ДзоДФорма № Н-9.02

Міністерство освіти і науки України

Відокремлений структурний підрозділ «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя»

(повне найменування вищого навчального закладу)

Відділення інформаційних технологій, менеджменту, туризму та підготовки іноземних громадян

(назва відділення)

Циклова комісія комп'ютерної інженерії

(повна назва циклової комісії)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

бакалавра

(освітньо-професійного ступеня)

на тему: **Розробка програмного модуля автоматичного перекладу повідомлень в інтернет- чаті**

Виконав: студент VII курсу, групи КІб-703

Спеціальності **123 Комп'ютерна інженерія**

(шифр і назва спеціальності)

_____ Максим БРИНИК

(ім'я та прізвище)

Керівник _____ Андрій ЛЯПАНДРА

(ім'я та прізвище)

Рецензент _____ _____

(ім'я та прізвище)

Тернопіль – 2026

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ТЕРНОПІЛЬСЬКИЙ ФАХОВИЙ КОЛЕДЖ
ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
імені ІВАНА ПУЛЮЯ»

Відділення **інформаційних технологій, менеджменту, туризму та підготовки іноземних громадян**
Циклова комісія **комп'ютерної інженерії**
Освітньо-професійний ступінь **фаховий молодший бакалавр**
Освітньо-професійна програма: **Обслуговування комп'ютерних систем і мереж**
Спеціальність: **123 Комп'ютерна інженерія**
Галузь знань: **12 Інформаційні технології**

ЗАТВЕРДЖУЮ
Голова циклової комісії
комп'ютерної інженерії
_____ Андрій ЮЗЬКІВ
"30" квітня 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Бринику Максиму Васильовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: **Розробка програмного модуля автоматичного перекладу повідомлень в інтернет- чаті**
керівник роботи Ляпандра Андрій Степанович
(прізвище, ім'я, по батькові)

Затверджені наказом Відокремленого структурного підрозділу «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя» від 18.05.2026р № 4/9-252.

2. Строк подання студентом роботи: 22 червня 2026 року.

3. Вихідні дані до роботи: плани приміщень, завдання на проєктування, стандарти побудови СКС, документація на мережеве обладнення і сервери

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): Загальний розділ. Розробка технічного та робочого проєкту. Спеціальний розділ. Економічний розділ. Безпека Життєдіяльності, основи охорони праці.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
   - Таблиця техніко-економічних показників;
   - Структурна схема програмної схеми;
   - Алгоритм роботи програмного модуля автоматичного перекладу повідомлень;
   - ER-діаграма бази даних;
   - UML-діаграма варіантів використання програмного модуля автоматичного перекладу повідомлень в інтернет-чаті
   - UML-діаграма класів програмної системи програмного модуля автоматичного перекладу в інтернет-чаті

6. Консультанти розділів роботи

Розділ	Ім'я, прізвище та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний розділ	Ірина ЛЕЩИК викладач		
Охорона праці, техніка безпеки та екологічні вимоги	Володимир ШТОКАЛО викладач		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Отримання і аналіз технічного завдання	18.05	
2	Збір і узагальнення інформації	21.05	
3	Написання першого розділу	52.05	
4	Розробка технічного та робочого проекту	1.06	
5	Написання спеціального розділу	8.06	
6	Розрахунок економічної частини	11.06	
7	Написання розділу охорони праці	12.06	
8	Виконання графічної частини	14.06	
9	Оформлення проекту	18.06	
10	Погодження нормоконтролю	19.06	
11	Попередній захист роботи	22.06	
12	Захист кваліфікаційної роботи		

7. Дата видачі завдання: 18 травня 2026 року

Студент _____ Максим БРИНИК
(підпис) (ім'я та прізвище)

Керівник роботи _____ Андрій ЛЯПАНДРА
(підпис) (ім'я та прізвище)

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ І СКОРОЧЕНЬ

API (Application Programming Interface) — програмний інтерфейс застосунку, набір засобів для взаємодії між програмними компонентами.

ADO.NET (ActiveX Data Objects .NET) — технологія платформи .NET для доступу до даних і взаємодії з базами даних.

БД — база даних, організована сукупність взаємопов'язаних даних для зберігання та обробки інформації.

Веб-сокети (Web Sockets) — технологія двостороннього обміну даними між клієнтом і сервером у режимі реального часу.

HTTP (HyperText Transfer Protocol) — протокол передачі гіпертексту, що використовується для обміну даними між клієнтом і сервером.

JSON (JavaScript Object Notation) — текстовий формат обміну даними між клієнтською та серверною частинами системи.

MVVM (Model–View–ViewModel) — архітектурний шаблон проєктування програмного забезпечення, який відокремлює бізнес-логіку від інтерфейсу користувача.

ORM (Object-Relational Mapping) — технологія об'єктно-реляційного відображення, що забезпечує взаємодію між об'єктами програми та реляційною базою даних.

REST (Representational State Transfer) — архітектурний стиль побудови веб-сервісів для взаємодії між клієнтом і сервером через HTTP-запити.

SQL (Structured Query Language) — структурована мова запитів для створення, зміни та керування даними в реляційних базах даних.

СКБД — система керування базами даних, програмне забезпечення для створення та адміністрування баз даних.

СУБД — система управління базами даних. У проєкті використовується MySQL.

DFD (Data Flow Diagram) — діаграма потоків даних, що використовується для моделювання процесів обробки інформації в системі.

IDEF0 (Integration Definition for Function Modeling) — методологія функціонального моделювання бізнес-процесів та інформаційних систем.

ВСТУП

У сучасному світі Інтернет відіграє ключову роль у спілкуванні, а значна частина взаємодії здійснюється через месенджери та онлайн-чати. Проте мовний бар'єр все ще створює труднощі для повного розуміння між користувачами, а використання сторонніх перекладачів часто виявляється незручним для швидкого обміну інформацією. Для розв'язання цієї проблеми був створений програмний модуль, який об'єднує функції месенджера з автоматичним перекладом. Цей інструмент дозволяє обмінюватися повідомленнями з перекладом у реальному часі, що відкриває можливість спілкуватися незалежно від знання мов. Запропоноване рішення значно спрощує комунікацію, усуває мовні бар'єри та стає незамінним інструментом в онлайн-бізнесі, сприяючи залученню нових клієнтів і розширенню міжнародного співробітництва. Крім того, впровадження такого модуля значно покращує зручність використання сучасних засобів зв’язку, зменшує ризик непорозумінь і забезпечує доступніше та ефективніше міжнародне спілкування для широкого кола користувачів. У перспективі подальший розвиток цього рішення може передбачати впровадження додаткових функцій, таких як підтримка голосових повідомлень, інтеграція з різними платформами та покращення якості перекладу за допомогою сучасних технологій штучного інтелекту. Це дозволить ще більше підвищити ефективність і універсальність системи. Таким чином, подібний підхід має великий потенціал для подальшого вдосконалення та широкого застосування в різних сферах діяльності.

1 АНАЛІЗ ІНТЕРНЕТ-ЧАТІВ

1.1 Коротка характеристика інтернет-чатів

Доступність комп’ютерів і смартфонів стала стимулом для значних змін у розвитку новітніх комунікаційних систем. Замість класичних телефонних дзвінків все більше використовуються голосові повідомлення та відеодзвінки. Суспільство активно перейшло до використання комп’ютерів, ноутбуків та Інтернету для спілкування. Первинним інструментом в таких взаємодіях стала електронна пошта, яка нагадувала традиційне листування, але була значно швидшою та зручнішою. Згодом популярність здобули чати, котрі дозволяли одночасно спілкуватися декільком користувачам у групових розмовах. Перший багатокористувацький чат у реальному часі, Talkomatic, був створений у 1973 році Дугом Брауном і Девідом Р. Валлі в рамках системи програмних логік для автоматизованих навчальних операцій (PLATO), розробленої в Університеті штату Іллінойс. Talkomatic пропонував шість каналів, кожен з яких міг одночасно обслуговувати до п’яти користувачів. Відтоді онлайн-чати зазнали стрімкого розвитку, і сучасні сервіси майже не обмежують кількість користувачів чи каналів зв’язку. Тепер чати оснащуються передовими технологіями, такими як нейронні мережі, штучний інтелект та машинне навчання, що значно покращують якість взаємодії між користувачами і технологіями. Паралельно з розвитком чатів з'явилися інші засоби комунікації — месенджери, які швидко здобули популярність. Найперші системи передачі даних, хоча й виглядали примітивно з огляду на сучасні стандарти, були створені ще в 1970-х роках. А вже в 1990-х роках зросла популярність графічних інтерфейсів для користувачів, що забезпечували зручне відображення отриманих повідомлень. До найперших месенджерів належать PowWow, ICQ та AOL Instant Messenger — вони стали основою для подальшого розвитку цих технологій.На рисунку 1.1 наведено інтерфейс ICQ 1996–1997 років.

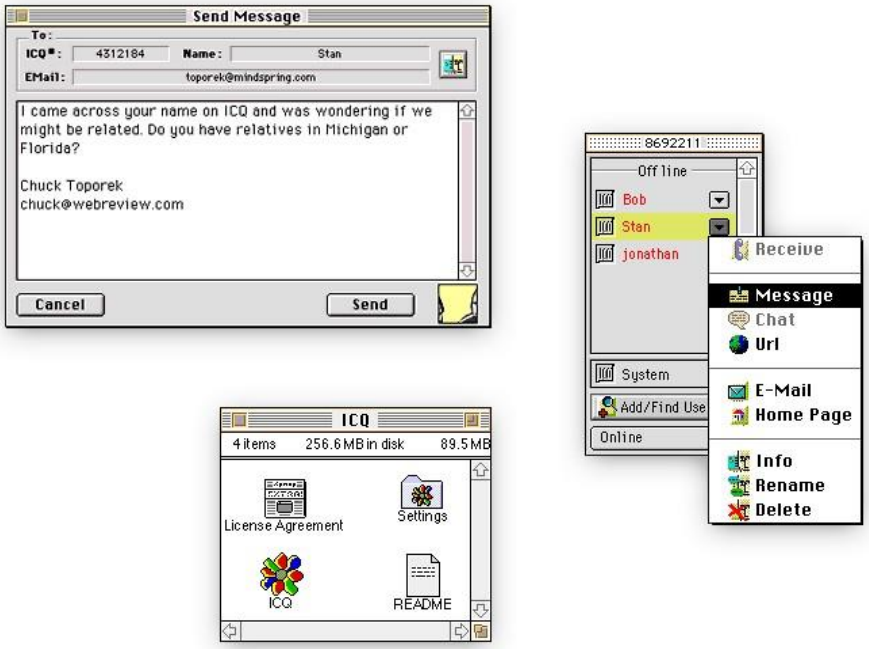


Рисунок 1.1 – Інтерфейс ICQ 1996–1997 роки

У той же час інші компанії вже розробили власне програмне забезпечення для обміну повідомленнями в реальному часі, зокрема Yahoo!, MSN, Ubique та Excite. Кожен із таких застосунків використовував унікальний протокол передачі даних і вимагав окремого клієнта. Тому користувачам, які хотіли взаємодіяти через кілька месенджерів одночасно, доводилося запускати декілька програм. У 2000 році було випущено у вільний доступ програмне забезпечення Jabber [1]. Цей протокол згодом стандартизували під назвою Extensible Messaging and Presence Protocol (XMPP). Сервери на основі XMPP могли працювати як шлюзи для інших протоколів обміну миттєвими повідомленнями, завдяки чому потреба у використанні кількох клієнтських додатків одночасно значно зменшилася. На рисунку 1.2 наведено інтерфейс AOL Instant Messenger у 2000 роках.

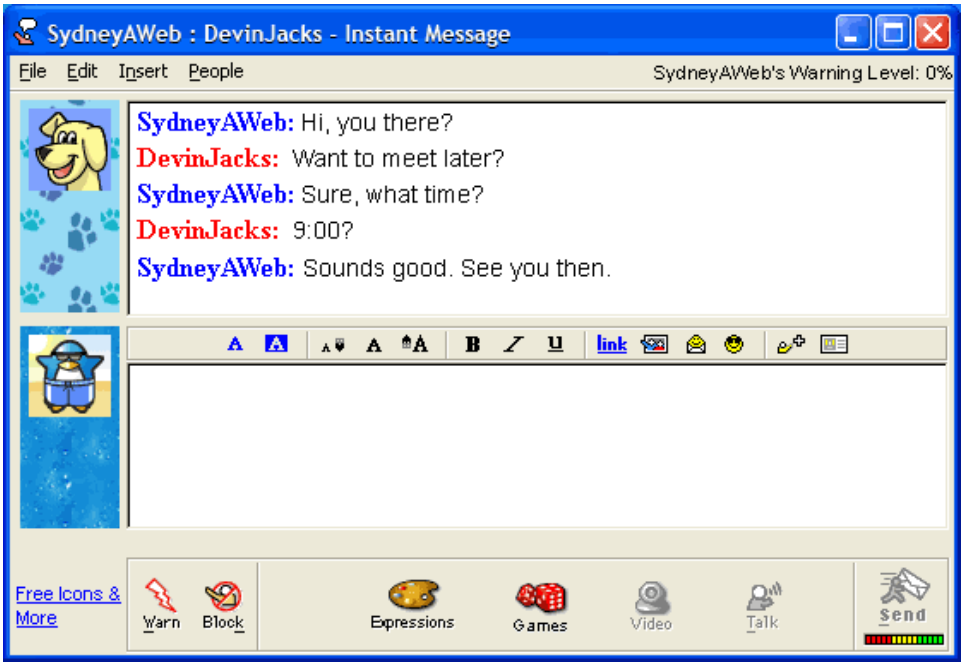


Рисунок 1.2 – Інтерфейс AOL Instant Messenger у 2000 роках

Перші користувацькі інтерфейси програм були доволі примітивними, надаючи обмежений функціонал за сучасними стандартами. Зокрема, ранні CMOS мали суттєві обмеження щодо максимальної довжини повідомлень, які могли надсилати користувачі. Разом із аналогічними обмеженнями для SMS це сприяло виникненню специфічних слів, скорочень та інтернет-сленгу. Завдяки їм користувачі могли швидше друкувати текст або зменшити кількість символів у повідомленні. Наприклад, популярні такі абревіатури: "lol" (сміятися вголос), "brb" (скоро повернуся) і "TTYL" (поговоримо пізніше). Такі скорочення стали невід'ємною частиною неформальної комунікації, і деякі люди навіть адаптували їх до своєї повсякденної мови. Вони значною мірою сформували культуру спілкування в мережі, без використання яких розуміння сучасної інтернет-комунікації стало б непростою задачею для багатьох нових користувачів. AIM можна сміливо назвати одним із прототипів сучасних соціальних мереж та месенджерів. У ньому профілі учасників дозволяли коротко описувати себе і свою біографію, а користувачі могли переглядати чужі профілі та знаходити інформацію про інших. Ця можливість тоді була

новаторською функцією. Першою соціальною мережею, близькою за форматом до сучасних платформ, став сайт Friendster, який за перші три місяці привернув увагу 3 мільйонів користувачів. Це означало, що кожна 126-та людина в Інтернеті була частиною Friendster. Успіху Friendster послідував MySpace, запущений у серпні 2003 року. На піку популярності кількість його зареєстрованих користувачів досягала 90 мільйонів. Facebook вперше побачив світ у 2004 році як платформа для об'єднання студентів американських університетів. Заснований у Гарварді Марком Цукербергом, Facebook спочатку працював лише для студентів цього університету, і доступ до нього можна було отримати виключно через запрошення. Такий підхід створив ефект елітарності, що сильно сприяло популярності мережі. У перший місяць понад половина студентів Гарварду вже використовували Facebook. Лише через два роки платформа стала відкритою для широкого загалу. У 2008 році саме Facebook перевершив MySpace та Friendster за кількістю користувачів. До 2017 року ця соціальна мережа зібрала понад 2 мільярди активних користувачів по всьому світу, серед яких було близько 10 мільйонів українців. Сучасні СМС (соціальні медіа та сервіси) продовжують активно розвиватися. Нещодавно відновилася популярність сервісів миттєвого обміну повідомленнями, таких як месенджери та чати. Ймовірно, це стало можливим завдяки технологічним досягненням: швидкісний мобільний Інтернет, доступність смартфонів та зниження вартості послуг зв'язку зробили ці інструменти максимально комфортними для великої кількості людей. На графіку (рисунок 1.3) можна побачити кількість активних користувачів у найпопулярніших месенджерах станом на квітень 2019 року за даними дослідження зі сайту statista.com. На рисунку 1.3 наведено кількість активних користувачів найпопулярніших месенджерів станом на квітень 2019 року.

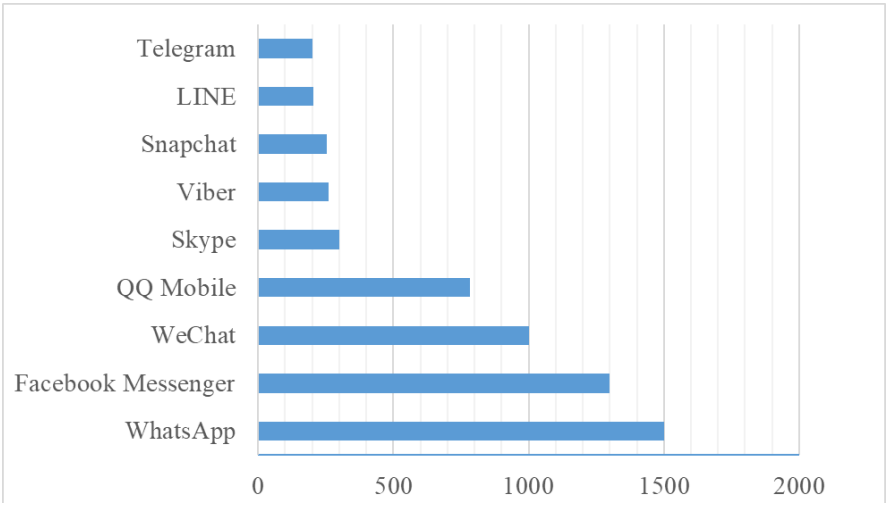


Рисунок 1.3 – Кількість активних користувачів у найпопулярніших месенджерах

Очевидно, що перше місце серед популярних месенджерів посідає WhatsApp із загальною кількістю користувачів близько 1,5 мільярда. На другому місці розташовується Facebook Messenger з 1,3 мільярдами активних користувачів щомісяця [1]. Незважаючи на очевидні відмінності між месенджерами, вказаними у цьому списку, вони мають спільні характеристики, які стали основою їхньої популярності. Практично всі ці платформи вирізняються зручним і зрозумілим для користувача інтерфейсом, що полегшує їх використання. Крім того, кожен із месенджерів підтримує спілкування в групових чатах. Деякі з них пропонують можливість створення каналів – інструменту, який схожий на функціонал месенджера, але з обмеженнями на публікації: право додавати контент мають лише певні користувачі з відповідним доступом. Однак для глибшого аналізу переваг і недоліків цих платформ варто звернути увагу й на їхні відмінності. Майже всі популярні месенджери забезпечують можливість аудіодзвінків, а більшість із них – і відеозв'язок. Сервіси, такі як Telegram, Viber, Skype, WeChat і Facebook Messenger, також дозволяють передавати файли між користувачами. Щоб краще зрозуміти особливості, можемо порівняти WhatsApp – найбільший за кількістю активних користувачів месенджер – із Telegram, який демонструє найшвидші темпи зростання. Динаміку зростання числа користувачів

WhatsApp за період із січня 2015 року по грудень 2017 року наведено на рисунку 1.4. За цей проміжок часу кількість активних користувачів платформи фактично подвоїлася: з 700 мільйонів до 1,5 мільярда осіб. Тобто кожна п'ята людина на планеті щомісяця використовує WhatsApp. На рисунку 1.4 наведено кількість активних користувачів WhatsApp за період з 2015 по 2017 роки.

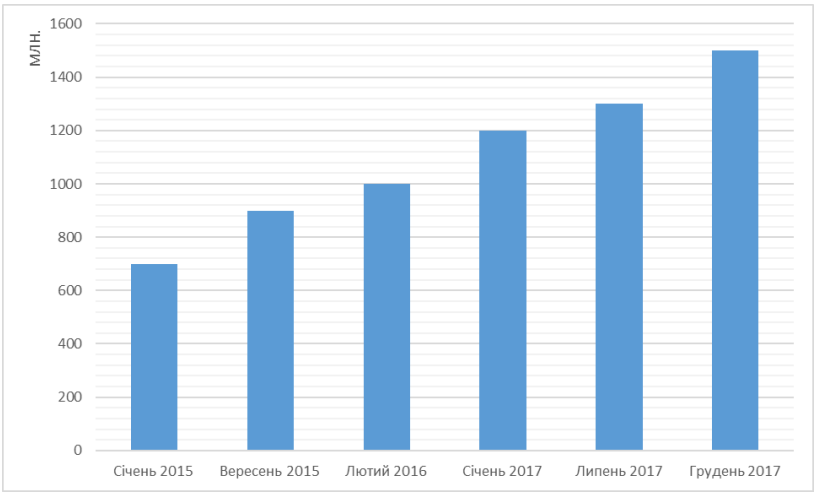


Рисунок 1.4 – Кількість активних користувачів WhatsApp за період з 2015 по 2017 роки.

Telegram — це месенджер, який по праву можна вважати інструментом комунікації майбутнього. Його популярність стрімко зростає у глобальному масштабі. Згідно з даними, за період із лютого 2016 року по грудень 2017 року кількість активних користувачів Telegram зросла на 80%, що проілюстровано на рисунку 1.5. Для порівняння, за аналогічний період зростання кількості користувачів у сервісі WhatsApp становило 50%. Таким чином, Telegram демонструє статус найбільш динамічного месенджера серед подібних платформ. Станом на квітень 2018 року месенджер подолав позначку у 200 мільйонів активних користувачів, підтверджуючи свій стрімкий прогрес. На рисунку 1.5 наведено кількість активних користувачів Telegram за період з 2014 по 2017 роки.

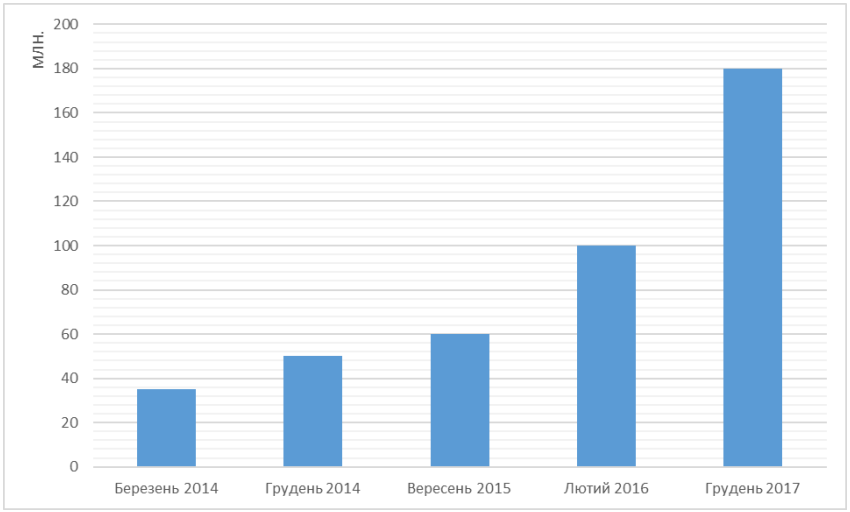


Рисунок 1.5 – Кількість активних користувачів Telegram в період з 2014 по 2017 роки.

Варто зауважити, що Telegram пропонує функцію таємних чатів, у яких доступ до змісту практично виключений. Крім того, у порівнянні з WhatsApp, Telegram працює швидше та забезпечує можливість відправляти файли практично будь-якого формату. Недоліком WhatsApp є те, що він може бути зареєстрований лише на одному пристрої, а відповідно до політики використання, часта зміна прив'язки до різних пристроїв загрожує блокуванням акаунта. Останніми роками прості телефонні дзвінки значною мірою втратили популярність, поступаючись місцем сучасним месенджерам та іншим платформам для обміну інформацією й повідомленнями, а також онлайн-сервісам зв'язку. Зараз більшість людей та компаній віддають перевагу текстовим або мультимедійним повідомленням замість звичних дзвінків. Аналіз сучасних соціальних медіа та онлайн-сервісів (SMOS) свідчить про їхню ключову роль у сучасній комунікації через інтернет. Сучасний користувач прагне спілкування, яке буде простим, економним і швидким, тому вибір месенджерів є досить широким. Однак Telegram виділяється серед інших значно швидшим зростанням кількості активних користувачів, що свідчить про його значний потенціал і перспективи подальшого розвитку. Таким чином, питання

підтримки та вдосконалення SMOS залишається актуальним і заслуговує на наукову увагу у поточних умовах.

1.2 Особливості проектування інтернет чату

Для розробки програмного забезпечення цього типу доцільно застосовувати такі технології, як Microsoft .NET, Xamarin та Entity Framework. Вони є оптимальним вибором для вирішення подібних завдань, оскільки гарантують стабільність роботи модуля та підвищують його надійність під час експлуатації. Завдяки цьому підходу можна створити високоякісний і конкурентоспроможний продукт. У наступних частинах цього розділу ці технології будуть розглянуті детальніше [3].

Технологія Microsoft .NET являє собою програмну платформу, розроблену компанією Microsoft для створення як десктопних, так і веб-застосунків. Вона багато в чому базується на ідеях та концепціях, закладених в технологію Java. Одним із ключових принципів .NET є забезпечення сумісності між службами, написаними різними мовами програмування. Хоча Microsoft позиціонує це як одну з головних переваг платформи, варто зазначити, що така функціональність вже тривалий час доступна і в платформі Java. Кожна бібліотека в .NET містить інформацію про свою версію, що дає змогу ефективно уникати конфліктів, пов’язаних із використанням різних версій тих самих збірок. .NET — це кросплатформна технологія, яка на сьогодні має реалізації для різних операційних систем. Зокрема, вона підтримується на Microsoft Windows, FreeBSD (у версії від Microsoft), а також на Linux завдяки проекту Mono, створеному в рамках угоди між Microsoft та Novell. Для захисту авторських прав технологія реалізує середовище виконання — Common Language Runtime (CLR), яке є основою для виконання програм, створених на платформі .NET. При цьому компілятори для роботи з .NET

пропонуються багатьма компаніями для різних мов програмування безкоштовно. Структура .NET поділяється на два основні компоненти: середовище виконання (яке фактично виконує функції віртуальної машини) та набір інструментів для розробки, що забезпечують зручність створення й підтримки програмного забезпечення. Середовища для розробки .NET-додатків включають такі інструменти, як Visual Studio .NET (C++, C#, J#), SharpDevelop, Borland Developer Studio (Delphi, C#) та інші. Навіть популярна платформа Eclipse пропонує доповнення для створення програм на .NET. Варто зазначити, що розробку застосунків можна здійснювати навіть у звичайному текстовому редакторі, використовуючи консольний компілятор. Для реалізації кросплатформних додатків суттєвою перевагою є використання платформи Xamarin.Forms. Цей інструмент стає особливо актуальним з огляду на те, що, за загальною статистикою, більшість мобільних додатків розробляються відразу для кількох платформ, як-от Android та iOS. Однак розробники часто стикаються з певними труднощами під час роботи: 1. Відмінності у підходах до створення графічних інтерфейсів. Це змушує адаптувати дизайн додатка під специфічні вимоги кожної платформи. 2. Несумісність API. Різниця в програмних інтерфейсах та реалізації певного функціоналу вимагає врахування особливостей кожної системи. 3. Особливості платформ для розробки: - Для створення додатків під iOS необхідне середовище Mac OS X і спеціальні інструменти на кшталт Xcode. Як мова програмування зазвичай обираються Objective-C чи Swift. - Для Android є широкий вибір середовищ розробки, приміром, Android Studio або Eclipse. Основна мова розробки для цієї платформи — Java. - Додатки для Windows створюються у Visual Studio з використанням мов програмування C#, VB.NET чи C++. Такий широкий спектр платформ, інструментів та мов програмування може ускладнювати та подовжувати час розробки програмного забезпечення. Очевидно, що наявність універсального інструменту, здатного спростити і прискорити процес створення

кросплатформних додатків, значно покращила б ситуацію. Саме таким рішенням виступає платформа Xamarin, яка об'єднує можливості розробки під різні операційні системи в єдиній екосистемі[7]. Xamarin дозволяє створювати єдиний програмний код для всіх трьох основних платформ — Android, iOS та Windows — використовуючи C# та технології .NET. Одна з ключових переваг Xamarin.Forms полягає в тому, що під час розробки значна частина коду є спільною для різних операційних систем. Крім того, Xamarin забезпечує прямий доступ до нативних API кожної платформи, що дає змогу створювати високоякісні додатки з повною інтеграцією. У процесі створення додатків застосовується платформа .NET і мова програмування C# (а також за потреби F#). Обидві мови відзначаються продуктивністю, а також зрозумілістю та легкістю у навчанні й використанні. З функціональної точки зору, Xamarin складається з декількох підплатформ, серед яких:

– Xamarin.Android — це інструменти та бібліотеки, призначені для розробки додатків під операційну систему Android.;

– Xamarin.Mac – це набір бібліотек, які забезпечують інструменти та функціональність для розробки додатків під операційну систему macOS;

– Xamarin.iOS — це набір бібліотек, які дозволяють розробляти мобільні додатки для iOS-платформи.

Субплатформи відіграють значну роль, оскільки саме через них додатки надсилають запити до прикладних інтерфейсів на пристроях із операційними системами Android або iOS [15].

Завдяки цим платформам ми маємо змогу розробляти окремі додатки для Android та iOS, але справжньою перевагою є їхня здатність підтримувати розробку кросплатформених застосунків. Це означає, що одна й та сама логіка може функціонувати на різних платформах. Ми можемо один раз створити візуальний інтерфейс і підключити до нього необхідну логіку на C#, після чого цей підхід без проблем працюватиме на

Android, iOS і Windows. Така можливість реалізована завдяки технології Xamarin.Forms. [14].

1.3 Аналіз і огляд існуючих аналогів

Перед початком проєктування та розробки системи доцільно здійснити глибокий аналіз ринкових рішень. Вивчення конкурентів, їх сильних і слабких сторін дозволяє сформулювати чіткі технічні й функціональні вимоги до майбутнього продукту, уникнути типової помилковості у процесі реалізації та значно покращити загальну якість розробки. Цей підхід також сприяє визначенню унікальних характеристик, які можуть вигідно виділити нову систему серед існуючих аналогів. У контексті досліджуваної галузі, для вирішення комунікаційних задач активно застосовуються різноманітні інтернет-месенджери, які забезпечують швидкий обмін текстовими повідомленнями та мультимедійним контентом між користувачами. Одним із найпопулярніших представників цієї категорії є Telegram — інноваційна платформа, доступна на смартфонах, планшетах і персональних комп’ютерах. Сервіс дозволяє надсилати текстові повідомлення, передавати файли різних форматів (включно із зображеннями та відео), а також здійснювати безкоштовні голосові виклики. Реєстрація у Telegram базується на використанні мобільного номера телефону: для авторизації система надсилає одноразовий SMS-код підтвердження. Такий механізм усуває необхідність запам’ятовування складних паролів і забезпечує підвищений рівень зручності для користувачів. Особливо варто відзначити широкий спектр додаткових функцій Telegram, таких як створення і адміністрування групових чатів, каналів для поширення інформації, інтеграція ботів, а також синхронізація даних між різними пристроями. Саме ці можливості формують його лідерські позиції серед сучасних інструментів онлайн-комунікації та можуть служити важливим орієнтиром

при розробці нових програмних продуктів. На рисунку 1.6 демонструється інтерфейс головного меню Telegram. На рисунку 1.6 наведено головну сторінку ресурсу Telegram.

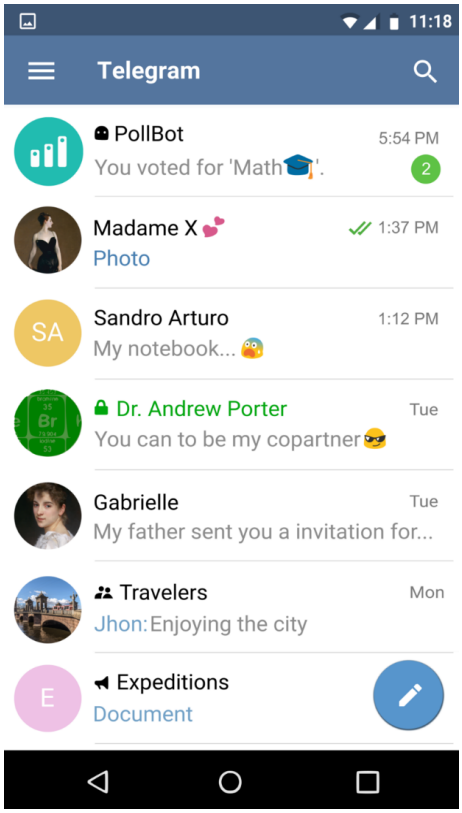


Рисунок 1.6 – Головна сторінка ресурсу Telegram

Переваги:

– секретні чати;
– вбудований VPN;
– можливість пошуку співрозмовників;
– зручний інтерфейс.

Недоліки:

– незручний пошук співрозмовників;
– відсутність можливості перекладу повідомлень при спілкуванні із іноземцями;

Mate Translate — це система перекладу, інтегрована в браузери Chrome, яка призначена для перекладу текстів. Завдяки їй можна швидко й зручно

перекладати повідомлення та іншу інформацію прямо в браузері. Головний інтерфейс користувача представлено на рисунку 1.7 На рисунку 1.7 наведено інтерфейс системи перекладу Mate Translate.

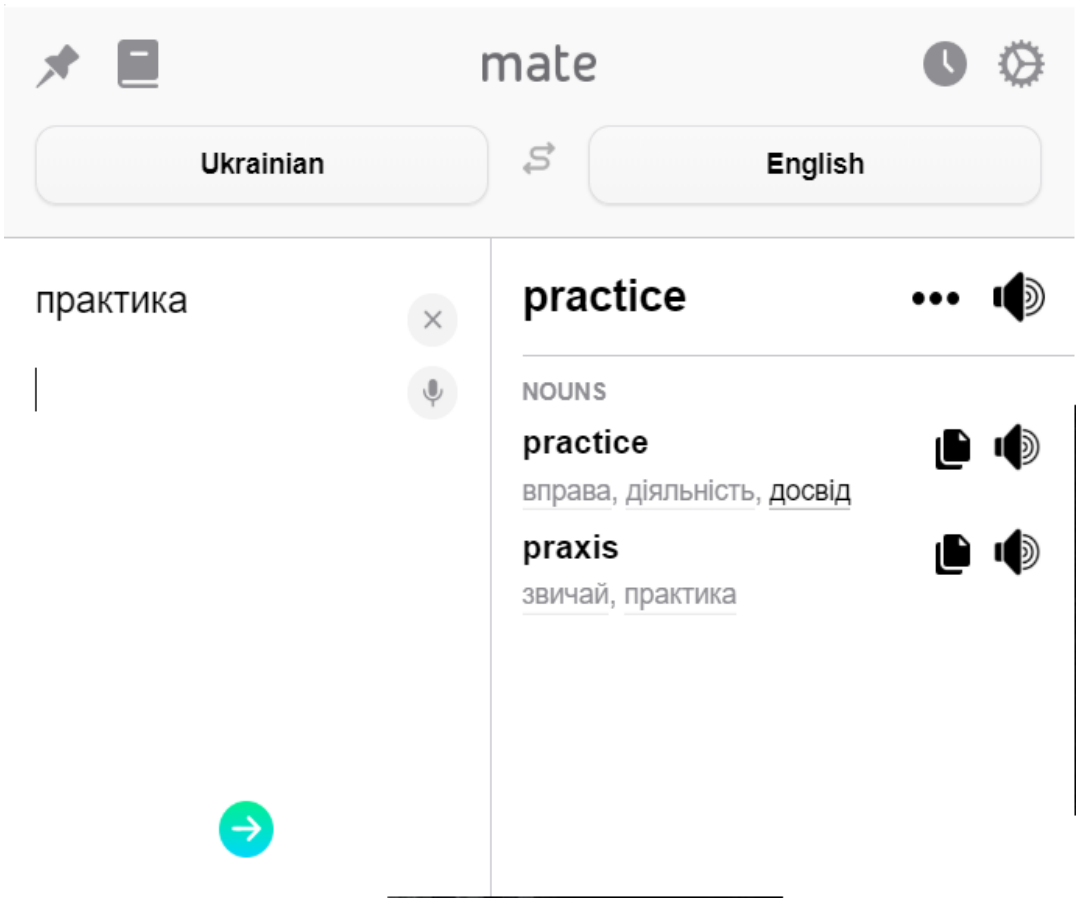


Рисунок 1.7 – Інтерфейс «Mate translate».

Переваги:

– зручний переклад інформації;
– велика кількість мов;
– можливість створення власного словника.

Недоліки:

– це розширення для Google Chrome, яке не працює за його межами
– відсутність можливості надсилати повідомлення;
– присутня платна преміум підписка;

Ttalk-Переклад Чат забезпечує можливість спілкування в реальному часі завдяки автоматичному перекладу, а також дозволяє легко перекладати тексти іноземними мовами для використання в інших додатках. На рисунку 1.8 представлено інтерфейс Android додатку «Ttalk-Перевод чат». На рисунку 1.8 наведено інтерфейс Android-додатка «Ttalk-Перевод чат».

Рисунок 1.8 – Інтерфейс Android додатку «Ttalk-Перевод чат»

Переваги:

– детальне відображення розкладу (нумерація тижнів, карточка деталей пари, маркування пар);

– журнал відвідуваності;

Недоліки:

– відсутність української локалізації;

– нестабільність роботи програми;

– незрозумілий інтерфейс.

У процесі проведеного аналізу існуючих рішень встановлено, що жодне з них не забезпечує повної відповідності поставленим вимогам, хоча окремі реалізації частково вирішують відповідні завдання. Було також

ідентифіковано низку їхніх недоліків, а також виявлено невідповідність таких рішень для ефективного розв’язання висунутих проблем.

1.4 Постановка задачі дослідження

На основі зібраної інформації були сформовані ключові концепції, що стали основою цього проєкту. Аналіз показав, що спілкування з людиною, яка зовсім не розуміє вашої мови, є досить складним завданням, особливо якщо ви також не володієте мовою співрозмовника. Хоча в інтернеті можна знайти чимало рішень для таких ситуацій, їхня зручність і надійність залишаються під питанням. Саме тому був запропонований інтернет-чат із функцією автоматичного перекладу повідомлень, який значно спрощує комунікацію. [7].

Аналіз існуючих систем обміну повідомленнями показав, що переклад часто виконується з помилками та затримками, а сам процес перекладу виявляється незручним для користувачів. Система повинна забезпечувати автоматичний переклад повідомлень у режимі реального часу, а також інформувати користувача про отримання нових повідомлень.

Основною ціллю даного програмного модуля, що розробляється, є автоматичний переклад повідомлень в реальному часі. До функціональних вимог можна віднести:

– система повинна проводити автоматичний переклад повідомлень в реальному часі;

– система повинна надавати змогу користувачеві переглядати його повідомлення;

– система повинна містити базу даних для зберігання повідомлень

– система повинна забезпечувати користувача управлінням повідомленнями;

– система містити певний рівень захисту даних користувача.

До нефункціональних вимог можна віднести:

– система повинна мати високу точність перекладу не менше 80%;

– кількість помилкових тривог повинна зводитись до мінімуму;

– система повинна бути проста в експлуатації;

– система повинна бути надійною та працювати без збоїв і помилок;

– система повинна задовольняти вимоги користувача.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ АВТОМАТИЧНОГО ПЕРЕКЛАДУ ПОВІДОМЛЕНЬ В ІНТЕРНЕТ ЧАТІ

2.1 Загальна структура програмного модуля

Проаналізувавши роботу інтернет чатів можна чітко побачити їх недоліки, а саме проблема комунікації людей, які спілкуються на різних мовах, незручні інтерфейси користувачів та платна підписка для взаємодії із додатком. Це є основними недоліками в існуючих аналогів розроблюваного програмного модуля Тому можна чітко визначити вимоги до розроблюваної системи яка буде реалізовувати дані функції:

- процес реєстрації;
- процес авторизації;
- процес пошуку співрозмовників ;
- процес перекладу повідомлень;
- процес обміну повідомленнями;
- процес обробки інформаційних запитів.

Розглядаємо детально кожен із представлених вище бізнес-процесів. При роботі з системою першочергово необхідно пройти реєстрацію.

На етапі реєстрації, майбутній користувач повинен надати інформацію необхідну для взаємодії з системою. Це має забезпечити захист від не санкціонованого доступу до профілю користувача.

Надані відомості повинні забезпечити ідентифікацію користувача та надання функціоналу в залежності від його прав.

Для активації усього спектра прав необхідно активувати профіль за допомогою пін-коду який відправляється на e-mail адресу користувача.

На рисунку 2.1 наведено загальну структуру програмного модуля.

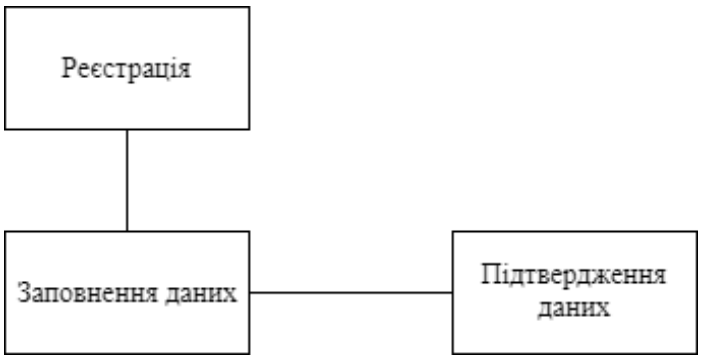


Рисунок 2.1 – Діаграма структури бізнес-процесу «Реєстрація»

Характеристику бізнес-процесу «Реєстрація» наведено в таблиці 2.1.

Після успішної реєстрації для постійної роботи з системою необхідно проходити процес авторизації. Після успішної авторизації дані зберігаються, щоб позбавити необхідності повторювати даний процес при наступних входах в систему.

Таблиця 2.1 - Характеристика бізнес-процесу Реєстрація

Назва характеристики	Значення характеристики
Ім’я бізнес-процесу	Реєстрація
Основні учасники	Користувачі
Вхідна подія	Запуск додатку
Вхідні документи	Логін, пароль, e-mail, прізвище та ім’я.
Вихідна подія	Звіт по успішності реєстрації
Вихідні документи	Особистий кабінет користувача
Клієнт бізнес-процесу	

На рисунку 2.2 наведено архітектуру програмної

с и с т е м и.

Рисунок 2.2 – Діаграма структури бізнес-процесу «Авторизація»

На етапі пошуку користувач отримує список доступних співрозмовників на основі заданих ним фільтрів.

Характеристику бізнес-процесу «Авторизація» наведено в таблиці 2.2

На етапі пошуку інформації про структурний підрозділ користувач отримує наявні дані про певного співрозмовника.

Таблиця 2.2 - Характеристика бізнес-процесу Авторизація

Назва характеристики	Значення характеристики
Ім’я бізнес-процесу	Авторизація
Основні учасники	Користувач
Вхідна подія	Запуск додатку
Вхідні документи	Логін та пароль
Вихідна подія	Звіт по успішності авторизації
Вихідні документи	Особистий кабінет користувача

На етапі «Обмін повідомленнями» відбувається комунікація користувачів між собою шляхом обміну текстових, файлових та графічних даних.

На рисунку 2.3 представлено діаграму структури бізнес-процесу «Обмін повідомленнями».

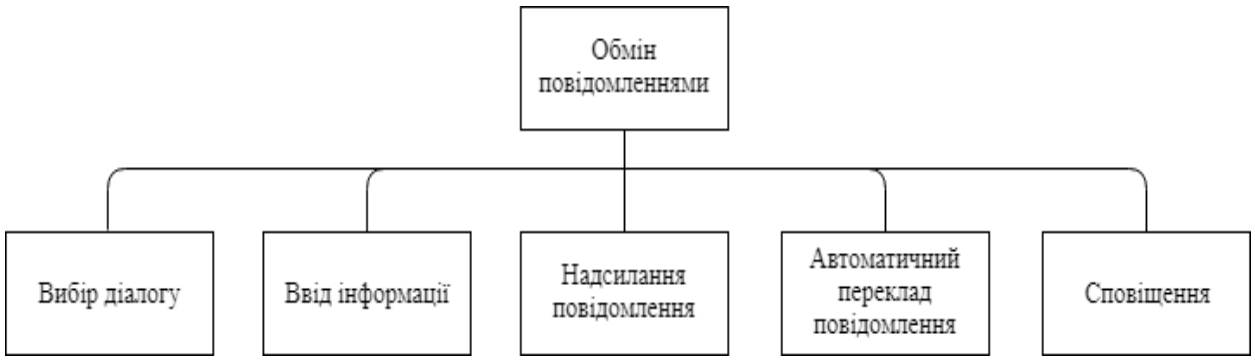


Рисунок 2.3 – Діаграма структури бізнес-процесу «Обмін повідомленнями»

Характеристику бізнес-процесу «Обмін повідомленнями» наведено в таблиці 2.3

Таблиця 2.3 - Характеристика бізнес-процесу Обмін повідомленнями

Назва характеристики	Значення характеристики
Ім’я бізнес-процесу	Обмін повідомленнями
Основні учасники	Користувач
Вхідна подія	Вибір адресата(-ів)
Вхідні документи	Сторінка із списком контактів
Вихідна подія	Надіслане повідомлення
Вихідні документи	Сторінка переписки з адресатом(-ами)
Клієнт бізнес-процесу	Користувач

2.2 Розробка архітектури програмної системи

Розробка даного продукту буде здійснюватись за допомогою патерну MVVM архітектури.

Патерн MVVM (Model-View-ViewModel) дозволяє відокремити логіку додатка від візуальної частини (подання). Даний патерн є архітектурним, тобто він задає загальну архітектуру програми.

Даний патерн був представлений як модифікація шаблону Presentation Model і був спочатку націлений на розробку додатків в WPF.І хоча зараз цей патерн вийшов за межі WPF і застосовується у най різномантніших технологіях, в тому числі при розробці під Android, iOS, проте WPF є досить показовою технологією, яка розкриває можливості даного патерну [9].

MVVM складається з трьох компонентів: моделі (Model), моделі подання (ViewModel) і уявлення (View).

Model описує використовувані в додатку дані.Моделі можуть містити логіку, безпосередньо пов'язану цими даними, наприклад, логіку валідації властивостей моделі.У той же час модель не повинна містити ніякої логіки, пов'язаної з відображенням даних і взаємодією з візуальними елементами управління.

View або подання визначає візуальний інтерфейс, через який користувач взаємодіє з додатком. Стосовно до WPF уявлення - це код в xaml, який визначає інтерфейс у вигляді кнопок, текстових полів та інших візуальних елементів.

Хоча вікно (клас Window) в WPF може містити як інтерфейс в xaml, так і прив'язаний до нього код C #, проте в ідеалі код C # не повинен містити якийсь логіки, крім хіба що конструктора, який викликає метод InitializeComponent і виконує початкову ініціалізацію вікна. Вся ж основна логіка додатки виноситься в компонент ViewModel.

Однак іноді в файлі пов'язаного коду все може знаходитися певна логіка, яку важко реалізувати в рамках паттерна MVVM у ViewModel [10].

Подання і не виконує жодних події за рідкісним винятком, а виконує дії в основному за допомогою команд.

ViewModel або модель представлення пов'язує модель і уявлення через механізм прив'язки даних. Якщо в моделі змінюються значення властивостей, при реалізації моделлю інтерфейсу INotifyPropertyChanged автоматично йде зміна відображуваних даних в поданні, хоча безпосередньо модель і уявлення не пов'язані.

ViewModel також містить логіку по отриманню даних з моделі, які потім передаються в уявлення. І також VewModel визначає логіку по оновленню даних в моделі.

Оскільки елементи уявлення, тобто візуальні компоненти типу кнопок, не використовують події, то уявлення взаємодіє з ViewModel за допомогою команд.

Наприклад, користувач хоче зберегти введені в текстове поле дані. Він натискає на кнопку і тим самим відправляє команду під ViewModel. А ViewModel вже отримує передані дані і відповідно до них оновлює модель.

Підсумком застосування патерну MVVM є функціональний розподіл програми на три компонента, які простіше розробляти і тестувати, а також в подальшому модифікувати і підтримувати. Робота патерну представлена на рисунку 2.4. На рисунку 2.4 наведено структуру бази даних програмного модуля.

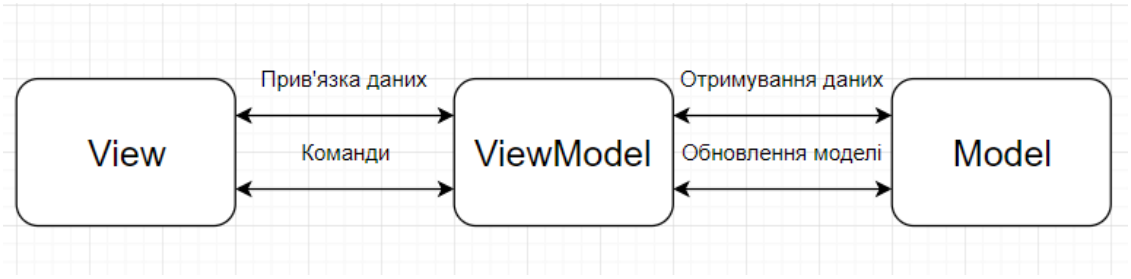


Рисунок 2.4 – Патерн MVVM

Для розробки системи програми для обміну повідомленнями та їх автоматчним перекладом було обрано трирівневу архітектуру клієнт-сервер. Трирівнева архітектура представлена на рисунку 2.5 включає в себе такі компоненти як: клієнтський додаток , підключений до сервер, який в свою чергу підключений до сервера бази даних. На рисунку 2.5 наведено діаграму взаємозв'язків між таблицями бази даних.

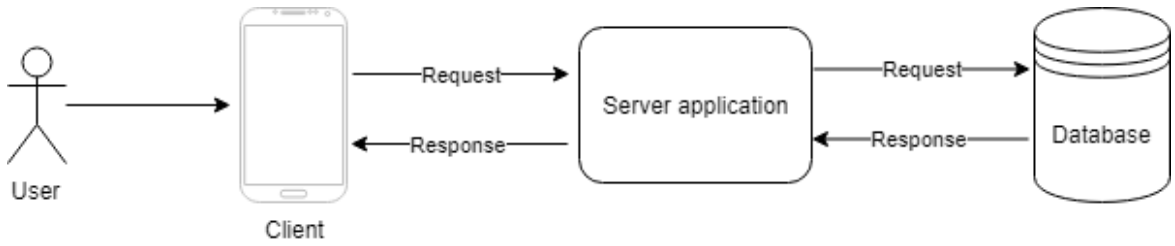


Рисунок 2.5 – Структурна схема програмної системи

Детальну взаємодію користувача з системою представлено у вигляді діаграми варіантів рисунку 2.6. Діаграма варіантів використання (use case diagrams) - для моделювання функціональних вимог до системи (у вигляді сценаріїв взаємодії користувачів з системою).

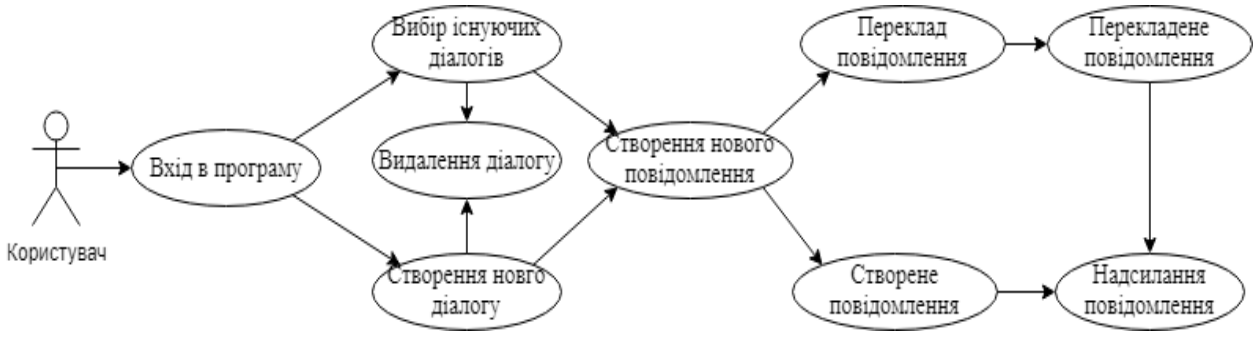


Рисунок 2.6 – Діграма сценаріїв взаємодії користувачів із розроблюваним програмним модулем.

Для формалізації і опису бізнес-процесів в системі було використано мову моделювання IDEF0. В цій методології система представляється, як

сукупність взаємодіючих робіт і функцій. Тут об'єкти представлені ієрархічно, що значно полегшує розуміння предметної області. В IDEF0 розглядаються логічні зв'язки між роботами, а не послідовність їх виконання в часі. В результаті застосування методології отримана модель, яка побудована з чітко сформульованою метою і з єдиною точкою зору, яка складається з діаграм, фрагментів текстів, що мають посилання один на одного [11].

Нижче зображена контекстна діаграма представлена на рисунку 2.7 та зображена діаграма декомпозиції див. рис. 2.8.

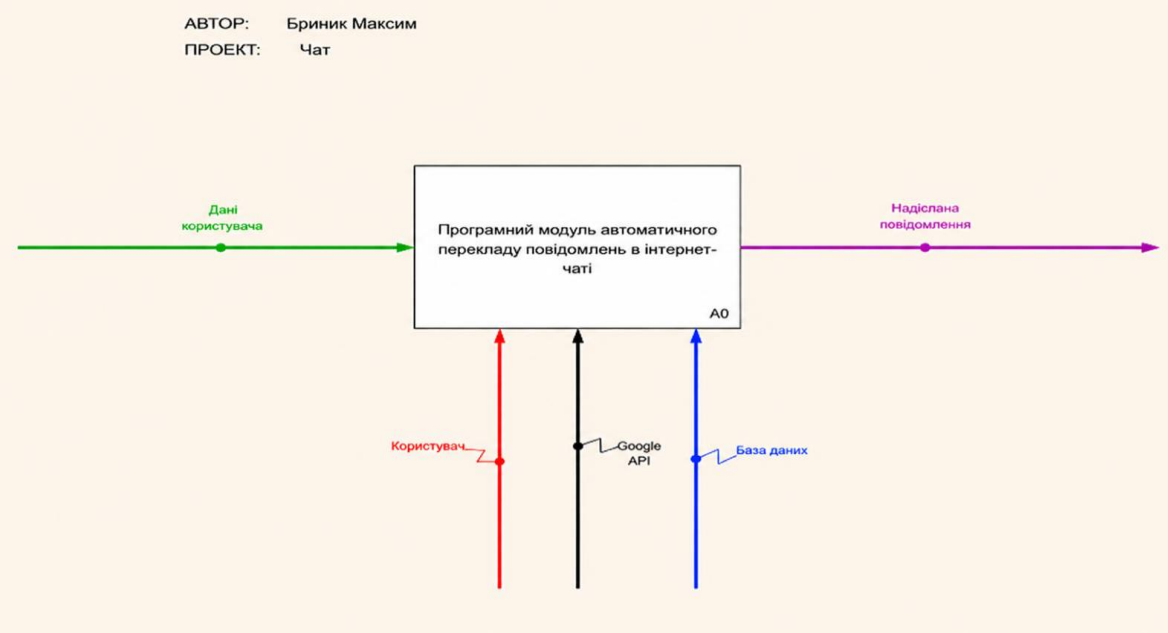


Рисунок 2.7 – Контекстна діаграма системи.

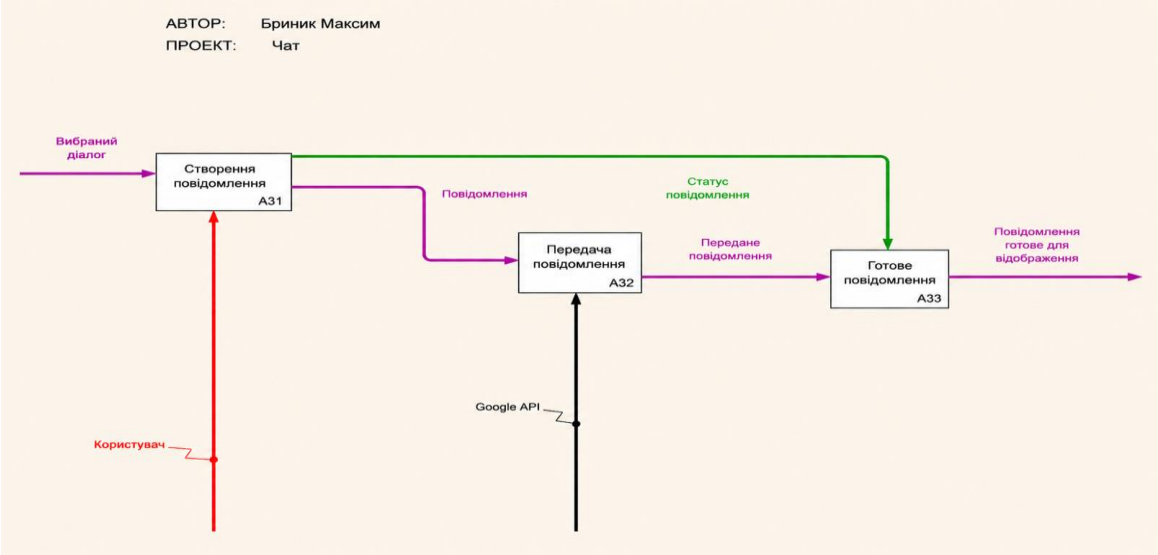


Рисунок 2.8 – Діаграма декомпозиції системи.

Також користувач повинен мати змогу авторизації. Цей процес призначений для того, щоб лише користувач який знає логін та пароль міг увійти на свій профіль. Це є захистом від не санкціонованого входу на його аккаунт. Щоб зайти на профіль користувачеві потрібно ввести дані з окрема логін та пароль. Даний процес представлений у вигляді DFD (Data Flow Diagram) діаграми яка розроблена в середовищі BPwin та представлена на рисунку 2.9.

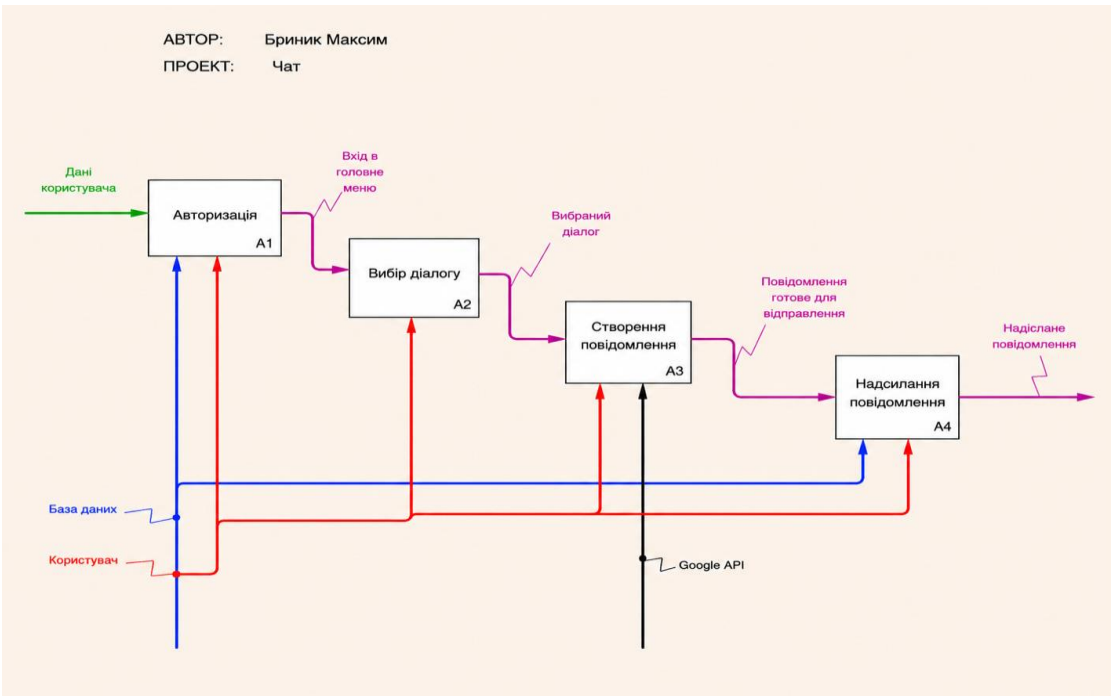


Рисунок 2.9 – Діаграма декомпозиції авторизації користувача

Також користувач повинен мати змогу вибрати існуючий діалог із представленого списку в головному меню або створити новий якщо такого діалогу немає у цьому списку поточних діалогів користувача, для того, щоб перейти до наступного процесу створення повідомлення.

Процес вибору діалогу або створення нового діалогу представлений в вигляді DFD (Data Flow Diagram) діаграми на рисунку представлений нижче

.

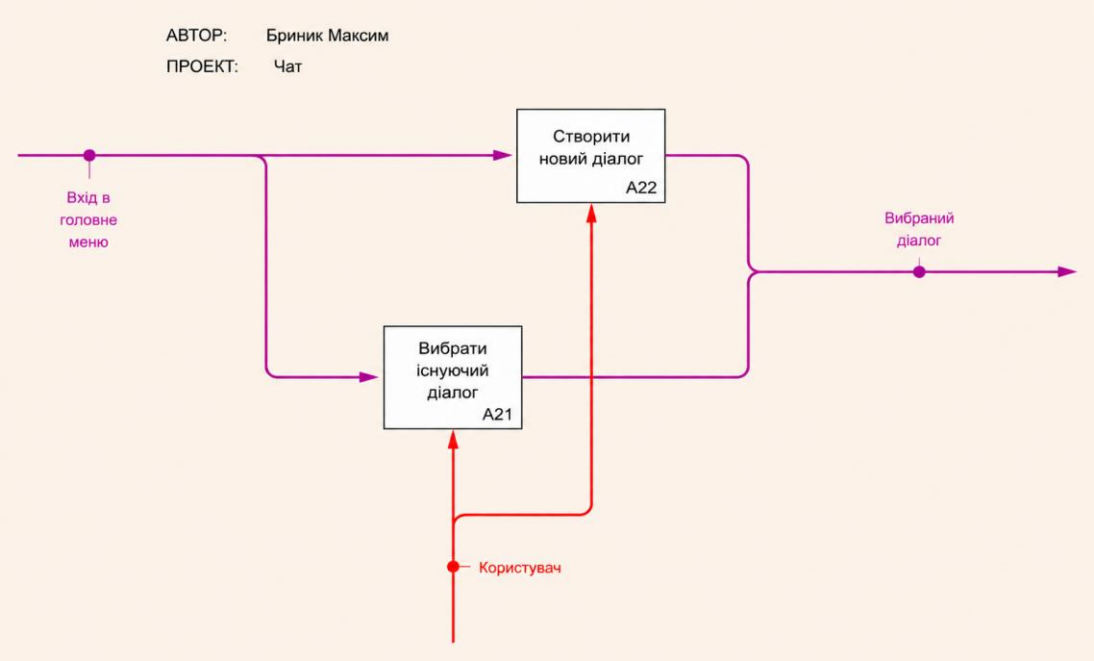


Рисунок 2.10 – Діаграма декомпозиції вибору діалогу або створення нового діалогу

Процес створення користувачем новго повідомлення та його автоматичний переклад представлено у DFD діаграмі на рисунку 2.11.

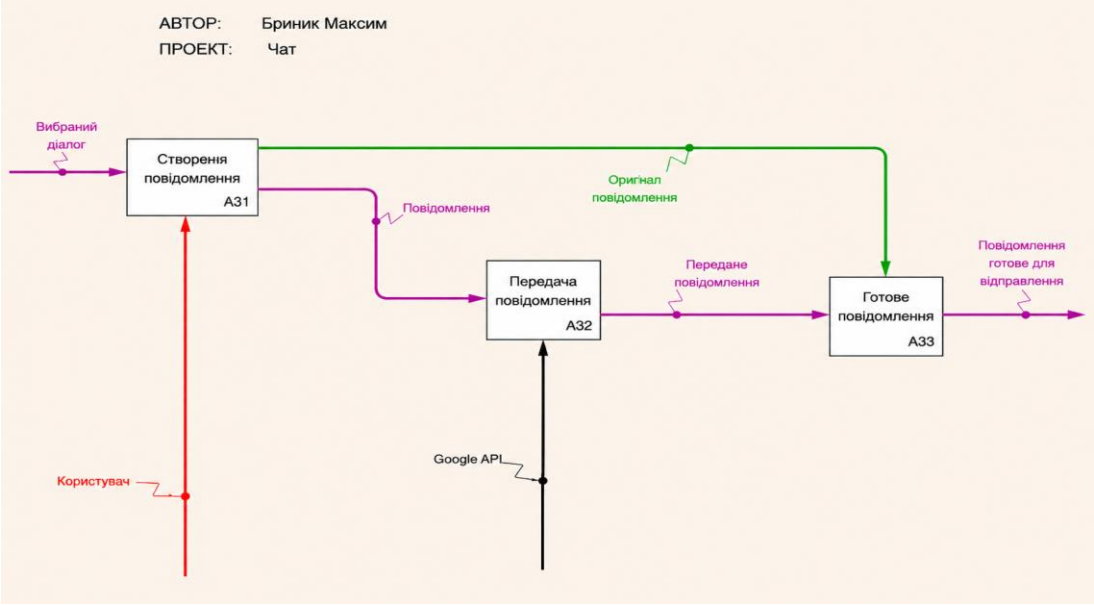


Рисунок 2.11 – Діаграма створення користувачем нового повідомлення з автоматичним перекладом.

Загальну схему роботи даного програмного модуля можна побачити дальше (див. Додаток А).

Дана схема розроблена для того, щоб можна було наглядно побачити весь функціонал програми та зрозуміти що потрібно програмно реалізувати для того аби даний модуль відповідав поставленим цілям.

При запуску програми першим має запускатися локально розташований сервер який реалізує весь обмін повідомленнями між користувачами. Наступним кроком є процес авторизації користувача для того щоб увійти на його профіль. Після введення даних користувача у відповідні поля введення йому буде потрібно натиснути кнопку «sing in» після чого відбудеться перевірка його даних на коректність і у випадку, якщо дані введені некоректно то йому прийдеться ввести їх ще раз, а у випадку якщо дані введено вірно відкриється головне меню де користувач може вибрати діалог для введення та надсилання повідомлень. Також у випадку коли діалогу ще немає у списку користувач може створити новий діалог після чого він з’явиться у спису існуючих діалогів та буде доступний для обміну

повідомленнями в реальному часі з іншим користувачем цього програмного модуля.

Наступним кроком є процес створення повідомлення яке користувач хоче надіслати своєму співрозмовнику. Для цього користувач повинен ввести текст повідомлення в текстове поле і натиснути кнопку надсилання після чого його одержувачу прийде оригінал повідомлення та одразу ж з цим і переклад оригіналу повідомлення на мову якою спілкується його одержувач. Про те самому відправнику повідомлення не буде відображатися переклад надісланих ним повідомлень, а лише повідомлень які він отримав. Оригінал повідомлення перекладається за допомогою Google Translate API яке є у вільному доступі, це процес надсилає запит на обробку оригіналу повідомлення в Google API після чого отримує від нього відповідь у вигляді перекладеного повідомлення.

Тепер якщо користувач захоче написати нове повідомлення він має знову повторити процес створення повідомлення просто ввівши текст свого повідомлення або в протилежному випадку він може вибрати інший діалог та створити новий для нових співрозмовників, або закінчити процес взаємодії цим програмним модулем просто закривши його. Та при наступному запуску програми користувачеві прийдеться пройти весь цикл знову і так кожен раз під час запуску.

2.3 Проектування структури бази даних

Етап проектування бази даних є одним з ключових та від правильності його реалізації залежить функціонування системи. Із доступних моделей представлення даних було обрано реляційну модель даних [12].

Реляційні бази даних використовуються в більшості систем різного рівня складності і фактично стали стандартом довівши свою ефективність та

практичність. Ця база даних сприймається користувачем як набір нормалізованих відношень різного ступеня.

Важливою особливістю реляційної бази даних є процес нормалізації, який має за мету усунути недоліки структури БД, що призводять до шкідливої надлишковості в даних, яка в свою чергу призводить до різноманітних аномалій та порушення цілісності даних.

На платформі .NET для роботи з базою даних буде використано Entity Framework це являє спеціальну об'єктно-орієнтовану технологію на базі фреймворка .NET для роботи з даними. Якщо традиційні засоби ADO.NET дозволяють створювати підключення, команди та інші об'єкти для взаємодії з базами даних, то Entity Framework являє собою більш високий рівень абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працює з об'єктами [13].

В якості СКБД необхідний компактний та багатопотоковий сервер баз даних. Для якого притаманним є висока швидкість, стійкість та простота використання. Ці вимоги виконуються програмним продуктом – MySQL. Враховуючи, що MySQL з відкритим кодом (розповсюджується відповідно до умов ліцензії GPL) вона ідеально вписується в архітектуру системи.

З клієнтської частини з допомогою REST інтерфейсу будуть відправлятись HTTP запити, які в свою чергу з сторони веб-сервера оброблятиме PHP формуючи міст з БД. Відповіді в зворотньому напрямку будуть прямувати у вигляді файлів JSON.

Веб-сокети (Web Sockets) - це передова технологія, яка дозволяє створювати інтерактивне з'єднання між клієнтом (браузером) та сервером для обміну повідомленнями в режимі реального часу[14]. Веб-сокети, на відміну від HTTP, дозволяють працювати з двонаправленим потоком даних, що робить цю технологію абсолютно унікальною. Давайте розберемося, як

працює ця технологія і чим вона відрізняється від HTTP. Схему обміну повідомленнями за допомогою HTTP та Web Sockets буде наведено на рисунках 2.11 та на рисунку 2.12.

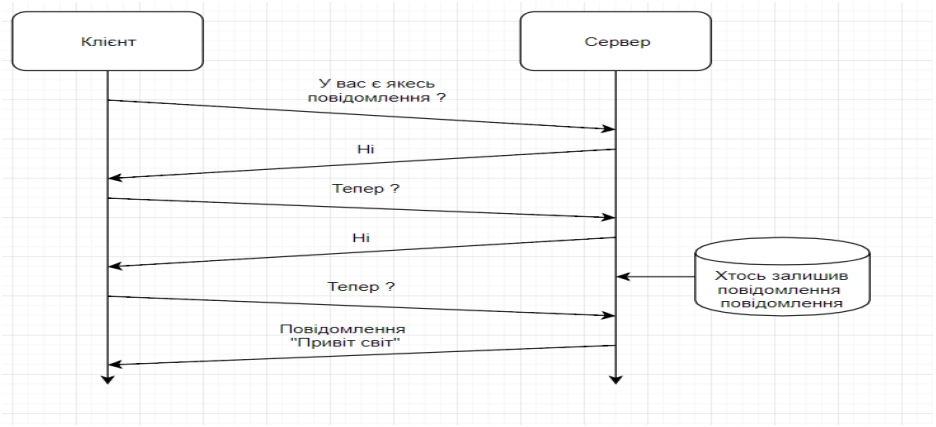


Рисунок 2.11 – Принцип роботи HTTP запитів

На наступному рисунку 2.12 зображено принцип роботи Web Sockets Де можна наглядно побачити різницю між HTTP та Web Sockets.

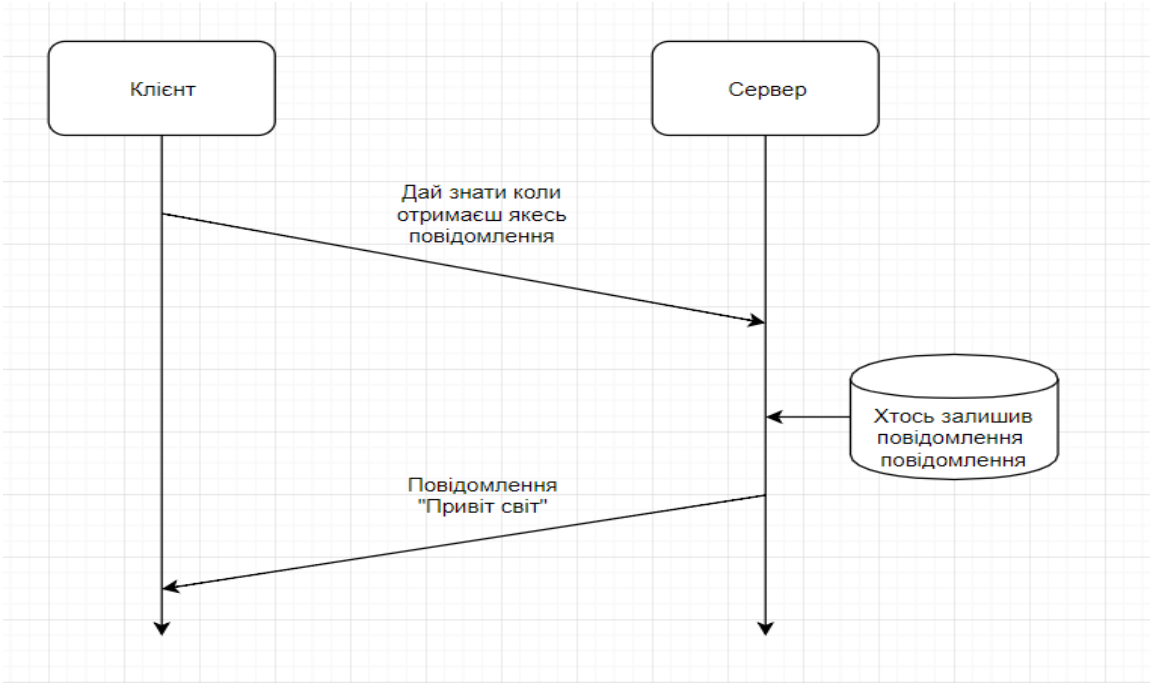


Рисунок 2.12 – Принцип роботи Web Sockets.

Разом з тим, в функціонал клієнтської частини повинні входити процедури для дублювання отриманої інформації в локальну пам’ять пристрою, тобто кешування даних. Це дозволить отримувати доступ до раніше переглянутої інформації без доступу до мережі інтернет.

Після визначення архітектури проектованої бази даних, вимог до неї та особливостей взаємодії можна переходити до її структури за допомогою якої буде відбуватися подальша програмна реалізація бази даних яка містить три таблиці для зберігання даних користувача, для зберігання повідомлень та для аккаунту користувача. Розробка цієї структури значно спростить її програмну реалізацію

Структуру бази даних для подальшої програмної реалізації можна побачити нижче на рисунку 2.13 .

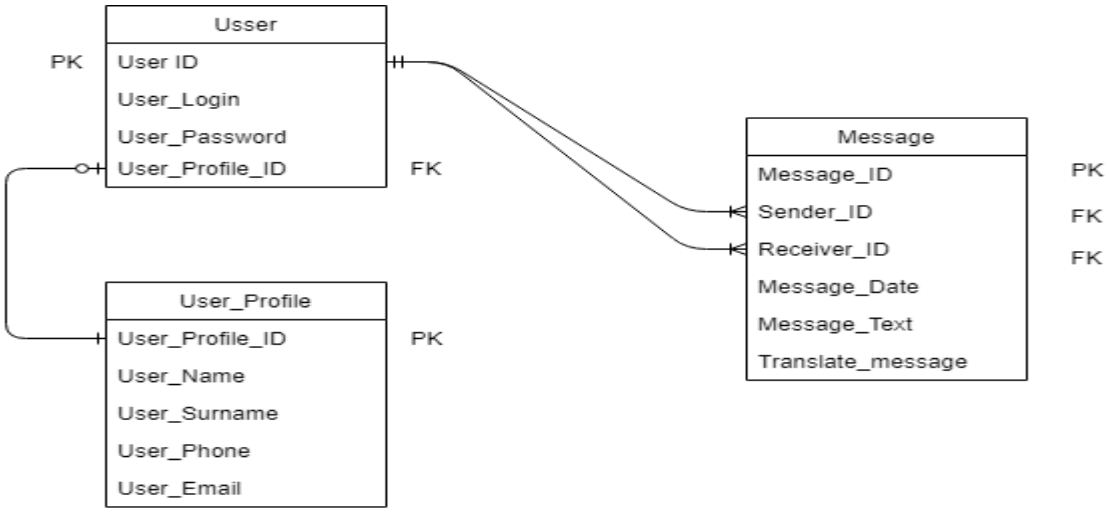


Рисунок 2.13 – Структура бази даних

Для функціонування додатку та програмної реалізації бази даних будуть використовуватись наступні об’єкти та дані про них описані нижче:

User

User_ID;

User_Login;

User_Password;

User_Profile;

Message

Message_ID;

Sender_ID;

Reseiver_ID;

Message_Date;

Message_Text;

Translate_Message;

User Profile

User_Profile_ID;

User_Name;

User_Surname;

User_Phone;

User_Email.

3 ПРОГРАМНО-ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ

3.1 Програмна реалізація бази даних

Для комунікації із системою було обрано СУБД MySQL. На сьогоднішній день MySQL набула широкого застосування і підтримується великою кількістю мов програмування [15]. Основним застосунком якої є керування реляційними БД (тих баз даних, у яких дані зберігаються в певних таблицях завдяки чому забезпечується максимальна швидкість та гнучкість).

Середовищем розробки і адміністрування БД MySQL було обрано MySQL Workbench, оскільки він є безкоштовним та зручним у використанні.

До основних переваг MySQL Workbench можна віднести:

- можливість редагування даних в таблиці, а також можливість легко представити графічну модель БД;

- наявність налагодженого і простого в застосуванні механізму створення зв’язків між таблицями;

- наявність простого редактора SQL-запитів, в результаті виконання

яких отримаємо таблицю із відповідними даними;

Основним інструментом для операцій з таблицями використано SQL запити. Існує 4 основні типи SQL-запитів, а саме:

- вибірка (SELECT назва_колонки_1, назва_колонки_2 FROM назва_таблиці);

- вставка (INSERT INTO назва_таблиці (назва_колонки_1, назва_колонки_2, назва_колонки_3,...) VALUES (значення_1, значення_2, значення_3,...));

- редагування (UPDATE назва_таблиці SET назва_колонки_ = значення_1, назва_колонки_2 = значення_2,... WHERE деяка_колонка = деяке_значення);

- видалення (DELETE FROM назва_таблиці WHERE деяка_колонка = деяке_значення);

Нижче наведено лістинг SQL коду для створення таблиць.

```
CREATE TABLE `Message` (
   `Id` int NOT NULL AUTO_INCREMENT,
   `Message_id` longtext NULL,
   `Sendler_id` longtext NULL,
   `Recevier_id` longtext NULL,
   `Message_date` longtext NULL,
   `Message_Text` longtext NULL,
   `Translate_message` int NOT NULL,
   CONSTRAINT `PK_Message_id` PRIMARY KEY (`Id`)
   CONSTRAINT `FK_Sendler_id` FOREIGN KEY (`Sendler_id `),
   CONSTRAINT `FK_Message_id` FOREIGN KEY (`Message_id `)
);
CREATE TABLE `User` (
   `Id` int NOT NULL AUTO_INCREMENT,
   `User_id` longtext NULL,
   `User_login` longtext NULL,
   `User_Password` longtext NULL,
   `User_Profile_Id` longtext NULL,
```

```
    CONSTRAINT `PK_User_id` PRIMARY KEY (`Id`),
    CONSTRAINT `FK_User_Profile_id` FOREIGN KEY (`User_Profile_id``)
);
CREATE TABLE `User_Profile` (
    `Id` int NOT NULL AUTO_INCREMENT,
    `User_Profile_id` longtext NULL,
    `User_Name` longtext NULL,
    `User_Surname` longtext NULL,
    `User_Phone` longtext NULL,
    `User_Email` longtext NULL,
    CONSTRAINT `PK_User_profile_id` PRIMARY KEY (`Id`),
REFERENCES `User_profile` (`Id`) ON DELETE CASCADE
);
```

Для роботи з базою даних буде використовуватись entity framework. На платформі .NET для роботи з базою даних буде використано Entity Framework це являє спеціальну об'єктно-орієнтовану технологію на базі фреймворка .NET для роботи з даними. Якщо традиційні засоби ADO.NET дозволяють створювати підключення, команди та інші об'єкти для взаємодії з базами даних, то Entity Framework являє собою більш високий рівень абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища.

Entity Framework зараз дуже популярна та швидка ORM технологія, яка здійснює перехід від концептуальної моделі даних до реляційної [16].

В якості СКБД необхідний компактний та багато-потоковий сервер баз даних. Для якого притаманним є висока швидкість, стійкість та простота використання. Враховуючи, що MySQL з відкритим кодом (розповсюджується відповідно до умов ліцензії GPL) вона ідеально вписується в архітектуру системи.

3.2 Програмна реалізація проекту

При створенні програмного продукту важливо, яким чином він буде реалізований, на якій платформі, на якій мові програмування, адже від цього залежить надійність, продуктивність, якість і розширюваність роботи розроблюваної програмної системи.

Оскільки система є платформною, то в процесі розробки буде використовуватись мова C# — об'єктно-орієнтована мова програмування з системою строгої типізації для платформи .NET. Для Платформа .NET Framework грає роль своєрідної операційної системи всередині операційної системи, яка дозволить створити нашу програму[16].

Microsoft .NET (читається дот-нет) — програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і мобільних додатків та веб-застосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Однією з ідей .NET є сумісність служб, написаних різними мовами [10]. Хоча ця можливість рекламується Microsoft як перевага .NET, платформа Java має таку саму можливість.

Кожна бібліотека в .NET має свідчення про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок. .NET крос-платформова технологія, в цей час існує реалізація для платформи Microsoft Windows, FreeBSD (від Microsoft) і варіант технології для ОС Linux в проекті Mono (в рамках угоди між Microsoft з Novell), DotGNU [17].

Захист авторських прав відноситься до створення середовищ виконання (CLR — Common Language Runtime) для програм .NET. Компілятори для .NET випускаються багатьма фірмами для різних мов вільно.

.NET поділяється на дві основні частини — середовище виконання (по суті віртуальна машина) та інструментарій розробки.

Середовища розробки .NET-програм: Visual Studio .NET (C++, C#, J#), SharpDevelop, Borland Developer Studio (Delphi, C#) тощо. Середовище Eclipse має додаток для розробки .NET-програм. Застосовні програми також можна розроблювати в текстовому редакторі та використовувати консольний

компілятор [18]. Для розробки даної системи буде використано Visual Studio 2017.

Для розробки крос-платформенного мобільного додатку буде використовуватись технологія Xamarin. Вирішено використовувати саме цю платформу, тому, що вона має деякі переваги. Є певні статистичні дані, що значна частина мобільних додатків створюється більш ніж для однієї платформи, наприклад, для Android і IOS. Однак неминуче розробники стикаються з наступними труднощами:

Відмінність в підходах побудова графічного інтерфейсу так чи інакше впливає на розробку. Розробники змушені підлаштовувати додаток під вимоги до інтерфейсу на конкретній платформі

Різні API - розбіжність у програмних інтерфейсів і реалізації тих чи інших функцій також вимагає від програміста облік цих специфічних особливостей. Різні платформи для розробки. Наприклад, щоб створювати додатки для IOS нам необхідна відповідне середовище - Mac OS X і ряд спеціальних інструментів, типу XCode. А в якості мови програмування вибирається Objective-C або Swift. Для Android ми можемо використовувати найрізноманітніший набір середовищ - Android Studio, Eclipse і т. д. Але тут для переважної більшості додатків застосовується Java.

Для реалізації даного програмного модуля автоматичного перекладу повідомлень в реальному часі потрібно реалізувати три основних частини: База даних де зберігатимуться дані користувачів, сервер за допомогою якого здійснюється обмін повідомленнями та їх автоматичний переклад в реальному часі , клієнтська частина за допомогою якої безпосередньо відбувається взаємодія користувачів із даним програмним модулем.

Також для для зв’язку із клієнтської частини із сервером буде використовуватися підхід REST[19].Структура побудови архітектури додатку представлена на рисунку 3.1 На рисунку 3.1 наведено структуру бази даних програмного забезпечення.

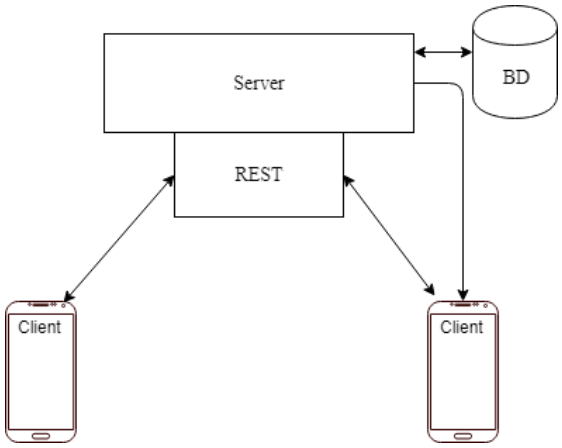


Рисунок 3.1–Структура побудови архітектури

Для того, щоб краще зрозуміти процес функціонування розроблюваного додатку було побудовано діаграми станів основних процесів такі як авторизація та створення нового повідомлення. На рисунку 3.2 наведено вікно авторизації користувача.

Діаграми станів було представлено на рисунку 3.2.

Рисунок 3.2 – Діаграма станів авторизації користувача

На рисунку 3.3 наведено форму реєстрації користувача.

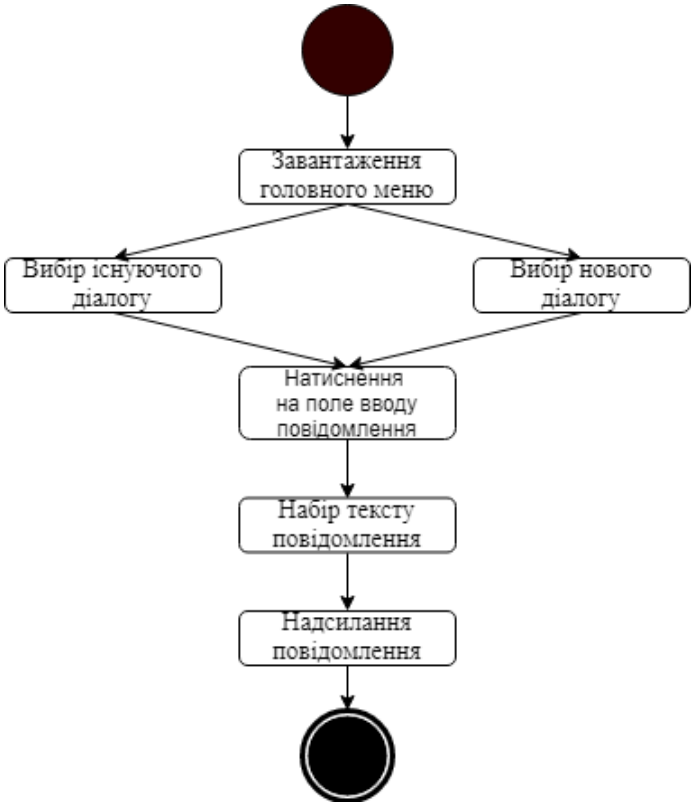


Рисунок 3.3 – Діаграма станів створення повідомлення користувачем

Основне завдання діаграми станів – це показати, що проектована система може бути описана як скінченне число станів. Така діаграма використовується для того, щоб надати абстрактний опис поведінки системи та її різних функцій. Ця поведінка — це проаналізована та відтворена послідовність подій, які відбуваються в одному і більше можливих станах системи. Зазвичай, одна діаграма описує один об’єкт і відслідковує зміну станів цього об’єкта в системі. За її допомогою можна наочно проаналізувати стани об’єктів та визначати можливі розгалуження та помилки.

При проектуванні інтерфейсу системи основна увага приділялася простоті та легкості сприйняття. Завдяки цьому у користувача який буде вперше працювати із системою не виникне ніяких труднощів, оскільки все буде зрозуміло на інтуїтивному рівні.

Також нижче було розроблено діаграму послідовності для того, щоб відобразити взаємодію об’єктів впорядкованих за часом на (рисунку 3.4).

Діаграма послідовності включає в себе виключно користувача та його взаємодію із системою в часі. Пунктирна лінія – лінія життєвого циклу роботи програмного модуля яка являє собою єдиний об’єкт на діаграмі послідовності та втілює в собі позначення періоду часу, упродовж якого користувач взаємодіє із модулем.На рисунку 3.4 наведено головне вікно програмного модуля.

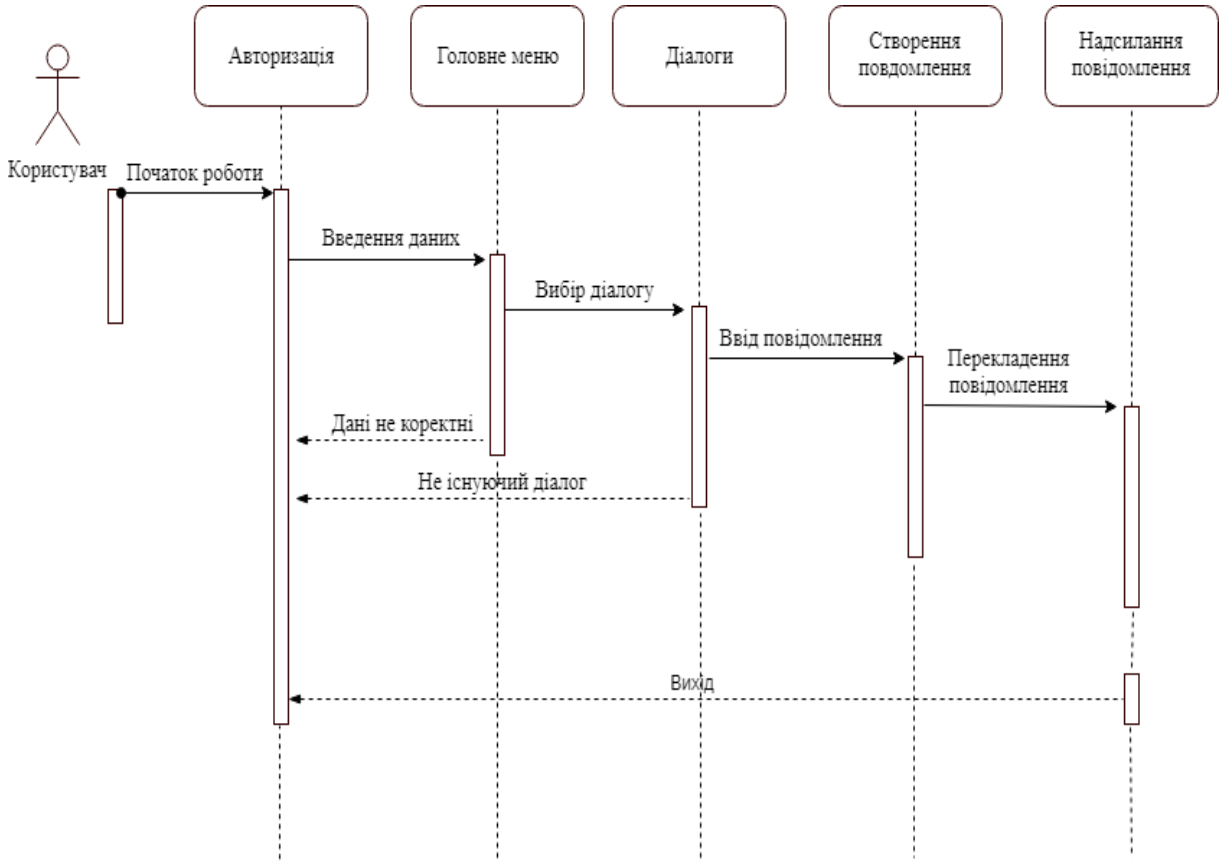


Рисунок 3.4 – Діаграма послідовності взаємодії з модулем

Для розробки інтерфейсу будуть створюватися сторінки та розмітка на мові XAML.

XAML – це декларативна мова розмітки яка значно спрощує розмітку інтерфейсів на базі .NET Framework з точки зору моделі програмування[7]. Можна створити видимі елементи інтерфейсу які потім легко відокремити від

основної логіки часу виконання, використовуючи файли коду програмної частини[20]

Нижче зображені основні класи та файли що описують розмітку сторінок представлено на рисунку 3.5

Для розробки системи використовується архітектурний шаблон проектування MVVM, згідно якого кожному представленню відповідає ViewModel клас, який містить основне бізнес логіку. Для прикладу в даному проекті клас Translator.cs – містить логіку для додавання, редагування, перекладу слів та у лістинг коду (дивитись додаток В).

Програмний модуль буде містити такі основні сторінки та класи :

- LoginPage – сторінка авторизації;
- MainPage – основна сторінка;
- MyMasterDetailPage – сторінка з висувним боковим меню;
- HomePage – сторінка з існуючими діалогами користувача;
- ProfilePage – сторінка користувача.

Класи представлено діаграмою класів у (дивитись додаток Б)

Дана діаграма класів будується автомачтично в середовищі розробки даного програмного модуля Microsoft Visual Studio – це, інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

Основними класами програмного модуля є:

- Program.cs – цей клас повинен бути у будь-якому проекті ASP.NET Core, із цього класу по суті починає виконуватись програма а також необхідний об’єкт IWebHost для якого використовується об’єкт IWebHostBuilder, цей об’єкт забезпечує розгортання програми;

- Startup.cs – цей клас налаштовує усі потрібні сервіси які додаток буде використовувати та сам є вхідною точкою в додаток. Також встановлює компоненти для обробки запиту. Ще цей класмістить обов'язковий метод Configure у якому встановлюється як додаток буде обробляти запит він є обов'язковим.

- ChatViewModel.cs – Цей клас містить усю логіку обміну повідомленнями між користувачами та сам процес перекладу повідомлень.

Загальну структуру класів зображено нижче на рисунку 3.5.

На рисунку 3.5 наведено інтерфейс списку чатів.

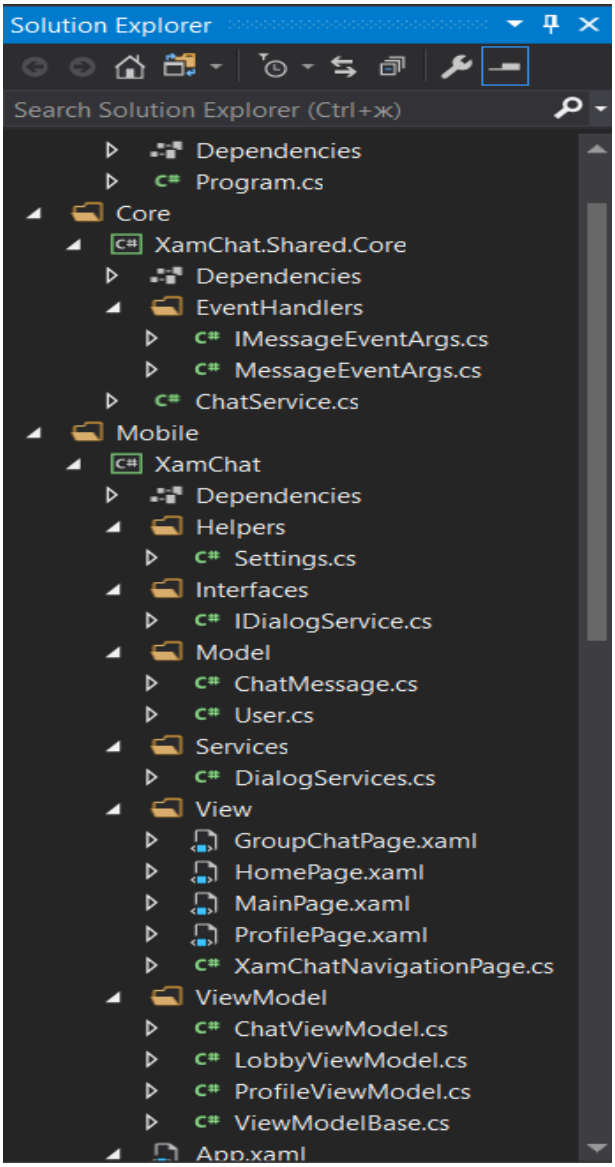


Рисунок 3.5 –. Структура основних класів та сторінок мобільного додатку

При компіляції та розгортанні мобільного додатку на його «рідну» платформу, наприклад при запуску Android версії, код розмітки XAML буде транслюватися у XML представлений у додатку Д.

- Translator.cs – у цьому класі міститься здійснення перекладу повідомлень користувача за допомогою Google Translate API. Цей клас посилає запит на обробку даних до Google Translate API після чого повертає оброблені дані.
- ChatViewModel.cs – Клас який являє собою модель представлення та через який створено взаємодію із сервером.
- ProfileViewModel.cs – Клас який являє собою модель представлення взаємодії з профілем користувача.
- Main.cs – Являє собою основний клас як і в будь якому проекті розроблюваному на мові C#.

Також даний додаток містить ще такі класи як:

- Setting.cs – Клас налаштувань
- User.cs – Клас реалізовано користувача
- ChatMessage.cs – Клас в якому реалізовано повідомлення
- ChatDialogService.cs – Сервіс де реалізовано обмін повідомленнями

Розроблюваний мобільний додаток використовує Google Translate API, для його використання потрібно створити клас, який здійснює переклад представлено у додатку Д.

Цей модуль приймає параметрами текст, мову оригіналу та мову для перекладу, відправляє запит на google.api та повертає результат, безпосередньо перекладений текст. Однією з найцікавіших та найкорисніших функцій для користувача є переклад повідомлень . Для реалізації цього функціоналу було використано Android API, яке надає зручні інструменти для подібних потреб.

Вимоги до апаратного забезпечення:

- смартфон на основі системи Android версії 4.4 та вище;
- підключення до мережі;
- 50МБ вільного простору на пристрої.

3.3 Інтерфейси користувача

Для того щоб користувачеві було легко та приємно коростуватися даним додатком потрібно зробити якомога простіший та зрозулий для усіх інтрерфейс. Тоді і самим додатком його аудиторія буде користуватися частіше та доше.

Адже основну взаємодію із додатком користувач проводить за допомогою інтерфейсу. Тому при складних та заплутаних інтерфесах користувачі просто відмовляються працювати із додатком, а також цей інтерфейс повинен мати приємний естетичний вигляд.

Інтерфейси користувача для даного програмоного модуля будуть виглядати наступним чином. При запуску програми першим буде появлятись вікно авторизації де користувач має ввести адресу свого електронного ящика та пароль, після чого натиснути на кнопку «Login in». Це має бути певним етапом захисту профілю користувача від несанкціонованих входів на його аккаунт. На рисунку 3.6 наведено інтерфейс обміну повідомленнями.

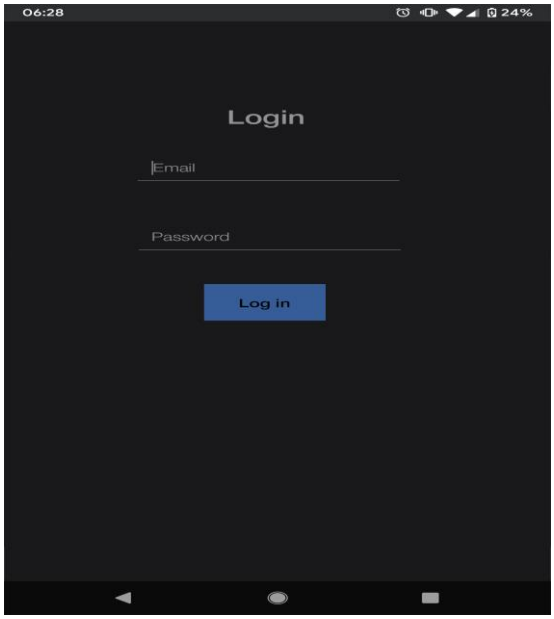


Рисунок 3.6 – Інтерфейс авторизації користувача

Наступним після вдалого проходження авторизації буде з’являтися інтерфейс головного меню де користувач може вибрати один з існуючих діалогів для спілкування а також створити новий при необхідності. Ще можна здійснити пошук діалогів якщо їх список є надто великим, також можна видалити уже існуючий діалог із цього списку. Інтерфейс меню з існуючими діалогами зображено на рисунку 3.7. На рисунку 3.7 наведено процес автоматичного перекладу повідомлень.

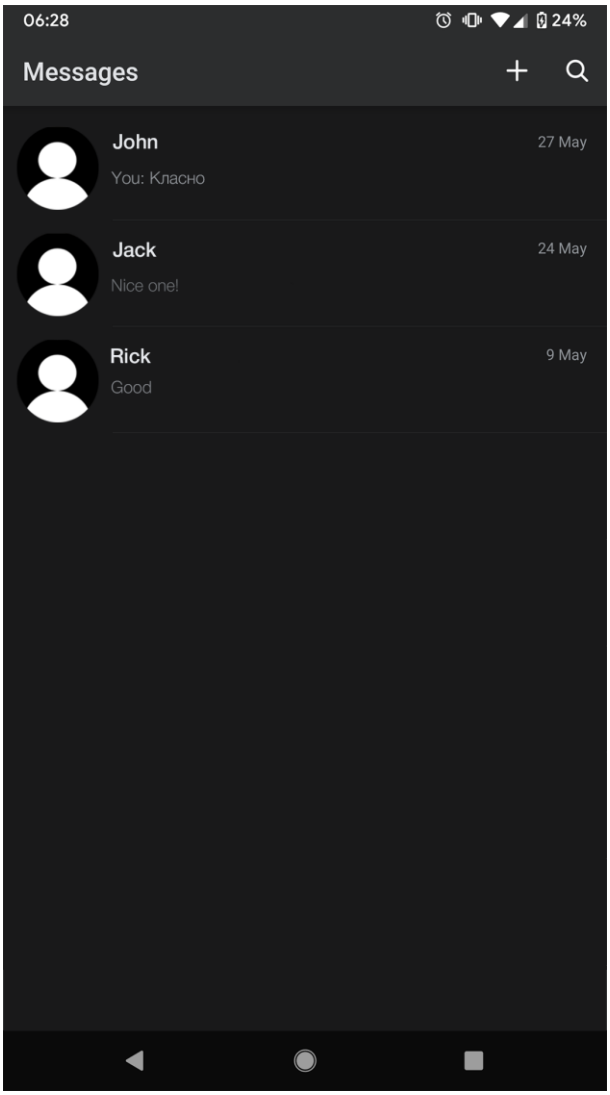


Рисунок 3.7 – Інтерфейс головного меню із існуючими діалогами

Після наведеного вище інтерфейсу, якщо користувач вибере одни із вже існуючих діалогів йому відкриє чат з даним користувачем де буде відображатися ім’я одержувача повідомлень, сама історія повідомлень, та поле для введення тексту з кнопкою надіслати. Повідомлення надіслані та отримані повідомлення будуть відображатись з різних сторін екрану для того, було простіше зрозуміти де чиє повідомлення.

Також користувач отримуватиме не тільки оригінал повідомлення а також і його автоматичний переклад. Кожен користувач бачить переклад лише тих повідомлень які йому прийшли а переклад своїх повідомлень для нього не відображається. Для цього нижче буде представлено два рисунки 3.8 та 3.9 для того, щоб з імітувати діалог двох користувачів даного програмного

модуля та побачити як відображається обмін повідомленнями та їх автоматичний переклад.На рисунку 3.8 наведено результати тестування програмного забезпечення.

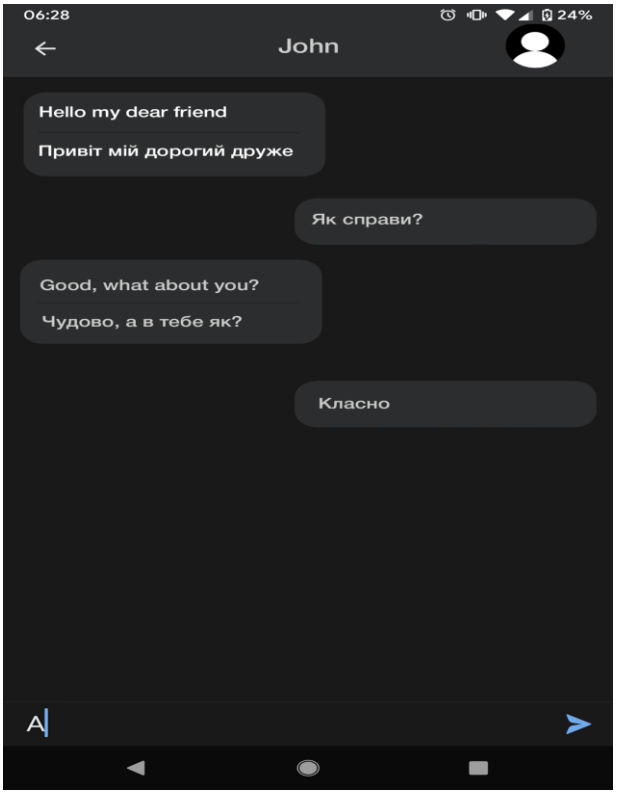


Рисунок 3.8 – Інтерфейс чату першого користувача

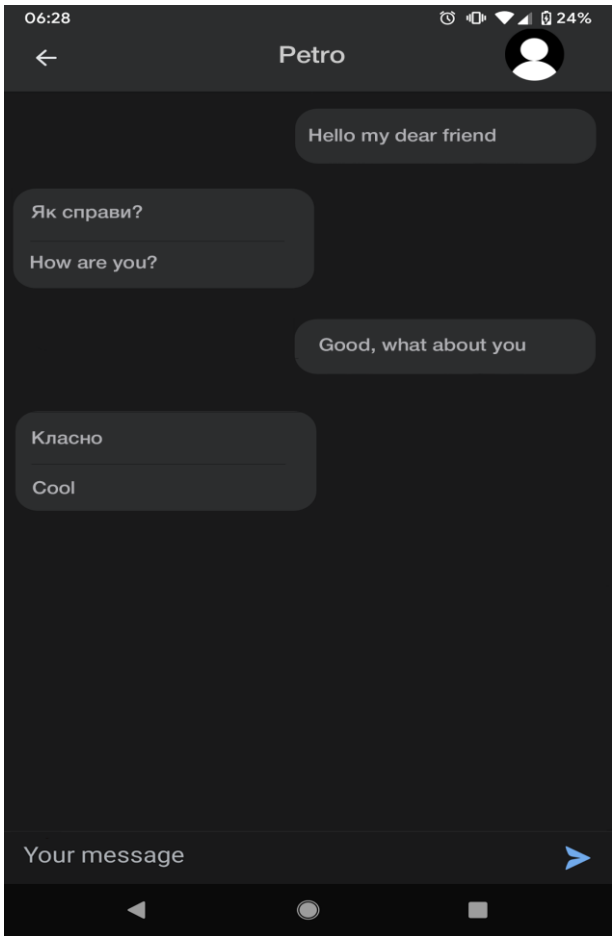


Рисунок 3.8 – Інтерфейс чату другого користувача

3.4 Тестування програмного забезпечення

Тестування – завершальний етап у розробці будь-якого програмного продукту, який відіграє дуже важливу роль в створенні якісного ПЗ. Сучасні методи розробки ПЗ передбачають що тестування відбувається під час розробки, тобто розроблені модулі тестуються по мірі їх розробки. З рівнем складності програми зростає і рівень складності її тестування. Основним завданням тестування ПЗ є визначення рівня якості програмного продукту. Ця задача реалізовується за допомогою двох основних чинників, а саме:

- демонстрація учасникам проекту відповідності програми до поставлених під час проектування вимог;

- визначення факторів, в ході яких певні функції системи різняться від тих або працюють не так, як описано в специфікації.

Розробка та підтримка ПЗ це дуже складна річ, тому практично неможливо зразу ж написати ідеальний програмний продукт який б не містив у собі ніяких помилок. І як буває у більшості випадків, винуватцями появи помилок у розроблюваному програмному продукті є самі програмісти, адже не варто відкидати людський фактор. Отож задачею QA спеціалістів та кваліфікованого програміста є не лише написання програми, а й налаштування роботи та забезпечення якості. Опираючись на це, тестування даного програмного модуля онлайн-чату з автоматичним перекладом повідомлень у реальному часі є не лише доцільним, а й необхідним етапом розробки. Існують три основних класифікації тестування такі як:

- функціональні;
- нефункціональні;
- пов’язані зі змінами.

Функціональні тестування можуть бути представлене на усіх етапах тестування, ці тести базуються на взаємодії з іншими системами та на функціях та особливостях програмного забезпечення. Такі тестування розглядає лише зовнішню поведінку системи[21].

Мета функціонального тестування иявлення невідповідностей між реальною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікації і вимог. Функціональні тести повинні охоплювати всі реалізовані функції з урахуванням найбільш ймовірних типів помилок.

Також функціональні тестування поділяються на кілька видів:

- тестування функцій;
- тестування безпеки;
- тестування взаємодії.

Нефункціональні тестування необхідні для визначення характеристик програмного забезпечення. Результати цих тестів можуть бути визначені різними величинами. Саме ці тестування визначають як функціонує система. Види цих тестувань перелічені нижче:

- тестування навантаження;
- стресове тестування;
- об’ємне тестування;
- тестування надійності;
- тестування встановлення
- тестування на відмову;
- тестування на зручність у користуванні;
- тестування конфігурацій.

Пов’язані зі змінами види тестувань – це повторне тестування після виправлення багу/помилки/дефекту у програмному забезпеченні для того, щоб впевнитись у тому, що проблему усунено остаточно. Такі види тестування повинні проводитись після завантаження програмного забезпечення, щоб впевнитись у тому, що раніше виявлені помилки відсутні, а також для підтвердження роботи функціоналу додатку. Види тестувань пов’язаних із змінами представлені внизу:

- димові тестування;
- тестування збірки;
- тестування узгодженості;
- регресійне тестування.

Одночасно із запуском даного програмного модуля запускається і локально розташований сервер для обміну та обробки повідомлень. Запуск сервера буде повторюватись кілька разів для того, щоб протестувати його роботу та впевнитись, що він працює справно і при його запуску не виникає ніяких помилок та підтягування усіх потрібних сервісів відбувається без зайвого втручання. Запуск серверу з усіма потрібними параметрами та інформацією про сам стан сервера представлено на рисунку 3.5 у консольному вікні Windows 10 яке появляється при запуску програмного модуля.

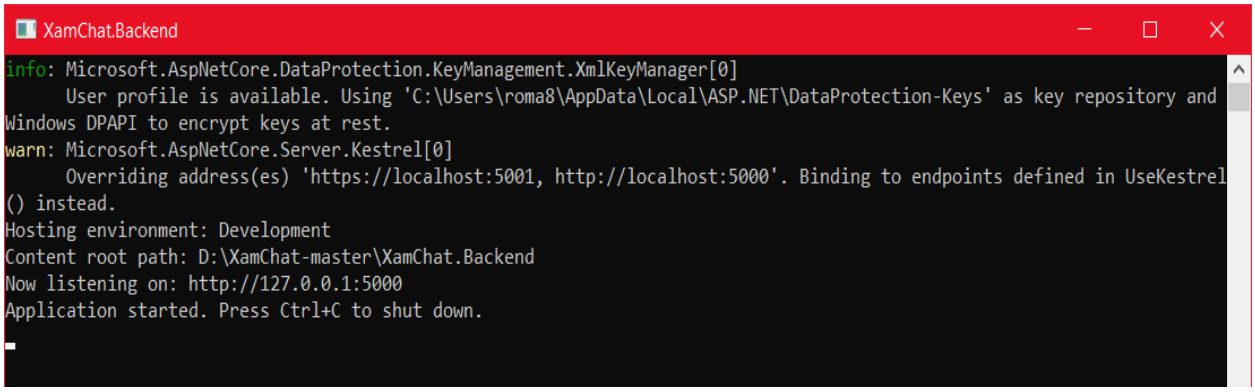


Рисунок 3.5 – Вікно запуску локального сервера

Із десяти запусків при стабільному підключенні до інтернету сервер запускається 10 із 10 разів

Наступним тестом є відклик серверу на запити для цього було обрано програму Postman яка надсилала запити до даного серверу на рисунку 3.6.

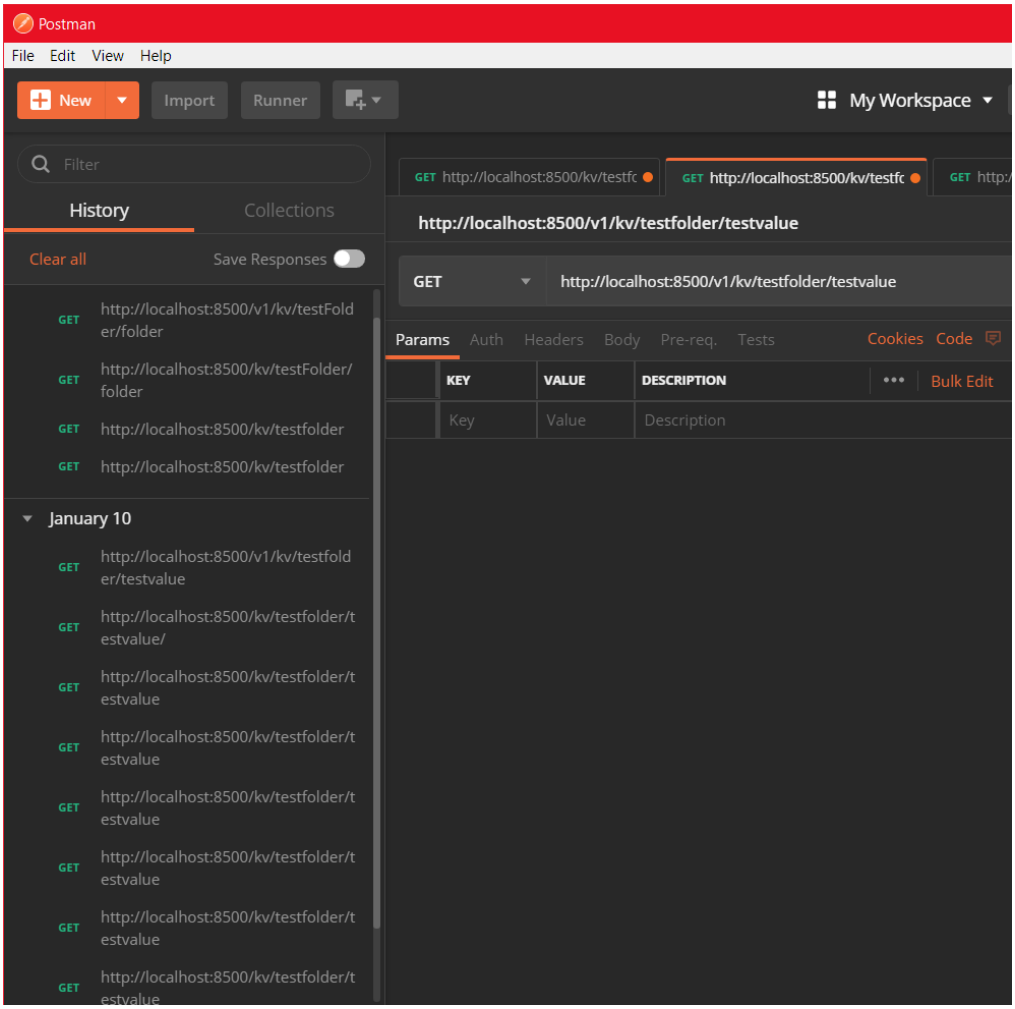


Рисунок 3.6 – Робота програми Postman

На знімку екрану роботи програми для тестування Postman можна побачити чотири успішні запити до сервера, що підтверджує його справну роботу та відклики на запити.

Також потрібно перевірити авторизацію та підключення клієнта до сервера для цього потрібно ввести ім’я користувача та з’єднатися із сервером. Цей процес можна побачити на рисунку 3.7

Рисунок 3.7 – Процес підключення клієнту до серверу.

При вірно введених даних клієнта підключення до сервера є успішним. Доказом цього є речення «You are connected…» та меню вибору існуючих діалогів.

Наступним кроком є перевірка надсилання повідомлень, для цього потрібно відкрити не менше двох клієнтів та приєднати їх до одного з існуючих діалогів які є у меню. Цей тест перевірить чи коректно надсилаються повідомлення та чи відбувається переклад у реальному часі.

Клієнти будуть спілкуватись на різних мовах один на українські повідомлення якого мають перекладатись на англійську та другий на англійські, повідомлення якого в свою чергу перекладатимуться на українську, сам переклад та оригінал повиненя повинен відображатися лише для одержувача а для відправправника лише оригінал його повідомлення. Також навпроти кожного повідомлення має бути зазначений його відправник. Цей тест є важливим етапом адже він перевіряє на роботоспроможність головний функціонал розроблюваного програмного модуля. Детальніше представлено на рисунку 3.8 та рисунку 3.9.

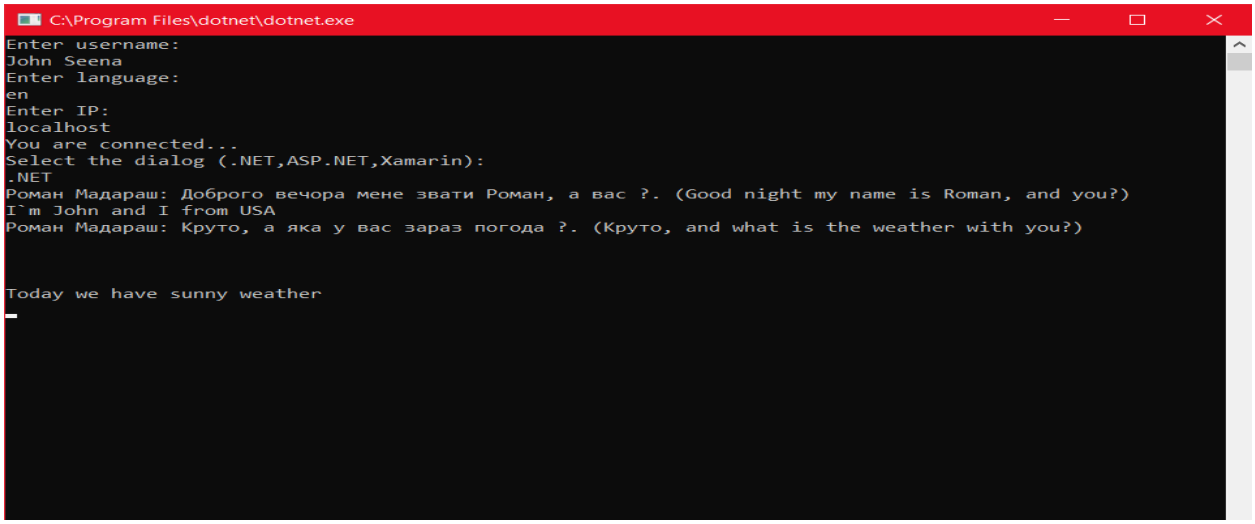


Рисунок 3.8 – Вікно клієнта який спілкується на англійські мові

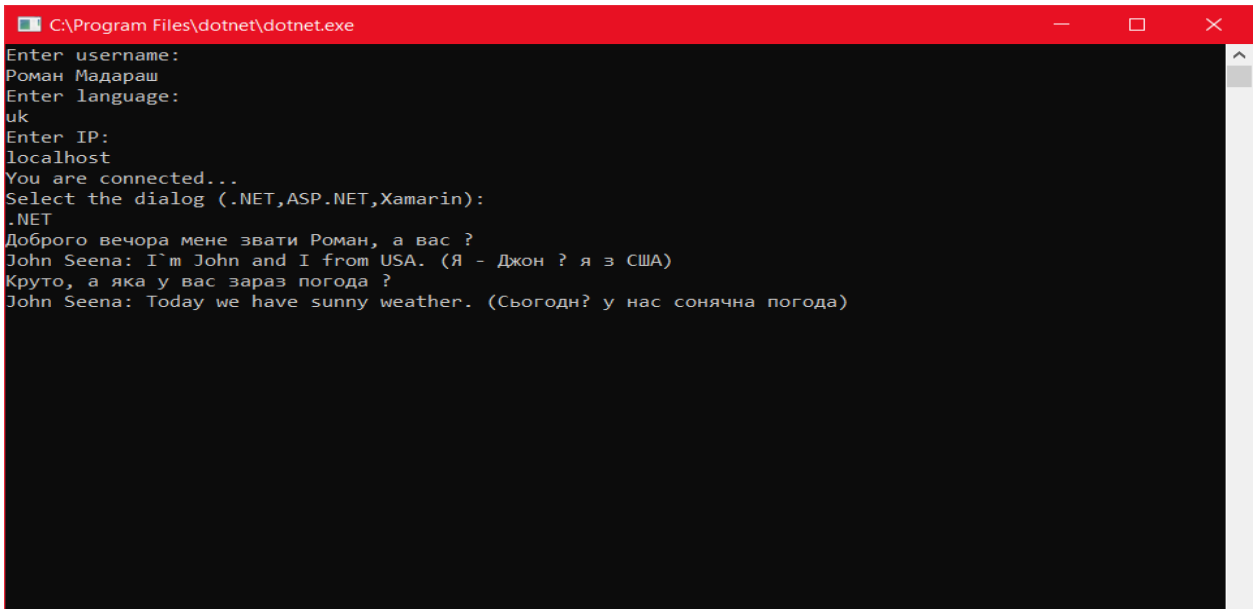


Рисунок 3.9 – Вікно клієнта який спілкується українською мовою мові\

На знімках екрану роботи даного програмного модуля можна побачити, що повідомлення передаються справно при стабільному підключенні до інтернету, а також сам переклад відбувається в обидві сторони та відображається в потрібних місцях. Сам переклад є достатньо точним, щоб зрозуміти, що говорить співбесідник. Тому з цього можна зробити висновки, що функціонал додатку працює так як потрібно та при тестуванні не було виявлено суттєвих відхелнь в роботі даного прогамного модуля.

4 ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ ПРОГРАМНОГО МОДУЛЯ АВТОМАТИЧНОГО ПЕРЕКЛАДУ ПОВІДОМЛЕНЬ В ІНТЕРНЕТ-ЧАТІ

4.1 Економічна характеристика проєкту

Розроблений програмний модуль призначений для забезпечення автоматичного перекладу повідомлень під час спілкування користувачів в інтернет-чаті. Використання такого програмного забезпечення дозволяє усунути мовні бар'єри між співрозмовниками, скоротити час на переклад повідомлень та підвищити ефективність комунікації між користувачами різних країн.

Економічна доцільність розробки полягає у зменшенні витрат часу користувачів на використання сторонніх сервісів перекладу та підвищенні продуктивності спілкування. Крім того, інтеграція перекладача безпосередньо в чат сприяє збільшенню кількості потенційних користувачів програмного продукту.

4.2 Розрахунок витрат на розробку програмного забезпечення

Основними складовими витрат на створення програмного модуля є:

- заробітна плата розробника;
- нарахування на заробітну плату;
- витрати на електроенергію;
- амортизація комп'ютерної техніки;
- накладні витрати.

Заробітна плата розробника

Тривалість розробки становила 2 місяці.

Середньомісячна заробітна плата програміста:

ЗП = 22000 грн.

Основна заробітна плата:

Зосн = 22000 × 2 = 44000 грн.

Додаткова заробітна плата (10%):

Здод = 44000 × 0,1 = 4400 грн.

Загальна заробітна плата:

Ззаг = 44000 + 4400 = 48400 грн.

Відрахування на соціальні заходи

ЄСВ становить 22%:

ЄСВ = 48400 × 0,22 = 10648 грн.

Витрати на електроенергію

Потужність комп'ютера:

P = 0,35 кВт.

Тривалість роботи:

t = 320 годин.

Тариф:

C = 4,32 грн/кВт·год.

Вартість електроенергії:

Ве = 0,35 × 320 × 4,32

Ве = 484 грн.

Амортизація обладнання

Вартість комп'ютера:

Вк = 45000 грн.

Термін експлуатації:

T = 5 років.

Річна амортизація:

Ар = 45000 / 5 = 9000 грн.

На період розробки (2 місяці):

Ам = 9000 / 12 × 2

Ам = 1500 грн.

Накладні витрати

Приймаємо накладні витрати у розмірі 20% від основної заробітної плати:

Нв = 44000 × 0,2

Нв = 8800 грн.

4.3 Загальна собівартість розробки

Таблиця 4.1 – Розрахунок загальної собівартості розробки програмного модуля

Стаття витрат	**Сума, грн**
Основна заробітна плата	44000
Додаткова заробітна плата	4400
ЄСВ	10648
Електроенергія	484
Амортизація	1500
Накладні витрати	8800
Разом	**69832**

Таким чином, загальна собівартість розробки програмного модуля автоматичного перекладу повідомлень становить **69 832 грн**.

4.4 Оцінка економічної ефективності

Використання програмного модуля дозволяє скоротити час на переклад повідомлень приблизно на 30–40 % у порівнянні з використанням сторонніх сервісів перекладу.

Припустимо, що програмний продукт використовують 100 користувачів, кожен з яких економить у середньому 15 хвилин на день.

Загальна економія часу:

100 × 15 хв = 1500 хв = 25 годин на день.

За середньої вартості робочої години 120 грн:

Ед = 25 × 120 = 3000 грн/день.

Річний економічний ефект:

Ер = 3000 × 250 = 750000 грн.

Коефіцієнт економічної ефективності:

Ке = 750000 / 69832 = 10,74

Термін окупності:

Ток = 69832 / 750000 = 0,093 року

або приблизно **1,1 місяця**.

5 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

Безпека життєдіяльності в галузі ІТ - це сукупність заходів, технологій і правил, спрямованих на забезпечення безпечного функціонування людини в інформаційному середовищі. Вона охоплює як захист фізичного здоров’я користувачів при роботі з комп’ютерною технікою (ергономіка, електробезпека, санітарно-гігієнічні норми), так і інформаційну безпеку - захист даних, програмного забезпечення та систем від вірусів, кібератак, збоїв і несанкціонованого доступу. Це важлива складова стабільної та ефективної роботи в цифровому світі.

В свою чергу охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров’я і працездатності людини у процесі трудової діяльності [5].

Охорона праці є важливою складовою будь-якого виробництва, відзначаючи людину, як головну цінність, адже її безпека і хороше здоров'я дозволяють зробити виробничий процес більш чітким, що підвищить рентабельність самого підприємства. Людське життя не повинно бути розмінною монетою заради гарної заробітної плати, або особливо цінного продукту, який виробляє підприємство. Ніщо не повинно бути понад забезпечення захисту людини від загроз його здоров'ю і життю. Правильно організована система охорони праці дисциплінує самого працівника і, як наслідок, веде до підвищення продуктивності виконуваної роботи і збільшення її ефективності.

5.1 Розрахунок системи штучного освітлення для приміщення, де здійснюється розробка програмного модуля автоматичного перекладу повідомлень в інтернет-чаті

Під час виконання робіт із розробки програмного забезпечення користувачі персональних комп'ютерів проводять значну частину робочого

часу за моніторами. Одним із важливих факторів, що впливають на працездатність та безпеку праці, є забезпечення належного рівня освітленості робочих місць. Недостатнє або надмірне освітлення може спричинити швидку втому органів зору, головний біль, зниження концентрації уваги та продуктивності праці.

Для приміщень, у яких здійснюється робота з комп'ютерною технікою, відповідно до вимог ДБН В.2.5-28:2018 «Природне і штучне освітлення», рекомендована освітленість робочої поверхні повинна становити **300–500 лк**. Для проведення розрахунків приймаємо нормативне значення освітленості **E = 300 лк**.

Вихідні дані для розрахунку:

- довжина приміщення – 6 м;
- ширина приміщення – 5 м;
- висота приміщення – 3 м;
- кількість робочих місць – 4;
- нормована освітленість – 300 лк;
- коефіцієнт запасу – 1,4;
- коефіцієнт використання світлового потоку – 0,6;
- тип світильників – світлодіодні LED-панелі;
- світловий потік одного світильника – 3600 лм.

Площа приміщення:

$S = a \cdot b$, м²

a – довжина приміщення, м;

$S = 6 \cdot 5 = 30$ м².

Загальний необхідний світловий потік визначається за формулою:

$F = (E \cdot S \cdot Kз) / \eta$, лм

де:

E – нормована освітленість, лк;

Підставляючи значення, отримаємо:

$F = (300 \cdot 30 \cdot 1{,}4) / 0{,}6 = 21000$ лм

Необхідна кількість світильників:

$N = F / F_1$

$N = 21000 / 3600 = 5{,}83$

Отримане значення округлюємо у більшу сторону:

N = 6 світильників.

$F = (E \cdot S \cdot Kз) / \eta$, лм

Необхідна кількість світильників:

$F = (E \cdot S \cdot Kз) / \eta$, лм

$N = F / F_1$

$F = (E \cdot S \cdot Kз) / \eta$, лм

Отримане значення округлюється у більшу сторону, тому для забезпечення нормативної освітленості необхідно встановити **6 світлодіодних світильників**.

Таким чином, запропонована система штучного освітлення забезпечує комфортні умови праці для працівників, які здійснюють розробку програмного модуля автоматичного перекладу повідомлень в інтернет-чаті.

План-схема розміщення світильників

Світильники рекомендується розташувати рівномірно по площі приміщення у два ряди по три світильники в кожному ряду.

На рисунку 5.1 наведено план-схема розміщення світильників у приміщенні

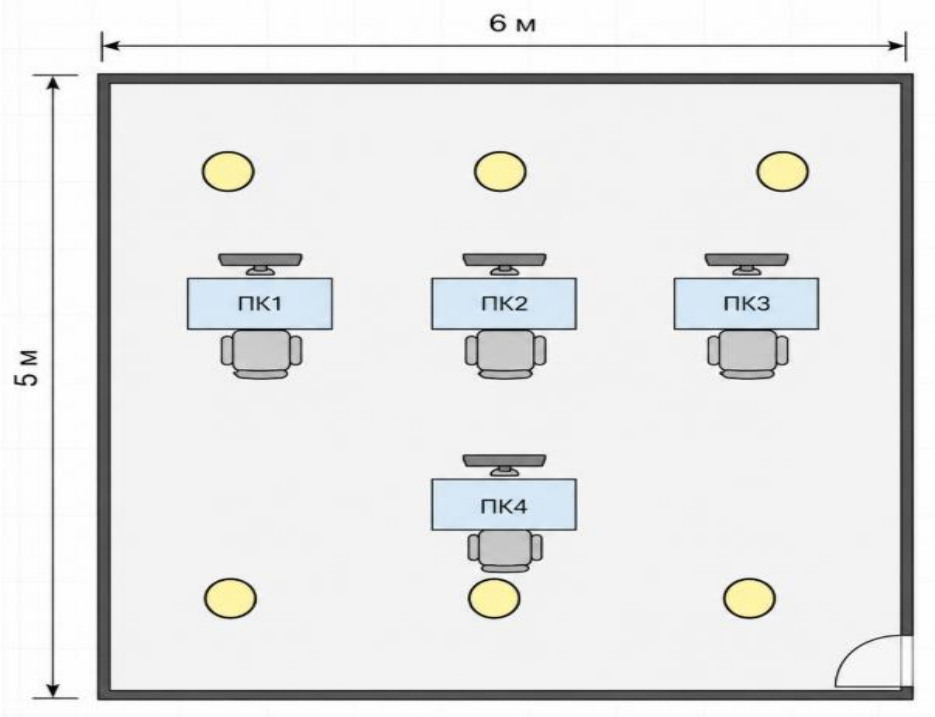


Рисунок 5.1 – План-схема розміщення світильників у приміщенні

На рисунку 5.1 наведено план-схему приміщення розміром 6 × 5 м, у якому здійснюється розробка програмного модуля автоматичного перекладу повідомлень в інтернет-чаті. Відповідно до результатів розрахунку системи штучного освітлення, для забезпечення нормативного рівня освітленості 300 лк передбачено встановлення шести світлодіодних світильників, рівномірно розташованих у два ряди по три світильники. Робочі місця користувачів ПК організовані таким чином, щоб забезпечити комфортні умови праці, зручність пересування приміщенням та відповідність ергономічним вимогам щодо експлуатації комп'ютерної техніки.

5.2 Організація раціонального режиму праці та відпочинку користувачів ПК

Під час виконання робіт з розробки програмного забезпечення працівники значну частину робочого часу проводять за персональними комп'ютерами. Такий характер роботи супроводжується підвищеним

навантаженням на органи зору, статичним навантаженням на опорно-руховий апарат, нервово-емоційним напруженням та зниженням рухової активності. У зв'язку з цим важливим завданням охорони праці є організація раціонального режиму праці та відпочинку.

Раціональний режим праці та відпочинку передбачає чергування періодів роботи з періодами відновлення працездатності працівника. Під час роботи за комп'ютером доцільно періодично змінювати вид діяльності, виконувати короткочасні фізичні вправи, а також вправи для очей, спрямовані на зменшення зорової втоми та покращення кровообігу.

Для профілактики перевтоми рекомендується:

– періодично змінювати робочу позу;

– виконувати вправи для м'язів шиї, плечового поясу та спини;

– здійснювати фокусування погляду на предметах, розташованих на різній відстані;

– забезпечувати достатній рівень фізичної активності протягом робочого дня;

– підтримувати оптимальний мікроклімат та провітрювати приміщення.

Особливу увагу необхідно приділяти ергономічній організації робочого місця. Робочий стіл повинен забезпечувати достатній простір для розміщення обладнання та документів. Конструкція робочого крісла має забезпечувати підтримку фізіологічно правильного положення тіла та можливість регулювання висоти сидіння і нахилу спинки.

Під час роботи за персональним комп'ютером рекомендується дотримуватися таких ергономічних вимог:

– відстань від очей користувача до екрана монітора повинна становити 50–70 см;

– верхня межа екрана має розташовуватися на рівні очей або дещо нижче;

– клавіатура повинна знаходитися на зручній відстані від працівника для забезпечення природного положення кистей рук;

– кут згину рук у ліктьових суглобах повинен наближатися до 90°;

– ступні повинні повністю спиратися на підлогу або спеціальну підставку;

– необхідно забезпечити достатній простір для ніг під робочим столом.

Важливим чинником збереження працездатності є підтримання сприятливого психоемоційного стану працівників. Для цього доцільно забезпечувати раціональне планування робочого часу, уникати надмірного інформаційного навантаження та створювати комфортні умови праці.

Дотримання вимог щодо організації режиму праці та відпочинку, а також ергономічного облаштування робочого місця сприяє зниженню професійних ризиків, попередженню захворювань органів зору та опорно-рухового апарату, підвищенню продуктивності праці та ефективності роботи розробників програмного забезпечення.

ВИСНОВКИ

У результаті виконання дипломної роботи було проведено аналіз сучасних інтернет-чатів та месенджерів, досліджено їхні функціональні можливості, переваги та недоліки. Особливу увагу приділено питанням подолання мовного бар’єру під час спілкування користувачів у мережі Інтернет. На основі проведеного аналізу встановлено актуальність розробки програмного модуля автоматичного перекладу повідомлень в інтернет-чаті.

У процесі виконання роботи було спроєктовано архітектуру програмної системи, розроблено структуру бази даних та реалізовано програмний модуль із використанням сучасних технологій програмування. Створене програмне забезпечення забезпечує автоматичний переклад повідомлень у режимі реального часу, зберігання історії повідомлень, обмін даними між користувачами та зручний інтерфейс взаємодії.

У роботі виконано програмну реалізацію основних функціональних можливостей системи, проведено тестування розробленого програмного продукту та підтверджено його працездатність. Результати тестування показали коректність роботи основних модулів системи та відповідність поставленим функціональним вимогам.

Також було проведено економічне обґрунтування проєкту, яке підтвердило доцільність розробки програмного забезпечення. У розділі з охорони праці виконано розрахунок системи штучного освітлення робочого місця розробника та розглянуто питання організації раціонального режиму праці й відпочинку користувачів персональних комп’ютерів.

Розроблений програмний модуль може бути використаний як основа для створення сучасних систем онлайн-спілкування з підтримкою автоматичного перекладу повідомлень. Подальший розвиток проєкту може

передбачати розширення переліку підтримуваних мов, впровадження технологій штучного інтелекту для підвищення якості перекладу, реалізацію голосового перекладу та інтеграцію з популярними месенджерами й соціальними мережами.

Таким чином, усі поставлені завдання дипломної роботи виконані в повному обсязі, а мета роботи — розробка програмного модуля автоматичного перекладу повідомлень в інтернет-чаті — досягнута.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Меседжери [Електронний ресурс] – Режим доступу:https: //ua.gecid.com/news/obzor_messendzherov_top-5_programm/
2. [Електронний ресурс] – Режим доступу: https://pingvin.pro/gadgets/article-gadget/najpopulyarnishi-mesendzhery-sered-ukrayintsiv.html
3. Язык программирования C# 6.0 и платформа .NET 4.6./Ендрю Троелсен, Філіпп Джепікс — Вильямс 2016. — 1440 с.
4. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Xamarin Xamarin
5. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/.NET_Framework - .NET Framework
6. [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/xamarin/11.1.php
7. Martin Fowler UML Distilled: A Brief Guide to the Standard Object Modeling Language/ Martin Fowler – Addison-Wesley Professional – 2003. – 208с.
8. NET [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/aspnet5/1.1.php.
9. Лешек А. Мацяшек Анализ требований и проектирование систем.
10. [Електронний ресурс] – https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/inside-a-program/index
11. .NET [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/ruru/dotnet/api/system.net.http.httpclient?view=netframework-4.7.2.

11 [Електронний ресурс] – https://uk.wikipedia.org/wiki/.NET_Framework.

12. [Електронний ресурс] – Режим доступу: https://play.google.com/store/apps/details?id=com.masterkeygames.sentencemasterpro&hl=uk – sentencemasterpro.

13. [Електронний ресурс] – Режим доступу:https://uk.wikipedia.org/wiki/Telegram.

14. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/C_Sharp C#

15. [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/net/3.1.php

16. [Електронний ресурс] – https://uk.wikipedia.org/wiki/.NET_Framework.

17. Фаулер Скотт До. UML в короткому викладі. Застосування стандартної мови об'єктного моделювання: Пер. з англ. – М.:Мир, 1999. – 191 с.

18. [Електронний ресурс] – https://docs.microsoft.com/en-us/aspnet/core/signalr/configuration?view=aspnetcore-2.2&tabs=dotnet.

19. [Електронний ресурс] – https://docs.microsoft.com/ruru/aspnet/core/?view=aspnetcore-2.2

20. [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/framework/wpf/advanced/xaml-overview-wpf.

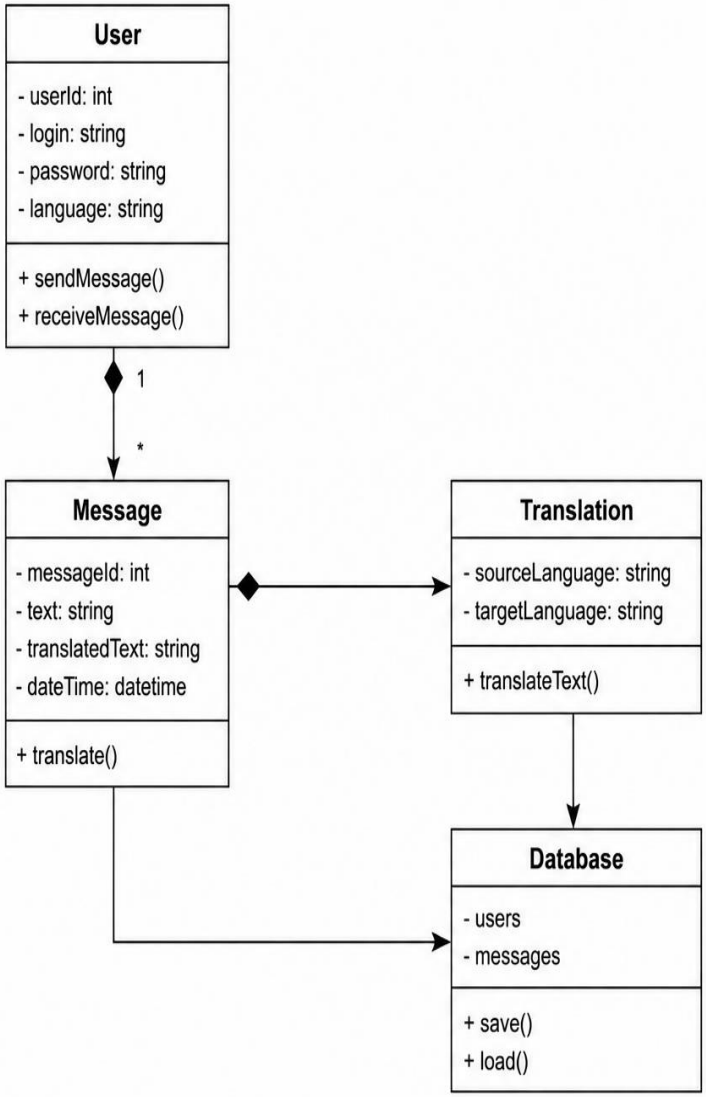
21. [Електронний ресурс] – Режим доступу: http://wiki.rosalab.ru/ru/index.php/%D0%92%D0%B8%D0%B4%D1%8B_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%9F%D0%9E.

22. Методичні вказівки до виконання дипломного проекту освітнього ступеня «бакалавр» напряму підготовки 6.050101 «Комп’ютерні науки» / Укл. А.О. Саченко, М.П. Комар, Н.М. Васильків, Г.М. Гладій, В.С. Коваль, І.П. Струбицька. – Тернопіль: ТНЕУ, 2017. – 56 с.

ДОДАТКИ

Додаток А. Загальна структура роботи програми

Додаток Б. Діаграма класів



Додаток В. Лістинг програмного коду

Код перекладача

```
public class Translator : ViewModelBase
   {
      readonly ITranslator _translator;

      #region Properties

      private IEnumerable<string> _languages;
      public IEnumerable<string> Languages
      {
         get { return _languages; }
         set { SetProperty(ref _languages, value); }
      }

      private string _value;
      public string Value
      {
         get { return _value; }
         set { SetProperty(ref _value, value); }
      }

      private string _translation;
      public string Translation
      {
         get { return _translation; }
         set { SetProperty(ref _translation, value); }
      }

      private string _languageToString;
      public string LanguageToString
      {
         get { return _languageToString; }
         set { SetProperty(ref _languageToString, value); }
      }

      private string _languageFromString;
      public string LanguageFromString
      {
         get { return _languageFromString; }
         set { SetProperty(ref _languageFromString, value); }
      }

      #endregion
      public ICommand SaveCommand { get; set; }

      public ICommand TranslateCommand { get; set; }

      public ICommand TextToSpeechCommand { get; set; }
```

```
        public ICommand ResultTextToSpeechCommand { get; set; }

        public WordPageDetailViewModel(INavigationService navigationService,
ITranslator translator) : base(navigationService)
        {
            _translator = translator;

            Languages = new[] { Constants.Ukrainian, Constants.English };

            TextToSpeechCommand = new DelegateCommand(OnTextToSpeech);
            ResultTextToSpeechCommand = new
DelegateCommand(OnResultTextToSpeech);

            TranslateCommand = new DelegateCommand(async () =>
            {
                var toLanguage = LanguageToString.Substring(0, 2).ToLower();
                var fromLanguage = LanguageFromString.Substring(0, 2).ToLower();

                var result = await _translator.TranslateAsync(fromLanguage, toLanguage,
Value);

                Translation = result;
            });

            SaveCommand = new DelegateCommand(async () => await SaveItemAsync());
        }

        private void OnResultTextToSpeech()
        {
            DependencyService.Get<ITextToSpeech>().Speak(Translation,
LanguageToString.StartsWith("Uk") ? "uk-UA" : "en-US");
        }

        private void OnTextToSpeech()
        {
            DependencyService.Get<ITextToSpeech>().Speak(Value,
LanguageFromString.StartsWith("Uk") ? "uk-UA": "en-US");
        }

        public override void OnNavigatedFrom(INavigationParameters parameters)
        {
            base.OnNavigatedFrom(parameters);

            _id = 0;
            Value = Translation = LanguageFromString = LanguageToString = string.Empty;
        }

        public override void OnNavigatedTo(INavigationParameters parameters)
        {
            base.OnNavigatedTo(parameters);
```

```
            if (parameters.TryGetValue("Word", out WordItem word))
            {

                _id = word.Id;
                Value = word.Value;
                Translation = word.Translation;
                LanguageFromString = word.LanguageFrom;
                LanguageToString = word.LanguageTo;
            }
        }

        private async Task SaveItemAsync()
        {
            var item = new WordItem()
            {
                Value = Value,
                Translation = Translation,
                LanguageFrom = LanguageFromString,
                LanguageTo = LanguageToString,
                Description = $"Source: {LanguageFromString}, target: {LanguageToString}"
            };

            var navParameters = new NavigationParameters();

            if (_id != 0)
            {
                item.Id = _id;
                navParameters.Add("EditWordItem", item);
            }
            else
            {
                item.Id = WordItem.NextId++;
                navParameters.Add("AddWordItem", item);
            }

            await NavigationService.GoBackAsync(navParameters);
        }

        private long _id;
    }
```

Код надсилання повідомлень

```
public class ChatViewModel : ViewModelBase
    {
        public ChatMessage ChatMessage { get; }

        public ObservableCollection<ChatMessage> Messages { get; }
        public ObservableCollection<User> Users { get; }

        bool isConnected;
        public bool IsConnected
        {
```

```
        get => isConnected;
        set
        {
            Device.BeginInvokeOnMainThread(() =>
            {
                SetProperty(ref isConnected, value);
            });
        }
    }


    public Command SendMessageCommand { get; }
    public Command ConnectCommand { get; }
    public Command DisconnectCommand { get; }

    Random random;
    public ChatViewModel()
    {
        if (DesignMode.IsDesignModeEnabled)
            return;

        Title = Settings.Group;

        ChatMessage = new ChatMessage();
        Messages = new ObservableCollection<ChatMessage>();
        Users = new ObservableCollection<User>();
        SendMessageCommand = new Command(async () => await SendMessage());
        ConnectCommand = new Command(async () => await Connect());
        DisconnectCommand = new Command(async () => await Disconnect());
        random = new Random();

        ChatService.Init(Settings.ServerIP, Settings.UseHttps);

        ChatService.OnReceivedMessage += (sender, args) =>
        {
            SendLocalMessage(args.Message, args.User);
            AddRemoveUser(args.User, true);
        };

        ChatService.OnEnteredOrExited += (sender, args) =>
        {
            AddRemoveUser(args.User, args.Message.Contains("entered"));
        };

        ChatService.OnConnectionClosed += (sender, args) =>
        {
            SendLocalMessage(args.Message, args.User);
        };
    }


    async Task Connect()
```

```
    {
        if (IsConnected)
            return;
        try
        {
            IsBusy = true;
            await ChatService.ConnectAsync();
            await ChatService.JoinChannelAsync(Settings.Group, Settings.UserName);
            IsConnected = true;

            AddRemoveUser(Settings.UserName, true);
            await Task.Delay(500);
            SendLocalMessage("Connected...", Settings.UserName);
        }
        catch (Exception ex)
        {
            SendLocalMessage($"Connection error: {ex.Message}", Settings.UserName);
        }
        finally
        {
            IsBusy = false;
        }
    }

    async Task Disconnect()
    {
        if (!IsConnected)
            return;
        await ChatService.LeaveChannelAsync(Settings.Group, Settings.UserName);
        await ChatService.DisconnectAsync();
        IsConnected = false;
        SendLocalMessage("Disconnected...", Settings.UserName);
    }

    async Task SendMessage()
    {
        if(!IsConnected)
        {
            await DialogService.DisplayAlert("Not connected", "Please connect to the server and
try again.", "OK");
            return;
        }
        try
        {
            IsBusy = true;
            await ChatService.SendMessageAsync(Settings.Group,
                Settings.UserName,
                ChatMessage.Message);

            ChatMessage.Message = string.Empty;
        }
        catch (Exception ex)
```

```
        {
            SendLocalMessage($"Send failed: {ex.Message}", Settings.UserName);
        }
        finally
        {
            IsBusy = false;
        }
    }

    private void SendLocalMessage(string message, string user)
    {
        Device.BeginInvokeOnMainThread(() =>
        {
            var first = Users.FirstOrDefault(u => u.Name == user);

            Messages.Insert(0, new ChatMessage
            {
                Message = message,
                User = user,
                Color = first?.Color ?? Color.FromRgba(0, 0, 0, 0)
            });
        });
    }

    void AddRemoveUser(string name, bool add)
    {
        if (string.IsNullOrWhiteSpace(name))
            return;
        if (add)
        {
            if (!Users.Any(u => u.Name == name))
            {
                var color = Messages.FirstOrDefault(m => m.User == name)?.Color ??
Color.FromRgba(0, 0, 0, 0);
                Device.BeginInvokeOnMainThread(() =>
                    {
                        Users.Add(new User { Name = name, Color = color });
                    });
            }
        }
        else
        {
            var user = Users.FirstOrDefault(u => u.Name == name);
            if (user != null)
            {
                Device.BeginInvokeOnMainThread(() =>
                {
                    Users.Remove(user);
                });
            }
        }
    }
```

```
    }
        }
```

Код сторінки авторизації

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:controls="clr-namespace:XLabs.Forms.Controls;assembly=XLabs.Forms"
             xmlns:local="clr-namespace:MobileClient.CustomControls"
             x:Class="MobileClient.Views.LoginPage">
    <ContentPage.Content>
        <StackLayout BackgroundColor="FloralWhite">
            <Frame Margin="20, 10, 20, 10" WidthRequest="400"
VerticalOptions="CenterAndExpand" HorizontalOptions="CenterAndExpand"
CornerRadius="10" BackgroundColor="#967BB6">
                <StackLayout>

                    <Image Source="GB_flag.png" HorizontalOptions="CenterAndExpand"></Image>
                    <Label Margin="0, 0, 0, 30" Text="Login page" TextColor="White"
HorizontalOptions="CenterAndExpand" FontSize="Medium"></Label>

                    <Frame CornerRadius="7" Margin="5, 0, 5, 5">
                        <local:CustomEntry Placeholder="Username" PlaceholderColor="Gray"
Text="{Binding Login}"></local:CustomEntry>
                    </Frame>
                    <Frame CornerRadius="7" Margin="5, 0, 5, 0">
                        <StackLayout Orientation="Horizontal">
                            <local:CustomEntry Placeholder="Password"
HorizontalOptions="FillAndExpand" x:Name="password" PlaceholderColor="Gray"
Text="{Binding Password}" IsPassword="true"></local:CustomEntry>
                            <Image Source="hide_pass.png" x:Name="hidePassImage">
                                <Image.GestureRecognizers>
                                    <TapGestureRecognizer Tapped="OnShowPass"/>
                                </Image.GestureRecognizers>
                            </Image>
                        </StackLayout>
                    </Frame>
                    <StackLayout Margin="5, 20, 5, 0" Orientation="Horizontal">
                        <Label Margin="5, 0, 0, 0" Text="Remember me:" FontSize="Medium"
TextColor="White"></Label>
                        <controls:CheckBox></controls:CheckBox>
                    </StackLayout>
                    <controls:ExtendedButton HeightRequest="70" Margin="5, 5, 5, 0" Text="Submit"
FontSize="Medium" TextColor="White" BackgroundColor="#6666ff" CornerRadius="10"
Command="{Binding LoginCommand}"></controls:ExtendedButton>
                </StackLayout>
            </Frame>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```