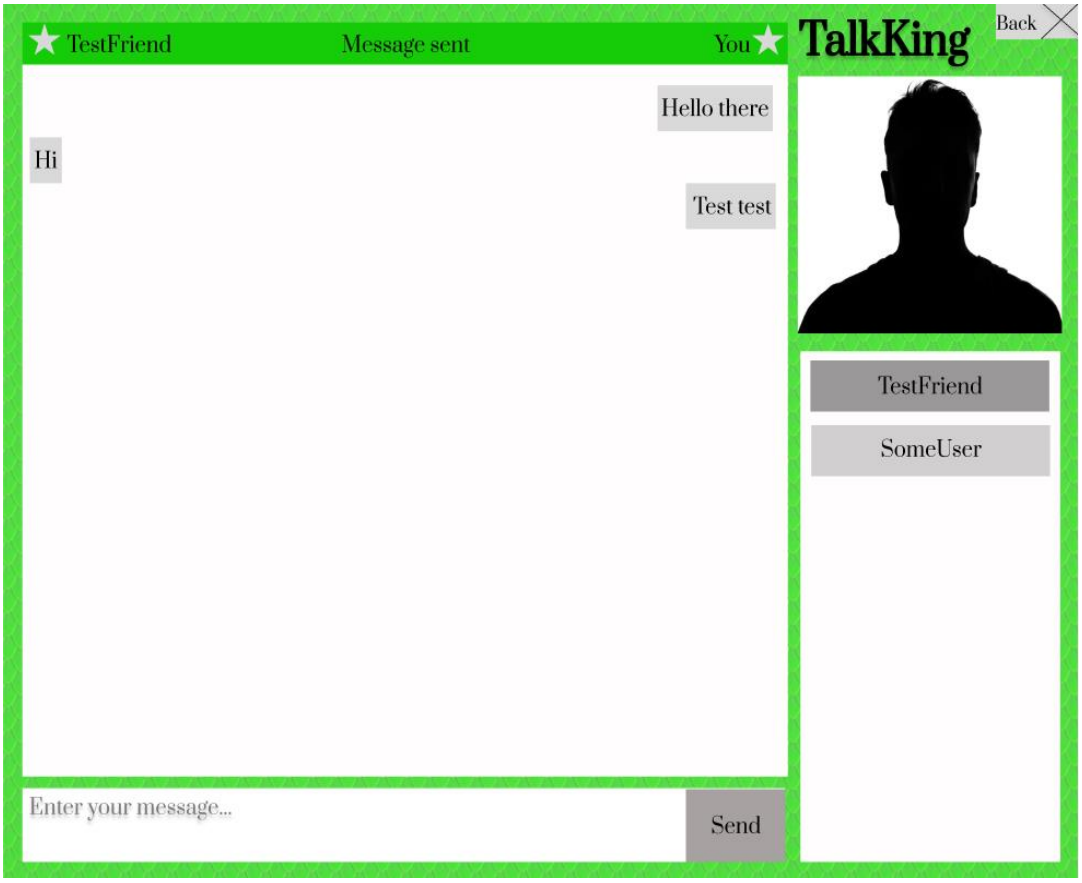


Рисунок 4.6 – Message sent

Повідомлення було успішно відправлено, про що свідчить напис зверху «Message sent».

Висновки до розділу 4

Під час роботи над четвертим розділом кваліфікаційної роботи бакалавра було описано реалізацію програмних компонентів логіки програмного забезпечення обміну повідомленнями в режимі реального часу «TalkKing». Представлено фрагменти коду важливих для працездатності функцій та створено мессенджер з використанням таких технологій, як React, Node.js, mongoDB та Socket.io. Протестовано основні сценарії використання системи обміну повідомленнями, а саме реєстрацію користувача, вхід в систему, вибір чату та відправку повідомлення.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було досягнуто поставленої мети, тобто підвищення зручності процесу комунікації учасників інтерактивного спілкування за рахунок створення системи обміну повідомленнями в режимі реального часу «TalkKing».

Проаналізовано існуючі аналоги систем обміну повідомленнями, такими, як: WhatsApp, Telegram, Facebook Messenger, Viber, Skype і Sluck. Визначено їх переваги та недоліки. На основі аналізу схожих застосунків, було визначено функціональні вимоги до створюваного застосунку.

Розроблено діаграму використання, ER-діаграму та макети інтерфейсу мессенджеру задля попереднього розуміння положення елементів інтерфейсу системи.

Протягом аналізу інтернет джерел, складено список технологій, які можна використати при розробці програмного забезпечення обміну повідомленнями в режимі реального часу, визначено їх переваги та недоліки.

Розроблено клієнтську та серверну частини програмного забезпечення «TalkKing» з використанням бази даних. Протестовано основні сценарії використання розробленого мессенджеру.

У розділі зі спеціальної частини з охорони праці було розглянуто особливості гігієни праці для фахівців IT-індустрії. В результаті кваліфікаційної роботи бакалавра отримано застосунок для обміну миттєвими повідомленнями для операційної системи Windows.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Найпопулярніші мессенджери у світі. URL: https://www.statista.com/statistics/258749/most-popular-globalmobilemessenger-apps/ (дата звернення: 24.03.2022)

2. Популярні мессенджери у 2020 р. URL: https://www.epravda.com.ua/rus/news/2020/11/14/653717/ (дата звернення: 02.04.2022)

3. Новини Telegram: вебсайт, URL: https://tlgrm.ru/blog (дата звернення: 05.04.2022)

4. Блог Viber: вебсайт. URL: https://www.viber.com/ru/blog/ (дата звернення: 07.04.2022)

5. Вразливість WhatsApp: вебсайт, URL: https://www.businessinsider.com/whatsapp-hacked-attackers-installed-spyware-2019-5?r=US&IR=T (дата звернення: 08.04.2022)

6. Діаграма варіантів використання: визначення та приклади: вебсайт, URL: https://www.indeed.com/career-advice/career-development/use-case-diagram (дата звернення: 09.04.2022)

7. Що таке макет?: вебсайт, URL: https://www.invisionapp.com/defined/mockup (дата звернення: 11.04.2022)

8. Макет: вебсайт, URL: https://www.productplan.com/glossary/mockup/ (дата звернення: 11.04.2022)

9. Що таке дані?: вебсайт, URL: https://www.javatpoint.com/what-is-database (дата звернення: 12.04.2022)

10. База даних: вебсайт, URL: https://corporatefinanceinstitute.com/resources/knowledge/data-analysis/database/ (дата звернення: 12.04.2022)

11. Діаграма взаємозв'язків сутностей (ERD): вебсайт, URL: https://www.datanamic.com/dezign/erdiagramtool.html (дата звернення: 13.04.2022)

12. Підручник із остаточної діаграми взаємозв'язків сутностей (діаграми ER): вебсайт, URL: https://creately.com/blog/diagrams/er-diagrams-tutorial/ (дата звернення: 14.04.2022)

13. Які мови програмування найкращі для створення програми чату?: вебсайт, URL: https://www.cometchat.com/blog/best-programming-languages-to-build-chat (дата звернення: 17.04.2022)

14. Топ-20 застосунків на React Native в 2022 році. UPD, URL: https://www.purrweb.com/ru/blog/top-prilozhenij-na-react-native/ (дата звернення: 17.04.2022)

15. Розуміння плюсів і мінусів MongoDB: вебсайт, URL: https://www.knowledgenile.com/blogs/pros-and-cons-of-mongodb/ (дата звернення: 19.04.2022)

16. Переваги та недоліки використання MySQL: вебсайт, URL: https://blueclawdb.com/mysql/advantages-disadvantages-mysql/ (дата звернення: 19.04.2022)

17. Переваги SQLite: вебсайт, URL: https://www.javatpoint.com/sqlite-advantages-and-disadvantages (дата звернення: 20.04.2022)

18. Технологія – переваги та недоліки SQLite: вебсайт, URL: https://lifeandwork.blog/2022/05/05/technology-advantages-and-disadvantages-of-sqlite/ (дата звернення: 20.04.2022)

19. Що таке PostgreSQL? Вступ, переваги та недоліки: вебсайт, URL: https://www.guru99.com/introduction-postgresql.html (дата звернення: 20.04.2022)

20. Порівняння систем управління базами даних: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch та інші: вебсайт, URL: https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/ (дата звернення: 20.04.2022)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ОБМІНУ ПОВІДОМЛЕННЯМИ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

СПЕЦІАЛЬНА ЧАСТИНА З ОХОРОНИ ПРАЦІ

ОСОБЛИВОСТІ ГІГІЄНИ ПРАЦІ ДЛЯ ФАХІВЦІВ ІТ-ІНДУСТРІЇ

Спеціальність 121 «Інженерія програмного забезпечення»

121 – КРБ.1 – 409.21810909

Студент _____ О. В. Гранченко
підпис
«__» _____ 2022 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва
підпис
«__» _____ 2022 р.

Миколаїв – 2022

ЗМІСТ

ВСТУП

На сьогоднішній день дуже важливо піклуватися про особисте здоров’я, особливо при роботі в сфері інформаційних технологій (далі – ІТ), адже з нею пов’язана велика кількість проблем з самопочуттям. Кожному ІТ-фахівцю важливо визначити свій особистий графік, піклуватися про здоровий сон, правильно харчуватися та займатися фізичною активність, адже кожен з цих пунктів безпосередньо впливає на якість виконання робочих задач. Обрана тема завжди буде актуальною, оскільки ІТ-сфера розвивається кожного дня величезними кроками і до компаній запрошуються все нові фахівці і їх кількість лише збільшується. Саме тому важливо правильно дбати про кожну людину там створювати індивідуальний підхід, що дозволить зберігати здоров’я та підвищити продуктивність.

Досить багато проблем зі здоров'ям можуть виникнути через використання комп'ютера. Наприклад, проблеми із зором, проблеми зі спиною, стрес, безсоння, тощо. Через це потрібно дотримуватися правил використання комп’ютера та не нехтувати нормами сну.

1 НАЙПОШИРЕНІШІ ПРОФЕСІЙНІ ЗАХВОРЮВАННЯ IT-СПЕЦІАЛІСТА

Здоров’я представників будь-якої сфери залежить від умов роботи та особливостей тієї чи іншої посади. Представники IT-сфери не є виключенням з правила. IT-спеціаліст можливо й не займається фізичною працею, але робота в цій сфері є надзвичайно виснажуючою в психологічному плані. Потрібно ще й взяти до уваги ненормований графік роботи та велике навантаження на орган зору. Слід розглянути найпоширеніші професійні захворювання IT-спеціаліста.

Одним за найголовніших захворювань є **синдром зап’ястного каналу**. Синдром зап’ястного каналу – одне із захворювань, які можуть виникнути у набирачів тексту за комп’ютером. Воно характеризується затерпанням у кисті та пальцях рук. З прогресуванням синдрому виникає біль, який може ускладнювати виконання звичайної щоденної роботи [1]. Його ще називають тунельним синдромом. Хвороба виникає за умов неправильного положення кисті руки та пальців, що створює надмірне навантаження. Для уникнення цього недугу, рекомендується виконувати спеціальні вправи протягом робочого дня. Також непоганим рішенням буде придбати спеціальний килимок зі вставками для зап'ястя, що створити правильні умови для кисті руки.

Іншим та не менш неприємним захворюванням є **синдром сухого ока.** Синдром сухого ока – ще один стан, який може виникати у спеціаліста, що проводить тривалий час за комп’ютером. Коли людина напружено працює за монітором, вона перестає систематично кліпати очима і саме через це рогівка ока не отримує достатнього зволоження, що у свою чергу спричиняє відчуття сухості. Також тривала робота очей на близькій відстані може викликати спазм акомодації – напруження м’яза, який відповідає за зміну кривизни кришталика. Це може призводити до порушення зору на далеку відстань.

Гіподинамія – ще одне захворювання, що іноді носить назву «Синдром сидячої смерті». Воно виникає внаслідок малорухомості людини протягом

робочого дня, що приводить до погіршення функціонування внутрішніх органів. Слід стежити за своїм організмом та звертати увагу на такі симптоми:

— відчуття втоми та хронічна слабкість;

— швидка втомлюваність;

— зниження розумової і фізичної працездатності;

— безсоння вночі й сонливість удень;

— дратівливість, часті зміни настрою, схильність до апатії та депресії;

— зниження, або навпаки, підвищення апетиту;

— зайва вага чи ожиріння;

— порушення в статевій сфері [2].

Якщо вдалось помітити дещо подібне, то найкращим рішенням буде звернутися до лікаря задля персональної консультації. Щоб запобігти цього захворювання, потрібно вести активний спосіб життя та проводити дозвілля з фізичним навантаженням. Навіть ходіння пішки буде чудовими ліками проти гіподинамії.

Безсоння – хронічний недуг, з яким постійно стикаються представники IT-сфери. Відомо, що людині слід спати в середньому по 8 годин на день. Нажаль, важкий графік роботи, постійний стрес та вплив світла і шуму – все це заважає нормальному сну і, наприклад, програміст може спати 4 години на добу, а то і менше. Ознаки безсоння:

— важко заснути;

— прокидаєтеся кілька разів за ніч;

— не можете спати вночі;

— прокидаєтеся вранці і не можете заснути;

— відчуваєте втому після пробудження;

— вам важко заснути вдень, навіть якщо втомилися;

— відчуваєте втому і дратівливість протягом дня;

— важко зосередитися протягом дня [3].

Задля того, щоб уникнути безсоння, людині достатньо налагодити свій режим сну. правильно харчуватися та займатися спортом. І хоча для ІТ-спеціаліста це є досить важким завданням, проте здоров’я потрібно берегти, щоб була змога працювати й далі.

Останнім захворюванням, що дійсно варто виділити є **тромбоз.** Тромбоз глибоких вен – це утворення тромбів, які можуть потрапити в мозок і легені, викликаючи інсульти, емболію легеневої артерії та інші невідкладні стани здоров’я. Термін електронний тромбоз був нещодавно введений, що стосується малорухливого способу життя багатьох користувачів комп’ютерів та ІТ-спеціалістів. Тривалі періоди сидіння можуть спричинити утворення цих небезпечних тромбів.

Профілактика: якщо людина схильна довго сидіти за письмовим столом, принаймні кожну годину потрібно розминатися. Навіть швидка прогулянка до ванної кімнати може розігнати кров і запобігти утворенню тромбів [4].

2 ФІЗИЧНА АКТИВНІСТЬ ТА ФОРМУЛА СНУ ДЛЯ ІТ-СПЕЦІАЛІСТА

Поширеним рішенням проблеми малорухливого способу життя є відвідування фітнес-клубу. У цьому дійсно є свої переваги: під одним дахом можна отримати повний спектр послуг – від тренажерів і басейнів до масажних салонів. Займатися спортом слід хоча б два-три рази на тиждень по 45-90 хвилин. Наслідком такого підходу буде гарний настрій і підвищена продуктивність, що є дуже важливим для ІТ-працівника.

Було доведено неодноразово, що фізична активність покращує самопочуття та здоров’я людини. Протягом тривалого часу це було тісно пов’язане лише з тією частиною здоров’я людини, яка пов’язана з її талією та вмістом жиру в організмі. Однак останні дослідження надають переконливі докази того, що підтримка фізичної активності може покращити когнітивні функції і навіть допомогти розвивати певні частини мозку. Когнітивні функції мозку - це всі функції, які допомагають виконувати завдання. Вони дозволяють отримувати, обробляти та перетворювати інформацію. Оскільки робота з інформацією – це все, роблять розробники програмного забезпечення, я думаю, що можна погодитися, що було б величезною перевагою, якщо б надавалися цим навичкам умови, необхідні для процвітання. Регулярні фізичні вправи допоможуть мозку створювати нові нейрони, ефективно створюючи нові вузли, де можна обробляти інформацію. Фізичні вправи підвищують активність префронтальної кори, частини мозку, яка відповідає за робочу пам’ять. Відомо, що розмір префронтальної кори більший у людей, які є фізично активними, що припускає, що фізичні вправи можуть справді розвивати частину вашого мозку, яка зберігатиме всю інформацію, пов’язану з завданням, упорядкованою та доступною під час роботи [5]. Щоб ІТ-спеціаліст мав достатньо сил на роботу та фізичну активність, йому необхідно спати. Для хорошого самопочуття можна використовувати формулу гарного сну: 10-3-2-1-0.

10 – за 10 годин до сну відмовитися від кави і та інших напоїв, що містять кофеїн.

3 – за 3 години до сну не вживати алкоголь і не навантажувати шлунок важкою їжею.

2 – за 2 години до сну завершити всі робочі справи.

1 – за 1 годину до сну вимкнути телефон, телевізор, комп'ютер і гаджети.

0 – 0 раз вранці перемикати будильник на «ще 5 хвилиночок». Вставати відразу [6].

Кофеїн надзвичайно популярний серед представників IT-сфери, адже він бадьорить. Проте напої з ним можуть порушити якість сну та негативно впливати на людину вранці після пробудження. Весь кофеїн зазвичай нейтралізується людським організмом за 10 годин.

Важка їжа і алкоголь також порушують природній цикл людського сну. Тому важливо не вживати їх за 3 години до сну. Важливо уточнити, що мова йде про невеликі об'єми важкої їжі та алкоголю. Надмірне їх споживання взагалі призведе до перевантаження організму, що призведе до проблем зі сном навіть на декілька днів вперед.

Важливо припинити відповідати на повідомлення, перевіряти електронну пошту і займатися робочими справами за 2 години до сну. Не думати про роботу, оскільки зайві думки у голові не сприяють сну. Якщо відволіктися від думок не вдається, чудовим варіантом буде записати всі думки на листі паперу.

Світло від екранів скорочує вироблення «гормону сну» мелатоніну і порушує циркадний ритм. Щоб легко засинати, потрібно відключати всі гаджети хоча б за годину до сну. Цей час краще провести за читанням, медитацією, спілкуванням. Можна прийняти перед сном теплу ванну, яка допоможе відновитися і легко зануритися в сон. Планшет і смартфон краще взагалі не брати з собою до спальні.

Не менш важливим моментом є вставання з ліжка відразу після пробудження, адже додаткові хвилини в ліжку без сну – роблять людину більш втомленою.

Дотримання усіх цих простих правил дозволять програмісту почуватися краще, позбутися від стресу та покращити свій особистий настрій. Найкращим результатом стане підвищена продуктивність, що дозволить виконувати робочі задачі швидше та ефективніше.

3 ПРАВИЛЬНЕ ХАРЧУВАННЯ ДЛЯ ІТ-СПЕЦІАЛІСТА

Вибір найкращої дієти часто спрямований на фізичний результат, а не на розумовий. Але вибір їжі так сильно впливає на концентрацію та розумову енергію, як і на тренування. Кожна людина відчувала туманність мозку і млявість після важкої їжі, повної неправильних видів їжі. Після цього може бути важко повернутися до продуктивного мислення. Багато робіт вимагають годин глибокої роботи на щоденній основі. Глибока робота – це безперервна, цілеспрямована робота над певним завданням. Приклади цих професій включають інженерів-програмістів, програмістів, науковців даних, аналітиків даних і програмістів. Сфери програмування, кодування та машинного навчання не тільки є конкурентоспроможними, вони ніколи не закінчуються. Все постійно змінюється, і завжди є навички, які можна вдосконалити. Найчастіше саме люди, які можуть постійно навчатися, досягають найбільшого успіху. Правильна дієта може дозволити людині більш ефективно входити і залишатися в цьому глибокому робочому стані. Дієта часто ігнорується у світі технологій. Більшість розробників просто випивають чашку кави і вважають, що їхня дієта підвищить продуктивність протягом дня.

Вибрати найкращу дієту для програмістів не складно. Насправді мова йде більше про вибір правильних поживних речовин, а не про конкретні продукти. Звичайно, поживні речовини та вибір їжі не виключають один одного. Справа в тому, що існує безліч варіантів складання правильної дієти. Мета будь-якого програміста, який виконує тривалу, розумово виснажливу роботу, полягає в тому, щоб мати постійну енергію. Споживання цукру та кофеїну забезпечить швидку енергію, але це не вихід.

Крім того, надто багато енергії одночасно може заважати зосередженню, як і відсутність енергії взагалі. Для досягнення оптимального рівня продуктивності необхідний належний баланс. З огляду на це, цукор/вуглеводи не є ворогом. Вуглеводи є чудовим джерелом енергії для мозку. Однак людям потрібна ця енергія, щоб вона надходила повільно і стабільно. Вуглеводи, як

правило, є швидкозасвоюваною поживною речовиною. У парадигмі вуглеводів є прості і складні вуглеводи. Різниця між ними іноді перебільшена, але факт залишається фактом, що складні вуглеводи перетравлюються трохи повільніше. Складні вуглеводи, такі як овес або картопля, являють собою довгі ланцюги глюкози, які перетравлюються довше, порівняно з простими цукрами, які перетравлюються швидше. Щоб ще більше сповільнити швидкість перетравлення і всмоктування, вуглеводи слід поєднувати з жирами. Клітковина також є варіантом, оскільки вона також уповільнює перетравлення інших поживних речовин. За швидкістю перетравлення, білок знаходиться між вуглеводами і жирами. Хоча білок не є основним джерелом енергії (він використовується в основному для відновлення м’язів та інших тканин), його не слід ігнорувати в раціоні. Хороша їжа, яка підтримає програміста протягом 3-4 годин, міститиме близько 50 грамів вуглеводів, 20 грамів жирів і 20 грамів білка. Деякі люди люблять йти без вуглеводів і споживають лише жири, білки та клітковину [7].

Визначивши частину макроелементів, можна розглянути окремі мікроелементи, які покращують когнітивні здібності. Одним з таких поживних речовин є холін. Холін, який перетворюється в ацетилхолін в організмі, може допомогти у виконанні завдань, пов’язаних із навчанням. Джерелами холіну є цілі яйця, яловичина та горіхи. Чорниця містить багато антиоксидантів, які можуть допомогти покращити пізнання. Ці антиоксиданти називаються антоціанами, які надають чорниці та іншим фруктам колір. Це один з найпотужніших продуктів харчування для загального здоров’я, гарного самопочуття та довголіття. Це важливо, особливо для сидячих робіт, таких як програмування та комп’ютерна інженерія, які можуть негативно вплинути на здоров’я в довгостроковій перспективі. Деякі поживні речовини найкраще споживати як дієтичні добавки, оскільки буває важко отримати достатню кількість лише з їжі. Родіола – це трава, яка покращує пізнання унікальним чином, зменшуючи втому. Програмісти та комп’ютерники дуже добре знайомі з «вигоранням». Було показано, що родіола зменшує цей ефект. Ашваганда не

має прямого впливу на розумову концентрацію, це, мабуть, найкраща добавка для зниження стресу та тривоги. В результаті цього ефекту було показано, що він опосередковано покращує мотивацію. Омега-3 жирні кислоти також можуть знижувати гормони стресу, такі як кортизол. Природно міститься в рибі, як лосось, люди, які зазвичай не їдять рибу, можуть доповнювати якісним риб’ячим жиром.

Настав час вибрати правильні продукти, щоб покращити розумову концентрацію для довгострокових комп’ютерних програм та інженерних проектів. Потрібно досягти поєднання корисних вуглеводів і жирів для тривалої енергії. Приклади їжі включають: миску вівсяної каші з арахісовим маслом, курку з рисом і брокколі, зварену в оливковій олії, або стейк з картоплею. Деякі ситуації вимагають чогось більш зручного, коли немає часу приготувати їжу або це може бути пізня ніч. Легким, здоровим, комбінованим перекусом вуглеводів і жирів будуть горіхи з сухофруктами, темний шоколад (какао в чорному шоколаді також має когнітивну користь) або протеїновий батончик. Споживання більшої частини продуктів у вигляді цільнозернових, фруктів, овочів, корисних олій, горіхів і нежирного м’яса є хорошою стратегією для успіху [7].

4 ВИМОГИ ДО КОРИСТУВАННЯ КОМП’ЮТЕРОМ

Існує багато проблем зі здоров’ям та безпекою, які можуть бути пов’язані з використанням комп’ютерів. Тому вжито заходів безпеки, щоб уникнути травм. Ці заходи включені в законодавство, яким роботодавці та працівники повинні дотримуватися, щоб забезпечити відповідальне використання комп’ютерів. Законодавство охоплює цілий ряд факторів, пов’язаних із використанням комп’ютера, які є основою для оцінки ризику робочої станції. Оцінка ризику робочої станції використовується, щоб переконатися, що кожен окремий працівник має відповідне обладнання, яке дозволить йому комфортно працювати перед комп’ютером. Це закріплено в законодавстві про охорону праці, щоб допомогти працівникам запобігти травматизму. Монітор комп'ютера - це екран, на який ви дивитеся, коли користуєтесь комп'ютером або ноутбуком. Багато проблем з зором можуть бути наслідком відсутності відповідного екрана комп’ютера для запланованого використання. Наприклад, якщо ви збираєтеся сидіти за комп’ютером протягом тривалого періоду часу, ви можете втомити очі, якщо ваш екран недостатньо великий або недостатньо яскравий. Ви також можете спричинити травму постави, якщо верхня частина екрана вашого комп’ютера знаходиться не на рівні ваших очей. Сидіти в будь-якому місці протягом тривалого періоду часу може бути незручно. Однак, якщо не сидіти на кріслі з належною підтримкою постави для вашого розміру, це може спричинити серйозні травми або проблеми з поставою. Якщо розглядати комп’ютерні стільці як частину охорони здоров’я та безпеки, можна включити інші елементи, такі як підставка для ніг, щоб вам було зручніше сидіти в кріслі. Розміщення предметів, які ви часто використовуєте, поруч із робочою станцією, може допомогти вам уникнути напруги. Однак наявність занадто великої кількості предметів навколо робочої станції також може призвести до неправильного розташування. Перерва є частиною охорони здоров’я та безпеки комп’ютера [8].

ВИСНОВКИ

Під час написання спеціальної частини з охорони праці було описано найпоширеніші захворювання ІТ-спеціалістів, способи їх запобігання та загальні поради щодо них. Приведено норми фізичної активності, що дозволяють фахівцю позитивно впливати на власне здоров’я, настрій та продуктивність праці. Це дозволить також покращити когнітивні функції, які є важливою частиною життя спеціалісту в цій сфері. Складено формулу сну, дотримуючись якої робітник зможе починати кожен день з гарним настроєм та високим рівнем бадьорості. Формула сну 10-3-2-1-0 дозволить працівникові налагодити свій режим життя, що позитивно позначиться на всіх аспектах роботи ІТ-спеціаліста. Проаналізовано харчування та, необхідні для розумових активностей, речовини, наведено приклад раціону. Виявлено ті речовини, які краще споживати у вигляді добавок через недостатню їхню кількість у звичайних продуктах харчування. Визначено вимоги до правильного користування комп’ютером, щоб працівник мав змогу з максимальним комфортом виконувати поставлені завдання. Підкреслено особливу важливість наявності коротких перерв під час роботи будь-якого ІТ-працівника, які дозволяють знизити рівень напруження органів зору та інших важливих частин людського тіла.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Здоров’я IT-спеціаліста: сон, харчування, фізична активність. URL: https://dou.ua/lenta/articles/how-to-be-healthy-general/
2. Як програмувати й залишатися здоровими? URL: https://issoft.ua/blog/yak-programuvati-y-zalishatisya-zdorovi/
3. Причини безсоння та його наслідки: як подолати проблеми зі сном. URL: https://life.pravda.com.ua/health/2021/02/16/243955/
4. 10 Major Health Concerns For IT Professionals. URL: https://www.businessinsider.com/if-you-work-in-it-these-10-things-are-probably-killing-you-2011-11
5. How Physical Activity Makes You a Better Software Developer. URL: https://medium.com/swlh/how-physical-activity-makes-you-a-better-software-developer-5ff409cae5ea
6. Формула гарного сну: 10-3-2-1-0. URL: https://hiitworks.com/blog/yak_vispatisya_formula_garnogo_snu/
7. Best Diet for Software Engineers and Computer Programmers. URL: https://www.mindsetsandreps.com/best-diet-for-software-engineers-and-computer-programmers/
8. Computer Health and Safety. URL: https://cpdonline.co.uk/knowledge-base/health-and-safety/computer-health-safety/