Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Система оповіщення для дистанційного навчання

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Пацула В.В,
(підпис) (прізвище та ініціали)

Керівник _____ Готович В.А.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Пастух О.А.
(підпис) (прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Пацулі Віктору Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Система оповіщення для дистанційного навчання

Керівник роботи Готович Володимир Анатолійович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 02 » березня 2021 року № 4/7-171

2. Термін подання студентом завершеної роботи 22 червня 2021р.

3. Вихідні дані до роботи Інтернет джерела, технічна документація

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. 1.1 Огляд платформи для дистанційного навчання. 1.1.1 Вступ. 1.1.2 Реєстрація. 1.1.3 Огляд функцій пошти дистанційного навчання. 1.2 Постановка проблеми. 1.3 Огляд месенджера «Телеграм». 1.3.1 Вступ. 1.3.2 Реєстрація. 1.3.3 Чати. 1.3.4 Боти. 1.4 Висновок до першого розділу. 2 РОЗРОБКА СИСТЕМИ ОПОВІЩЕННЯ. 2.1 Огляд інструментів та технологій розробки. 2.1.1 .NET. 2.1.2 Visual Studio Community 2019. 2.1.3 Selenium. 2.1.4 AngleSharp. 2.2 Розробка системи оповіщення. 2.2.1 Загальна архітектура. 2.2.2 Архітектура сповіщень. 2.2.3 Розробка модуля «Crawler». 2.2.4 Розробка модуля «TelegramWrapper». 2.3 Огляд функцій системи оповіщення. 2.4 Висновок до другого розділу. 3. Безпека життєдіяльності, основи хорони праці. 3.1 Стихійні лиха та їх класифікація. 3.2 Розрахунок місцевого освітлення для роботи з комп'ютером. 3.3 Санітарно-гігієнічні вимоги до умов праці. 3.4 Висновок до третього розділу. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Актуальність теми роботи. Мета і задачі дослідження. Платформа для дистанційного навчання. Пошта дистанційного навчання. Мессенджер "Телеграм". Чат боти. Реєстрація чат бота. Загальна архітектура. Архітектура сповіщень. Обробка команд "Телеграм". Додавання нового Встановлення таймера. Отримання даних від браузера. Перетворення даних у модель. Доступ до бота. Огляд доступних команд. Вхід на дистанційне навчання. Перевірка повідомлень. Отримання тіла повідомлення. Відповідь. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Гурик О.Я., доцент кафедри МТ		

7. Дата видачі завдання 25 січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	25.01.2021	*Виконано*
2.	Підбір джерел про розробку програмного забезпечення	26.01.2021 -	*Виконано*
	На мові програмування C#	10.02.2021	
3.	Переклад та опрацювання джерел про роботу з API	11.02.2021 -	*Виконано*
	Телеграм	12.02.2021	
4.	Виконання дослідження щодо можливостей бібліотеки	13.02.2021 -	*Виконано*
	Selenium	14.02.2021	
5	Розроблення системи оповіщення для дистанційного	15.02.2021 -	*Виконано*
	навчання	09.03.2021	
5.	Оформлення розділу «Аналіз предметної області»	10.03.2021 -	*Виконано*
		14.04.2021	
6.	Оформлення розділу «Розробка системи сповіщення»	15.04.2021 -	*Виконано*
		15.05.2021	
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	16.05.2021 - 17.05.2021	*Виконано*
8.	Виконання завдання до підрозділу «Основи хорони	18.05.2021 -	*Виконано*
	праці»	09.06.2021	
9.	Оформлення кваліфікаційної роботи	10.06.2021	*Виконано*
10.	Нормоконтроль	12.06.2021	*Виконано*
11.	Перевірка на плагіат	12.06.2021	*Виконано*
12.	Попередній захист кваліфікаційної роботи	21.06.2021	*Виконано*
13.	Захист кваліфікаційної роботи	22.06.2021	

Студент _____ Пацула В.В,
(підпис) (прізвище та ініціали)

Керівник роботи _____ Готович В.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Система оповіщення для дистанційного навчання // Кваліфікаційна робота освітнього рівня «Бакалавр» // Пацула Віктор Васильович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2021 // С. 48, рис. – 21, табл. – 1, додат. – 1, бібліогр. – 33.

Ключові слова: система оповіщення, чат бот, Телеграм, дистанційне навчання.

Кваліфікаційна робота присвячена розширенню функціональних можливостей платформи дистанційного навчання ТНТУ ім.І.Пулюя.

Мета роботи: вирішити проблему несвоєчасного отримання сповіщень на платформі дистанційного навчання.

В першому розділі кваліфікаційної роботи розглянуто можливості дистанційного навчання в ТНТУ та функції месенджера «Телеграм». Проведено аналіз недоліків дистанційного навчання. Запропоновано вирішення проблем та огляд технологій що зможуть вирішити проблеми ефективно.

В другому розділі кваліфікаційної роботи розглянуто технології для побудови системи оповіщення та її проектування. Розглянуто архітектуру системи оповіщення, взаємодію її модулів. Також розглянуто самі модулі, їх принцип роботи.

Розглянуто функції системи оповіщення із сторони користувача у месенджері «Телеграм». Показано як почати роботу з чат ботом та його використання у подальшому. Наведено переваги роботи з чат ботом.

ANNOTATION

Notification system development for distance learning // Qualification work of the educational level «Bachelor» // Patsula Viktor Vasylovych // Ternopil National Technical University named after Ivan Pulyuy, Faculty of Computer Information Systems and Software Engineering // Ternopil, 2021 // Explanatory note size – 48 pages, contains 21 illustrations, 1 table, 1 appendix, 33 bibliography items.

Keywords: notification system, chat bot, telegram, distance learning

Qualification work is devoted to expanding the functionality of the distance learning platform of TNTU named after I. Pulyuy.

Purpose: to solve the problem of late receipt of notifications on the distance learning platform.

In the first section of the qualification work the possibilities of distance learning in TNTU and the functions of the messenger "Telegram" are considered. The analysis of shortcomings of distance learning is carried out. Problem solving and review of technologies that can solve problems effectively are offered.

In the second section of the qualification work the technologies for construction of the alarm system and its design are considered. The architecture of the alarm system, the interaction of its modules are considered. The modules themselves, their principle of operation are also considered.

The functions of the notification system on the part of the user in the "Telegram" messenger are considered. It shows how to start working with a chat bot and its use in the future. The advantages of working with a chat bot are given.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ТНТУ – Тернопільський державний університет [1].

ЦДН – Центр дистанційного навчання [1].

Visual Studio – Visual Studio Community 2019.

Система оповіщення – Система оповіщення для дистанційного навчання.

Пошта – Пошта у кабінеті користувача дистанційного навчання [1].

.NET – .NET Framework фреймворк від компанії Microsoft [9].

Токен – Стрічка закодованих символів що використовується для підключення до API Телеграм [6].

Selenium – Selenium Web Driver.

ЗМІСТ

ВСТУП

Актуальність теми. ТНТУ надає можливість студентам навчатися дистанційно, що зручно в умовах карантину. Дана платформа має усе необхідне для навчання. Матеріали з дисциплін згруповані в курси. На курсах є можливість читати лекції із заданих дисциплін, проходити тестування для перевірки рівня знань студентів та місце для збереження студентських робіт. Курси адмініструють інструктори – викладачі дисциплін.

Платформа дистанційного навчання має пошту для спілкування студентів та викладачів у межах університету. Доступ до пошти проводиться із облікового запису користувача на платформі дистанційного навчання. Для онлайн спілкування важливо вчасно отримувати повідомлення для швидкої відповіді. Це сприяє пришвидшенню навчання студентів, адже таким чином можна швидко отримати та обробити інформацію.

Не менш важливою складовою онлайн спілкування є зручність доступу до повідомлень. Прикладом швидкого доступу до спілкування є месенджери. Вони доступні на багатьох платформах та мають зручний функціонал для обміну повідомленнями що дозволяє вчасно отримувати та відповідати на повідомлення.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є:

– Проаналізувати варіанти вирішення проблеми несвоєчасного отримання повідомлень на платформі дистанційного навчання;

– Запропонувати можливі варіанти вирішення проблеми несвоєчасного отримання повідомлень;

– Спроектувати систему сповіщення що буде швидкою та зручною для користування;

– Проаналізувати технології для вирішення проблеми затримки комунікації;

– Реалізувати систему оповіщення для дистанційного навчання;
– Протестувати застосунок на функціональному рівні;
– Висвітлити питання щодо безпеки та охорони праці.

Практичне значення одержаних результатів. Для студентів ТНТУ буде надана можливість скористатися системою оповіщення для дистанційного навчання, що надасть їм можливості зручно мати доступ до пошти платформи та відповідати на повідомлення. Зручність виявляється у доступності платформи та її використанню. Для зручності буде використана програма месенджер що є доступною на всіх основних мобільних та настільних платформах. Це надасть можливість вчасно отримувати сповіщення та швидко на них відповідати.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд платформи для дистанційного навчання

1.1.1 Вступ

ТНТУ надає студентам можливість навчання онлайн через платформу для дистанційного навчання. Керування платформою здійснюється через ЦДН, головне завдання якого полягає у забезпеченні онлайн навчання [1].

Дана платформа це важлива складова навчання для студентів та викладачів що надає можливість студентам навчатися та засвоювати новий матеріал. Навчання відбувається у форматі віддаленого спілкування викладачів та студентів за допомогою платформи дистанційного навчання [1].

У платформі дистанційного навчання містяться курси що пройшли атестацію у комісії експертів [1]. Кожен курс містить усі необхідні матеріали для визначеної дисципліни, а також електронні тести для закріплення знань студенців. Окрім цього, курси надають можливість для перевірки студентських робіт викладачами та проведення занять в онлайн режимі. Також кожен курс має інструктора що здійснює керування та наповнення курсу відповідної дисципліни [2].

Дистанційне навчання надає наступні можливості для студентів [2]:

– Доступ до лекційного матеріалу.
– Лекції та практичні заняття онлайн.
– Місце для збереження лаболаторних робіт.
– Microsoft Office 365 online.
– Електронна залікова книга.
– Доступ до графіку навчання.
– Електронна пошта.
– Розклад занять студентів та викладачів.

– Здача тестів.

1.1.2 Реєстрація

Для реєстрації на платформі дистанційного навчання слід натиснути на посилання «Реєстрація». Головна сторінка курсу знаходиться за веб адресою - https://dl.tntu.edu.ua/ (рис. 1.1) [3].

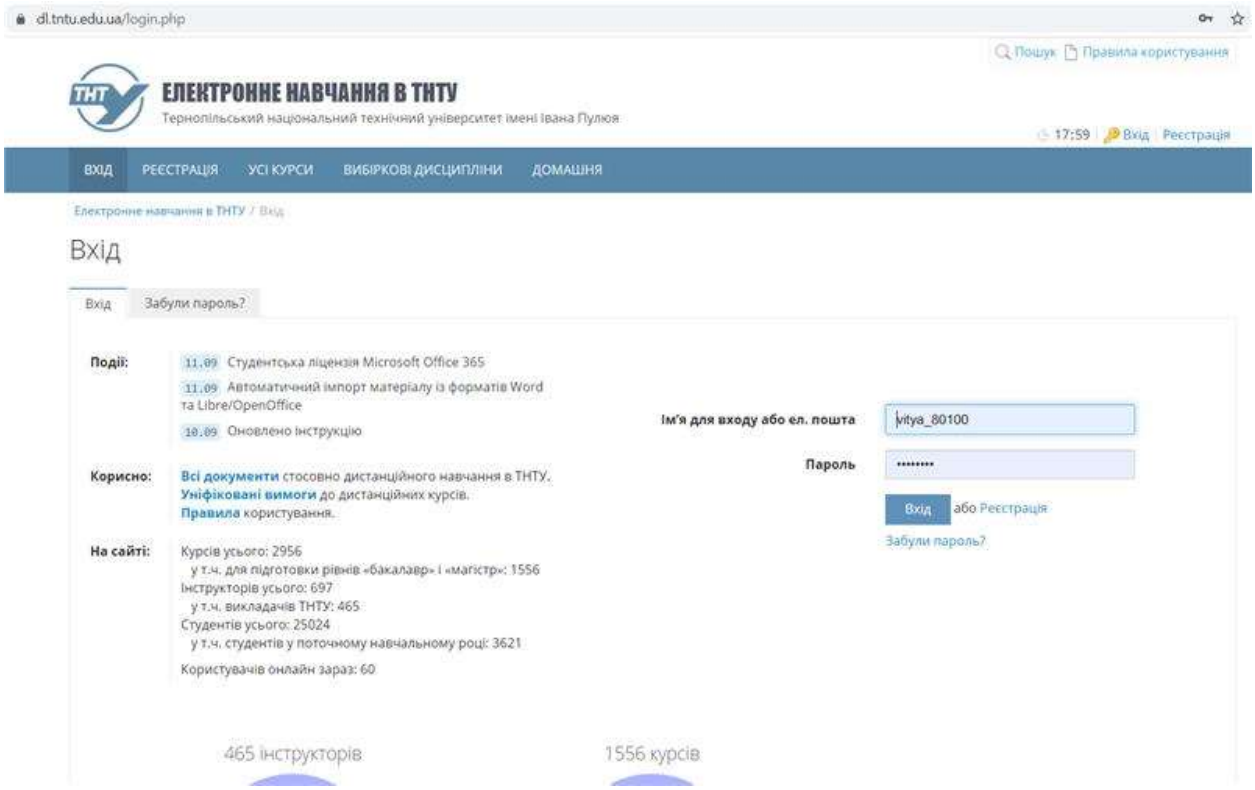

Рисунок 1.1 – Головна сторінка дистанційного навчання

Перейшовши на сторінку реєстрації потрібно заповнити необхідні поля про студента відповідно до паспорта. Після цього реєстрація студента завершена. Наступний крок – подати запит на курси з дисциплінами відповідно до навчальної програми студента що міститься у вкладці «НАВЧ. ПЛАН» у шапці головної сторінки користувача [3]. Також на сторінці профілю показано курси які він проходить чи є випускником.

1.1.3 Огляд функцій пошти дистанційного навчання

Пошта надає можливість спілкуватися із студентами та викладачами у межах університету. Перейти до пошти можна з будь-якої сторінки платформи. Для цього потрібно натиснути посилання з текстом «Пошта» у верхньому правому кутку (рис. 1.2). Також біля посилання на пошту можна дізнатися про кількість отриманих повідомлень із мигаючого жовтого кругу з числом нових повідомлень. У випадку відсутності нових повідомлень кругу не буде.

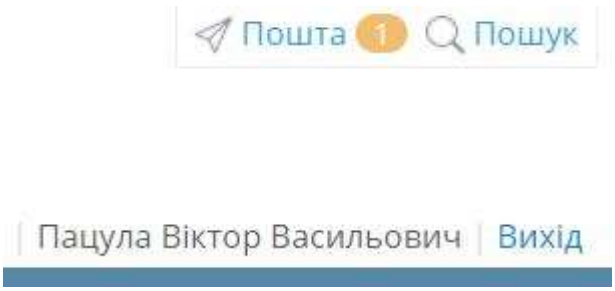

Рисунок 1.2 – Посилання для переходу до пошти

Головна сторінка поштового клієнту містить таблицю із вхідними повідомленнями що посортовані за часом у спадному порядку (рис. 1.3). Таблиця з листами складається з п’яти колонок: поле вибору, поле статусу, відправник, тема та дата отримання повідомлення.

Нижче наведено опис колонок:

– Поле вибору надає можливість вибрати повідомлення до видалення, або ж видалити усі повідомлення виділивши їх за допомогою поля вибору в заголовку таблиці.

– Поле статусу показує наявний стан повідомлення, а саме: «Новий!» коли повідомлення не прочитано, «Відповідь надіслана» коли відправлена відповідь та пусте значення коли повідомлення прочитане прочитано.

– Поле «Від» показує прізвище, ім’я та по батькові отримувача

– Поле «Тема» містить короткий опис листа.

– У полі «Дата» вказано час прибуття повідомлення у форматі «День, число місяць рік, час»

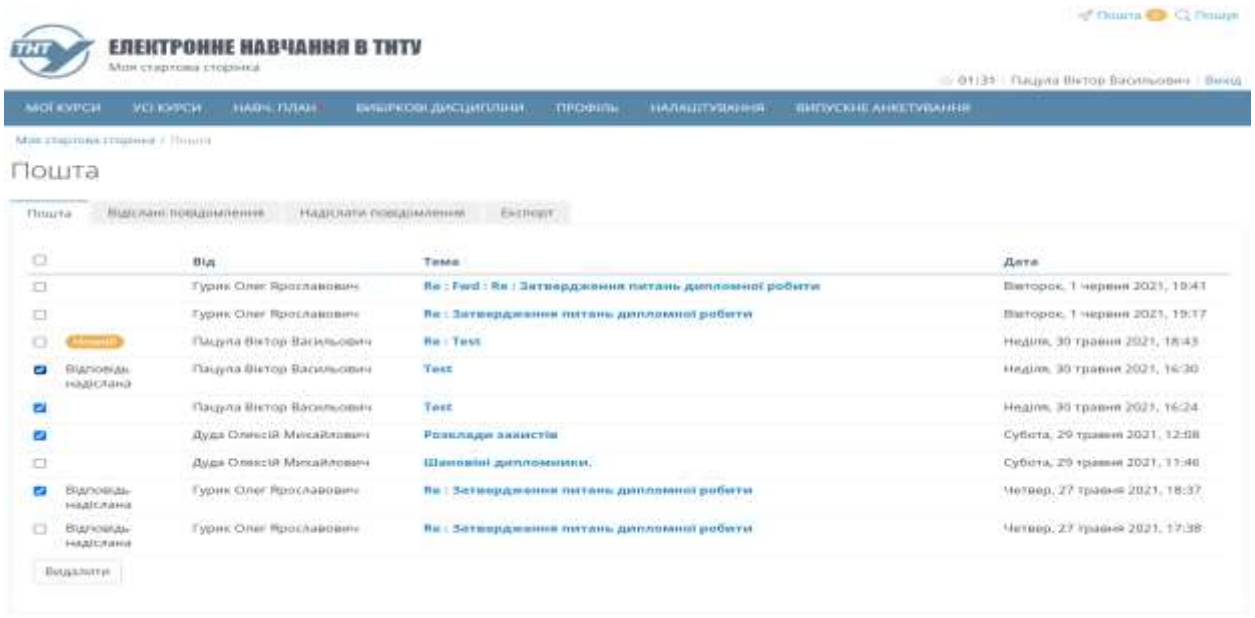

Рисунок 1.3 – Сторінка вхідних повідомлень

Натиснувши на поле «Тема» можна перейти до тіла повідомлення (рис. 1.4).

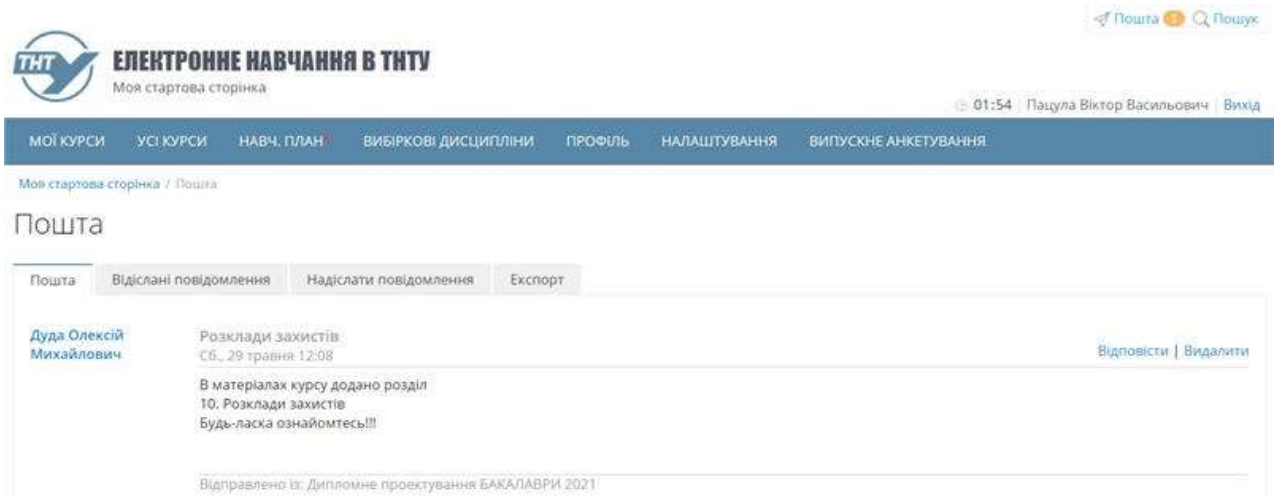

Рисунок 1.4 – Тіло повідомлення

У тілі повідомлення, окрім вже вказаних у таблиці даних та самого тіла повідомлення, можна відповісти на лист або видалити його. Також знизу тіла вказано з якого курсу відправник написав листа.

Щоб відповісти на повідомлення слід натиснути однойменне посилання, після чого відкриється меню для відправки відповіді (рис. 1.5). У відповіді поля для відправлення вже будуть заповнені.

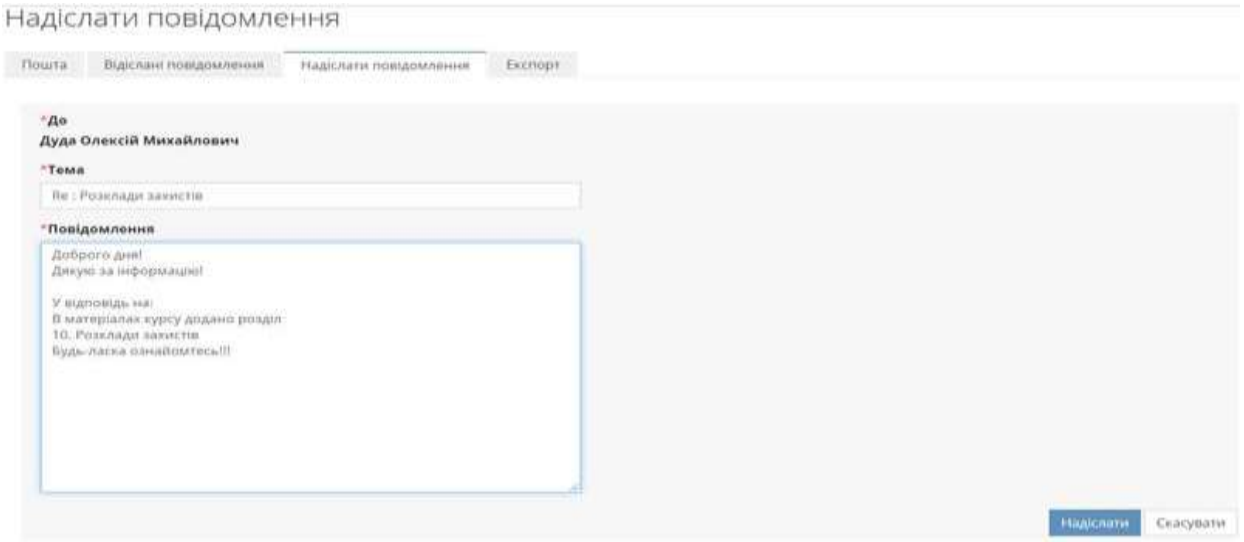

Рисунок 1.5 – Меню відповіді

Для відправлення нового повідомлення потрібно перейти у вкладку «Написати повідомлення» на сторінці пошти (рис. 1.3 та рис. 1.6).

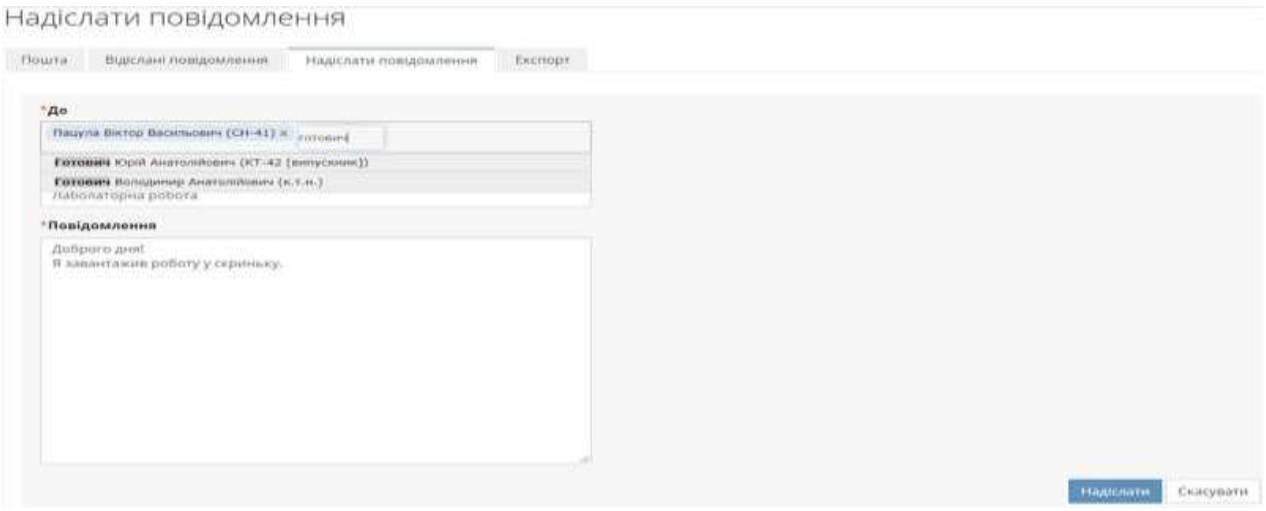

Рисунок 1.6 – Меню відправки повідомлень

Наступним кроком потрібно заповнити поля «До», «Тема» та «Повідомлення» - вони є обов’язковими. У полі «До» потрібно почати вводити отримувача, після чого з’явиться випадний список із результатом пошуку отримувачів. Їх можна вибрати кілька для відправки всім одного тексту. Для відправки потрібно натиснути кнопку «Надіслати».

1.2 Постановка завдання

Доступ до інформації з платформи для дистанційного навчання можна отримати лише зайшовши в електронний кабінет. З цього випливає що переглянути наявність нових повідомлень та відповісти на них можна лише з кабінету користувача. Це створює незручності у вигляді затримки відповіді на лист. Тому слід періодично заходити в кабінет користувача. Таку роботу можна автоматизувати.

Основні вимоги для вирішення проблеми:

– своєчасне отримання сповіщень;

– швидкий доступ до відповіді на повідомлення;

– швидкий доступ для написання нового листа;

– система повинна бути доступною для користування.

Своєчасне отримання сповіщень забезпечить прискорення спілкування мім користувачами платформи для дистанційного навчання. Швидкий доступ до відповіді надасть можливість своєчасно реагувати на повідомлення. Щоб попередні пункти виконались система повинна бути максимально доступною для користувачів.

Критерій доступності задовільняють додатки для смартфонів, а саме додатки для спілкування - месенджери. Вони є надзвичайно популярними саме через швидкий доступ до повідомлень. Один з таких месенджерів це «Телеграм».

Платформа «Телеграм» має наступні характеристики для вирішення проблеми:

– доступ здійснюється з усіх основних операційних систем;

– платформа є безкоштовною для використання;

– платформа має API для доступу через HTTP протокол;

– платформа має бібліотеки для використання API через мови різні мови програмування;

– доступні чат боти та API до них;

– детальна документація;

– усі дані зберігаються на веб серверах, а не на пристроях;

– одночасний доступ до платформи з кількох пристроїв.

Для логічної частини системи оповіщення буде використано мову програмування C# та платформу .NET [11]. Мова програмування C# є загального призначення. З її допомогою можна створювати застосунки для різних платформ. У межах кваліфікаційної роботи буде використано консольний застосунок. Його можна запускати як на операційну систему «Windows» так і на «Linux». Крос-платформеність мови дозволяє тримати одну кодову базу для усіх платформ. У випадку прив’язки до конкретної операційної системи кодову базу можна логічно розділити на бібліотеки. Це надасть можливість винести загальну кодову базу в крос-платформену бібліотеку та використовувати її модулями що прив’язані до певної операційної системи.

Наступна перевага мови програмування C# це велика кількість бібліотек для підключення що дозволить обрати найбільш оптимальний варіант для розробки.

Для доступу до кабінету користувача буде використано бібліотеку Selenium [12]. Вона дозволяє імітувати дії користувача у браузері. З її допомогою можна швидко та просто скласти алгоритми дій для потрібних функцій.

1.3 Огляд месенджера «Телеграм»

1.3.1 Вступ

Месенджер «Телеграм» це гнучка та безкоштовна для користування платформа, основний функціонал якої призначений для обміну повідомленнями та медіа (рис. 1.7).

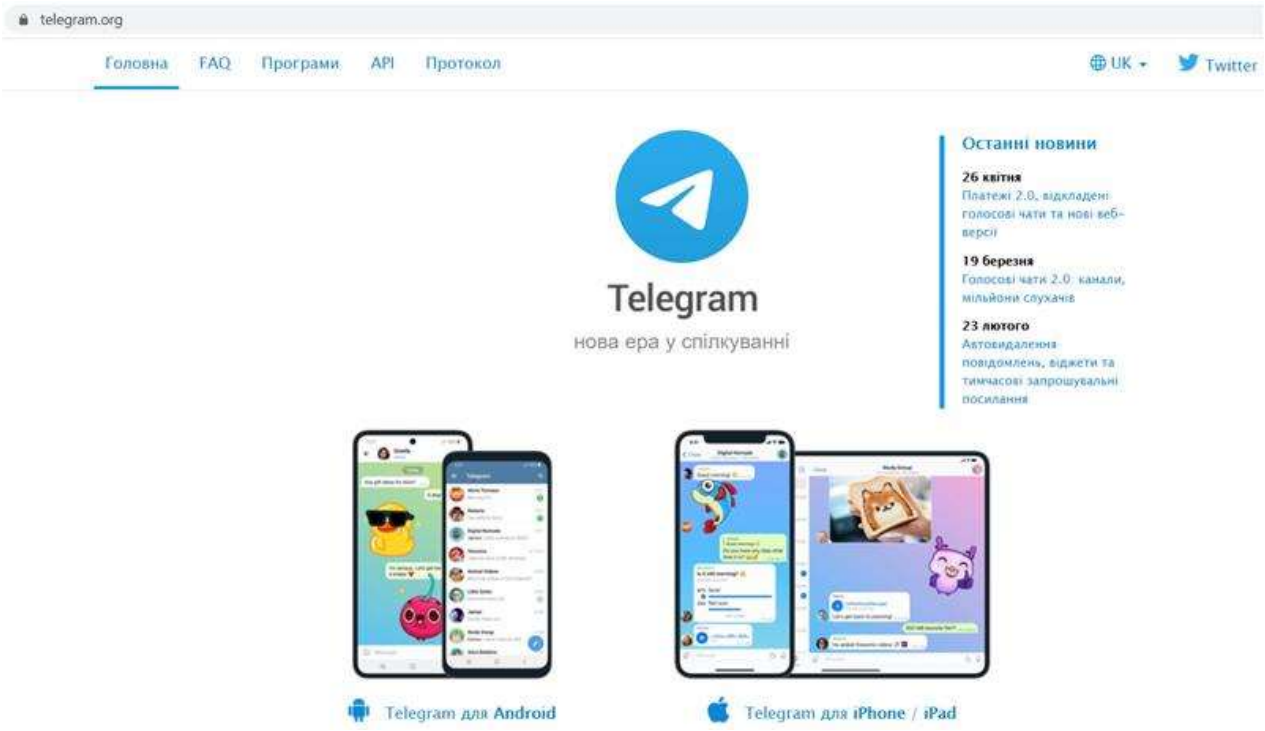

Рисунок 1.7 – Головна сторінка «Телеграм» [3]

Головний пріоритет платформи це швидкість та захищеність. У платформі можна створювати групи для спілкування з обмеженням до 200000 учасників, здійснювати захищені дзвінки, зберігати медіа, проводити голосові конференції та здійснювати оплату через підключені сервіси [4].

Доступ до платформи можна здійснювати через веб-застосунок, або ж завантажити версію для потрібного пристрою. Підтримуються такі платформи як Android, IOS, Windows, Linux. Тобто месенджер можна завантажити на смартфон, планшет чи комп’ютер з підтримуваними операційними системами. Веб застосунок знаходиться за адресою

https://webk.telegram.org/, https://webz.telegram.org/ або https://web.telegram.org [3] (рис. 1.8).

Платформа розробила віджети для веб браузерів. З їх допомогою можна додавати визначений функціонал платформи до користувацьких веб сайтів, а саме [5]:

– кнопка для надсилання веб сайту в чат або групу;

– віджет для перенесення посту з «Телеграм» на веб сайт;

– віджет для авторизації на веб сайтах через мессенджер;

– віджет для з’єднання коментарів на веб сторінці з чат групою месенджера.

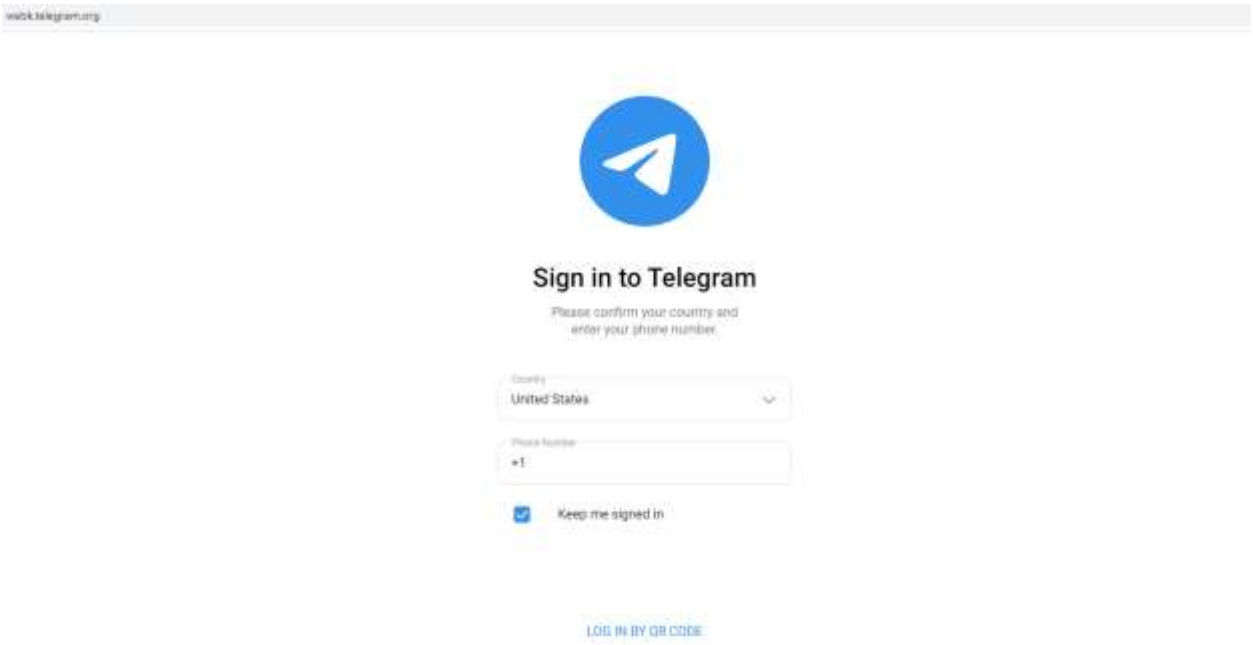

Рисунок 1.8 – Головна сторінка входу «Телеграм»

Для тих хто хоче зробити свій клієнт для платформи існує «Telegram API». З його допомогою можна зробити повноцінний клієнтський застосунок для інших платформ або зробити власний для підтримуваних платформ [5]. Реалізувати можна не тільки спілкування з API платформи, але й стильове оформлення клієнта.

1.3.2 Реєстрація

Реєстрація у месенджері відбувається за номером телефону з доступного клієнта месенджера. Для прикладу буде використаний додаток для Android. Спершу слід завантажити додаток з магазину додатків «Play Market» за посиланням - https://play.google.com/store/apps/details?id=org.telegram.messenger&hl=en_US&gl=US або з головної сторінки платформи - https://www.telegram.org/android [3]. Після входу в додаток буде запропоновано ввести актуальний номер телефону (рис. 1.9).

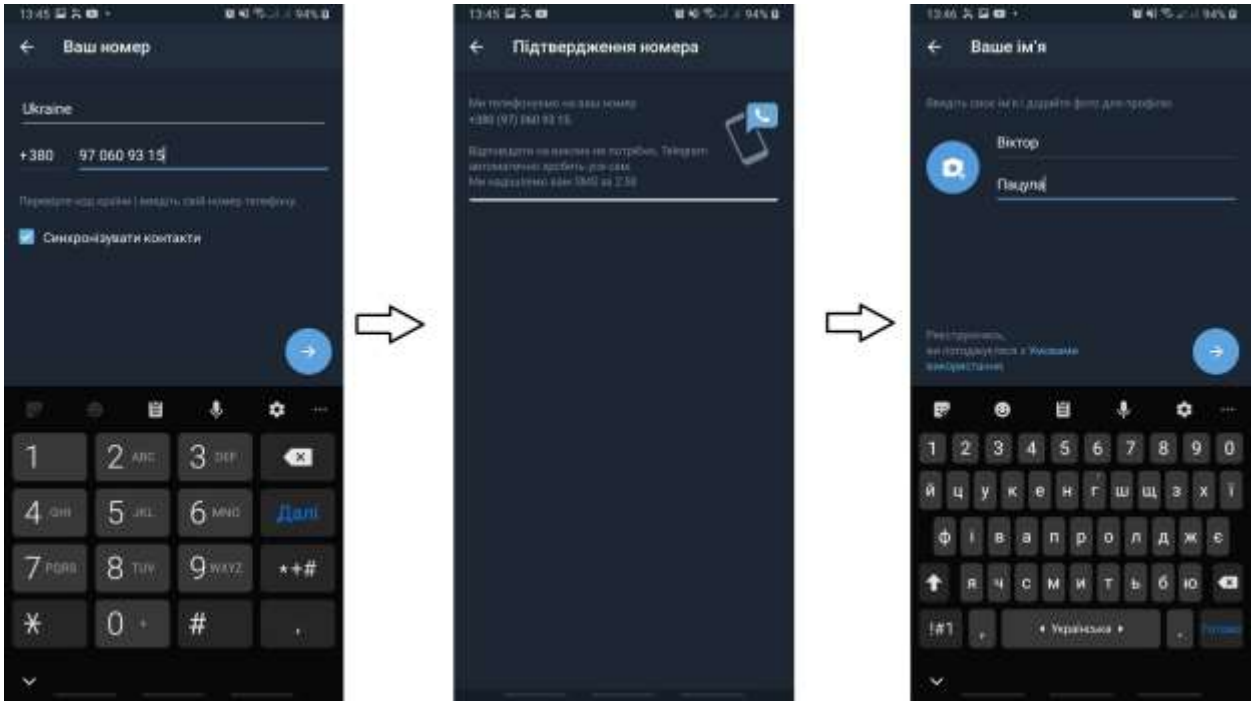

Рисунок 1.9 – Етапи реєстрації через додаток

На номер телефону прийде смс повідомлення з кодом який потрібно ввести у відповідне поле, або до користувача подзвонить сервіс що таким чином сам проведе верифікацію номеру (Рис. 1.9). Для цього потрібно надати права керування дзвінками для додатку. На дзвінок не потрібно відповідати.

Пройшовши верифікацію номеру телефону буде запропоновано ввести свої дані що будуть відображатися у профілі користувача (рис. 1.9).

1.3.3 Чати

Після завершення реєстрації додаток перейде на сторінку з чатами. Додаток також пройдеться по списку номерів телефону користувача та перевірить які з номерів вже зареєстровані у «Телеграм», тому користувач зразу після реєстрації матиме можливість почати спілкуватися. Для цього потрібно перейти у вкладку «Контакти» та вибрати контакт для спілкування, після чого відкриється чат (рис. 1.10). Інший спосіб знайти людину – за її ім’ям користувача [4].

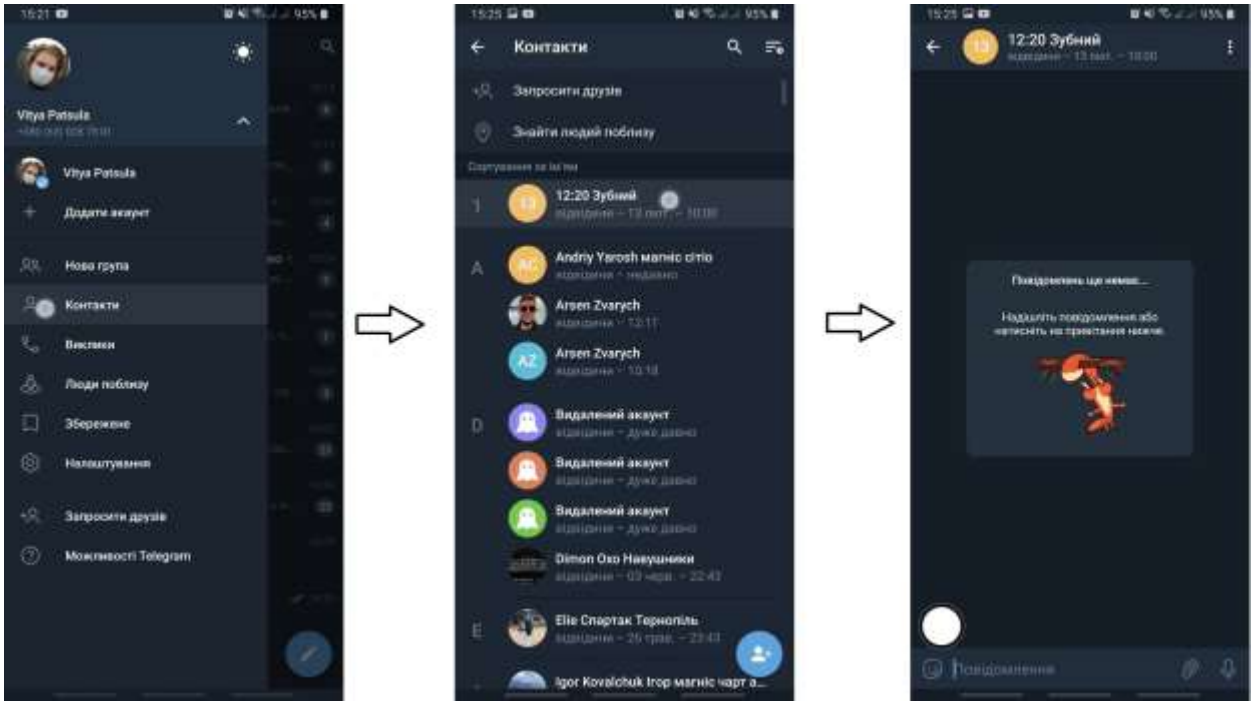

Рисунок 1.10 – Вибір контакту для спілкування

Окрім текстового повідомлення месенджер підтримує аудіоповідомлення, відеоповідомлення, дзвінки та аудіодзвінки [3]. Серед медіа можна відправляти (рис. 1.11):

– фото та відео;
– файли будь-якого розширення;
– розташування;
– контакти;

– музику.

Рисунок 1.11 – Відправка медіа у месенджері

«Телеграм» забезпечує безпечність спілкування за допомогою секретних чатів за допомогою захищеного протоколу передачі даних MTProto. Цей протокол забезпечує високу швидкість шифрування та дешифрування, внаслідок чого майже не впливає на швидкість передачі даних [4]. Для ще більш високого рівня безпеки месенджер має функцію секретних чатів. У таких чатах повідомлення шифруються на обох пристроях та не зберігаються на хмарних сервісах «Телеграм» [4].

1.3.4 Боти

Боти надають можливість створювати сторонні програми всередині «Телеграм». Користувачі взаємодіють з ботами шляхом відправки команд та повідомлень [6].

Боти мають такі можливості як [6]:

– користувацькі сповіщення та новини;

– інтеграція з іншими сервісами;

– проведення платежів;

– створення ігор;

– створення користувацьких застосунків.

Для того щоб створити бота, слід скористатися ботом @BotFather. У чаті бота надіслати команду для створення нового бота - /newbot (рис. 1.12).

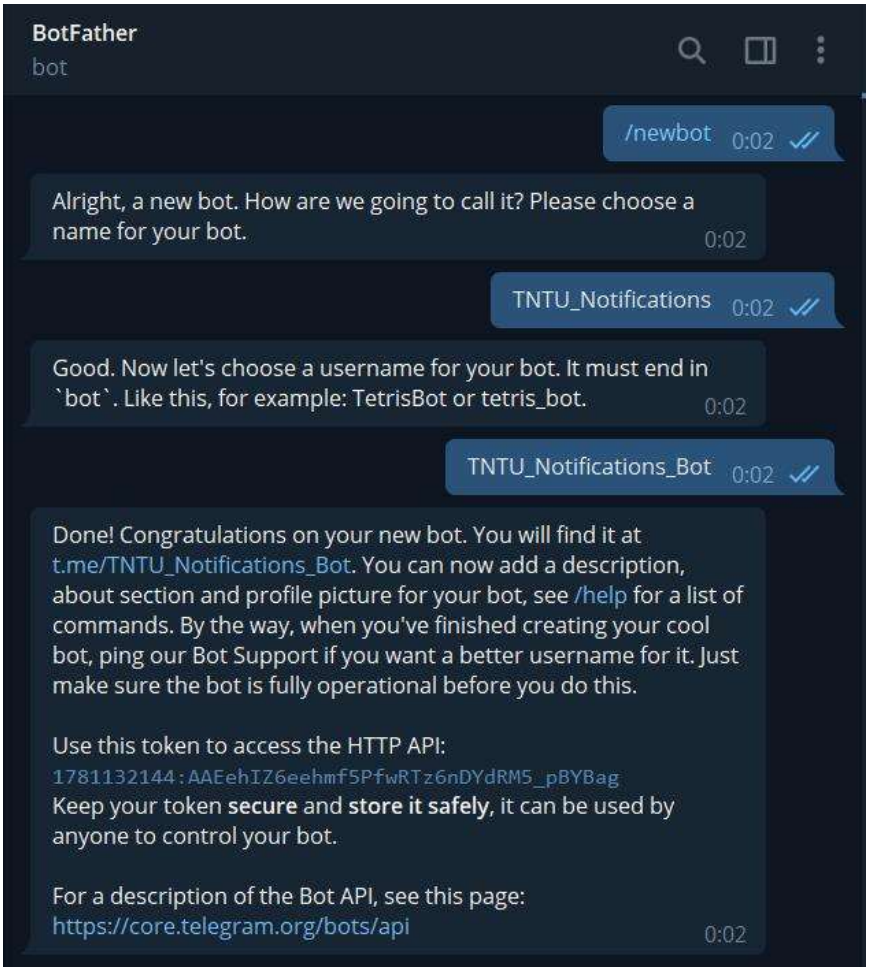

Рисунок 1.12 – Створення нового бота

Для створення бота слід ввести його назву та ім’я користувача. Після створення бота, до нього можна звертатись за іменем користувача. Також створюється HTTP API token за яким можна отримати доступ до керування ботом за допомогою веб протоколу HTTP [7].

1.4 Висновки до першого розділу

ТНТУ надає студентам зручне серидовище для освіти. Дисципліни поділені на курси, де є все необхідне для навчання: матеріали дисципліни, тести для перевірки знань, та можливість занять онлайн. Не менш важливо зберігати комунікацію між студентами та викладачами у межах університету. Для цього дистанційне навчання має пошту. Так як доступ до пошти здійснюється через дистанційне навчання то перевірити її можна лише зайшовши у свій обліковий запис. Це створює дискомфорт у вигляді періодичного входу в облікових запис або несвоєчасності отримання повідомлення.

Для вирішення проблеми пропонується підключити сповіщення про прихід повідомлення на дистанційному навчанні до месенджера «Телеграм». Він реалізує захищені протоколи зв’язку, є гнучким для розробки та безкоштовним. «Telegram API» є гнучким інструментом для розробки функціоналу швидких повідомлень. Це забезпечить необхідний рівень швидкості сповіщень що вирішить проблему з несвоєчасним отриманням повідомлень на пошті. Для цього буде використано можливості чат-бота у месенджері.

2 РОЗРОБКА СИСТЕМИ ОПОВІЩЕННЯ

2.1 Огляд інструментів та технологій розробки

2.1.1 .NET

Це платформа для створення крос-платформених застосунків розроблена компанією Microsoft. У .NET входіть кілька мов програмуванн, а саме [9]:

– C#;
– C++;
– Visual Basic;
– F#.

Для розробки системи оповіщення застосовується мова програмування C#. Це об’єктно-орієнтована та строго типізована мова програмування загального призначення з багатим функціоналом.

2.1.2 Visual Studio Community 2019

Visual Studio це зручне та безкоштовне серидовище для програмування на мовах .NET. З її допомогою можна розробляти консольні додатки, бібліотеки, десктопні та веб застосунки. Програма має зручний функціонал для розробки додатків [10]:

– IntelliSense;
– функціонал для доступу до системи контролю версій Git;
– інструменти для відлагодження.

Завдяки зручним можливостям для розробки це програма буде використана під час створення системи оповіщення.

2.1.3 Selenium

Selenium це інструмент до надає доступ до автоматизації дій у браузері. Selenium надає розробникам один спільний інтерфейс для роботи з основними браузерами, а саме [12]:

– імітація користувача у браузері;

– виконання JavaScript;

– керування вікнами браузера;

– завантаження та відправка файлів;

– можливість робити знімки екрану.

Даний інструмент значно полегшить розробку системи оповіщення. Він дозволить опустити роботу з протоколом Http, так як вона буде делегована реалізації Selenium драйвера.

2.1.4 AngleSharp

AngleSharp це бібліотека для маніпулювання HTML розміткою, написана на .NET. Вона надає наступний функціонал для розробників [13]:

– перетворення HTML розмітки в програмний об’єкт для подальших маніпуляцій;

– виконання JavaScript над заданою розміткою;

– пошук елементів за допомогою CSS селекторів;

– можливість додати, видалити чи редагувати елементи розмітки.

Дана бібліотека буде застосована для отримання інформації з HTML розмітки дистанційного навчання.

2.2 Розробка системи оповіщення

2.2.1 Загальна архітектура

Для забезпечення логічного розділення функцій [15], їх було виділено у модулі. Архітектура системи оповіщення розділена на 3 основних модулі, а

саме:

1. модуль для роботи із дистанційним навчанням ;
2. модуль для роботи із «Телеграм» ботом;
3. модуль що зберігає загальні моделі для спілкування інших модулів.

Загальний принцип роботи системи оповіщення проходить у 6 кроків (рис. 2.1).

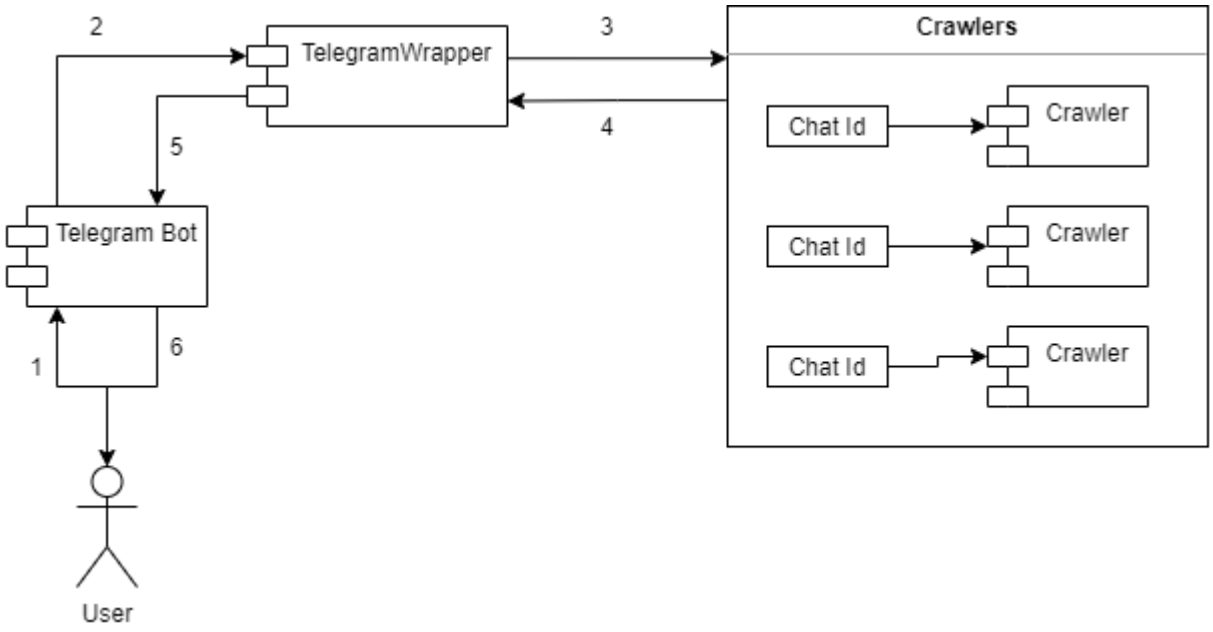

Рисунок 2.1 – Загальна архітектура системи сповіщення

Перший крок це взаємодія користувача із чат ботом. Користувач надсилає боту команду. Другий крок це обробка команди у модулі «TelegramWrapper». Команда перевіряється на валідність та в разі успіху передає обробку запиту потрібній функції.

Для доступу до дистанційного навчання використовується модуль «Crawler». Щоб забезпечити можливість одночасної обробки повідомлень від кількох користувачів було вирішено створювати новий об’єкт модуля «Crawler» для кожного користувача. Для цього було створено масив, де індекс комірки це ідентифікатор користувача, та сама комірка містить екземпляр модуля для роботи із дистанційним навчанням. Тому третій крок

це отримання потрібного екземпляра модуля «Crawler» для відповідного користувача.

Четвертий крок це дії безпосередньо із дистанційним навчанням. Після виконання необхідних команд модуль «Crawler» приводить до загальної моделі результат та передає його назад у модуль «TelegramWrapper». Після чого модуль оформлює результат функції у повідомлення для користувача, що є п’ятим кроком. Останній, шостий крок це відправка повідомлення користувачу чат ботом.

2.2.2 Архітектура сповіщень

Після надсилання користувачем своїх даних, система оповіщення буде перевіряти наявність нових листів на пошті дистанційного навчання. Для надсилання повідомлення про результат перевірки використано паттерн проектування «Наглядач» [16]. У .Net даний паттерн реалізують події (рис. 2.2).

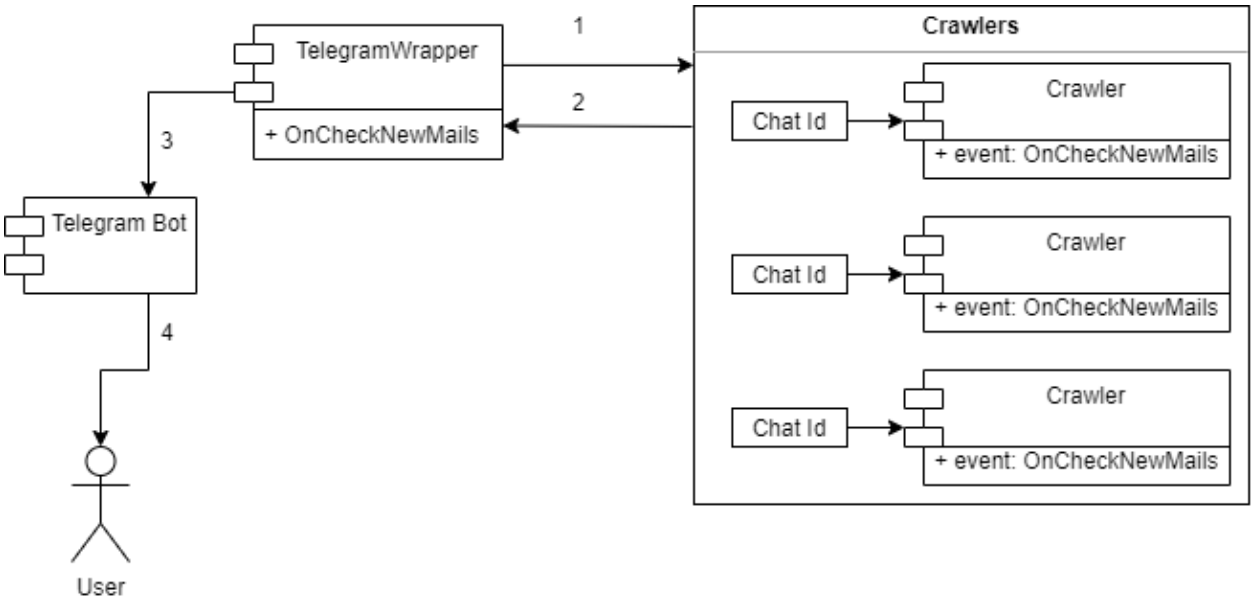

Рисунок 2.2 – Архітектура для сповіщень

Після успішного входу користувача для нього створюється екземпляр модуля «Crawler» та розміщується у масиві з іншими екземплярами. Перед

розміщенням новоствореного об’єкта, модуль «TelegramWrapper» підписується на подію «OnCheckNewMails», присвоївши йому обробника. Таким чином кожен екземпляр модуля «Crawler» матиме змогу повідомляти модуль «TelegramWrapper» про результат перевірки пошти.

2.2.3 Розробка модуля «Crawler»

Даний модуль відповідає за пряму взаємодію із дистанційним навчанням та представлений однойменним класом (додаток А, лістинг 1.1). Даний клас має конструктор з 2 параметрами для отримання даних входу конкретного користувача. Це потрібно для забезпечення доступу до облікового запису на дистанційному навчанні.

Клас відповідає за усі дії користувача:

– вхід та вихід з облікового запису;

– отримання списку листів з пошти дистанційного навчання;

– отримання тіла повідомлення;

– відповідь на конкретний лист.

Загальний алгоритм роботи наступний:

1. вхід в обліковий запис;
2. отримання необхідних даних або виконання потрібних дій;
3. у випадку отримання даних, їх привести до загального формату;
4. вихід з облікового запису;
5. повернення даних якщо потрібно.

Для виконання дій користувача та отримання даних у форматі HTML розмітки [17] використовується бібліотека Selenium [12]. Для маніпулювання з HTML розміткою використовується бібліотека AngleSharp [13]. На лістингу 2.1 показано приклад використання бібліотек у функції «GetNewMailes».

Лістинг 2.1 – Приклад використання бібліотек

```
private EdgeDriver _driver;
public IEnumerable<MailItem> GetNewMailes()
```

```
            {
                lock (_driver)
                {
                    _driver.Login(Login, Password);

                    int newMailes = CheckNewMailes();
                    if (newMailes == 0)
                        return Array.Empty<MailItem>();

                    _driver.GoToMailBox();
                    var document = GetMailesHtml();
                    var mailesNew =
ParseMailesNew(document).ToList();

                    _driver.Logout();

                    return mailesNew;
                }
            }
      // go to MailBox page first
             private IHtmlDocument GetMailesHtml()
             {
                 IWebElement elem = null;
                 lock (_driver)
                 {
                     elem =
_driver.FindElementByCssSelector(Css.MailAnswerBody);
                 }
                 string mailesHtml =
elem.GetAttribute("outerHTML");

                 //Use the default configuration for AngleSharp
                 var config = Configuration.Default;
                 var context = BrowsingContext.New(config);
                 var parser = context.GetService<IHtmlParser>();

                 var document =
parser.ParseDocument(mailesHtml);
                 return document;
             }
```

```
private IEnumerable<MailItem> ParseMailesNew(IHtmlDocument
document)
                {
                    var mailes =
document.QuerySelectorAll(Css.MailBoxRow).ToMailNewItemList();
                    return mailes;
                }
```

У функції «GetNewMailes» спершу виконується вхід до облікового запису. Для цього використовується змінна «_driver» типу «EdgeDriver». Даний тип використовується для проведення усіх дій користувача та отримання елементів веб сторінки.

Для зручності виконання загальних дій, таких як вхід та вихід з облікового запису було використано методи розширення [18]. Наступний крок - перевірити чи є нова пошта. Якщо немає то повернути пустий масив. Інакше перейти до пошти. Виконується це методом розширення «GoToMailBox».

Після переходу до пошти скриньки необхідно отримати HTML розмітку сторінки для подальшого отримання списку повідомлень. Для цього використовується метод «GetMailesHtml». Усередині методу використовуючи інший - «FindElementByCssSelector» отримується елемент веб сторінки за допомогою css селекторів пошуку [19]. Селектори винесені в константи в класі «Css» (додаток А, лістинг 1.2). Константи та класи названі згідно з «PascalCasing» [20]. Після отримання елементів розмітки, їх необхідно перетворити у клас загального використання «MailItem» (додаток А, лістинг 1.7). Для цього використовується метод розширення «ToMailNewItemList» (додаток А, лістинг 1.5). У даному методі повертається інтерфейс загальної колекції «IEnumarable<MailItem>» [21]. Для повернення елементів масиву використовується оператор мови C# «yield return» [22].

Отримавши потрібні елементи HTML розмітки та перетворивши їх у загальний формат моделі виконується вихід із облікового запису та повернення даних.

Ще одна важлива функція модуля це періодична перевірка пошти на нові листи. Для цього використовується таймер [23] (лістинг 1.2). Таймер встановлюється у методі «SetUpTimer». Для оповіщення про отримання нових повідомлень використовується подія [24] «OnCheckNewMails».

Лістинг 2.2 – Встановлення таймера

```
public delegate void NewMailesEventHandler(object sender, NewMailesEventArgs e);
            public event NewMailesEventHandler OnCheckNewMails;

            private void SetUpTimer()
            {
                _timer = new Timer(TimeSpan.FromMinutes(TimerOffsetMinutes).TotalMilliseconds);
                _timer.Elapsed += OnElapsed;
                _timer.Start();

                void OnElapsed(object sender, ElapsedEventArgs e)
                {
                    var newMailes = GetNewMailes().ToList();
                    OnCheckNewMails?.Invoke(sender, new NewMailesEventArgs(newMailes));
                }
            }
```

Для запобігання несинхронізованих дій всередині класу «Crawler» Використовується конструкція lock [25] всюди де використовується змінна «_driver».

2.2.4 Розробка модуля «TelegramWrapper»

Даний модуль відповідає за взаємодію телеграм бота із модулем «Crawler» (додаток А, лістинг 1.8). Для використовування API чат бота була використана бібліотека «Telegram.Bot» [26].

Початкові налаштування класу відбуваються в асинхронному [27] методі «SetUp». Спершу ініціалізується екземпляр класу бібліотеки із заданим токеном бота. Після чого відбувається підписка на події що сповіщають про прихід нового повідомлення: «OnMessage» та «OnCallbackQuery». Після підписки на необхідні події починається прослуховування повідомлень бота за допомогою метода «StartReceiving» (додаток А, лістинг 1.8).

Первинна обробка повідомлень відбувається у методі «BotOnMessageReceived». Тут за допомогою конструкції switch case вирішується яка команда прийшла від користувача.

Першим кроком для використання бота це відправка даних для входу в обліковий запис дистанційного навчання. Після приходу відповідної команди викликається метод «CheckCredentials», де відбувається пробний вхід та вихід. Після чого створюється екземпляр класу «Crawler» із заданими даними входу та додається у словник «_crawlers», де ключ це ідентифікатор мату звідки прийшли дані про вхід. Також відбувається підписка на подію «OnCheckNewMails» для отримання сповіщень про наявність чи відсутність нових повідомлень. Підписка відбувається за допомогою lambda expression [28], усередині якого викликається метод «AnswerCheckNewMailes». Він використовується при відправці повідомлень як із команди, так і з підписки на подію (додаток А, лістинг 1.8).

Тепер при виклику інших команд для заданого користувача братиметься конкретний екземпляр класу «Crawler». Для усіх інших команд алгоритм дії наступний:

– витягнути екземпляр класу «Crawler» для користувача;

– викликати необхідну функцію модуля;

– відправити дані користувачу.

Модуль «TelegramWrapper» запускається з головного методу «Main». Це метод що є вхідною точкою для кожної програми .Net. Завдяки можливостям 5 версії фреймворку його можна зробити асинхронним що забере необхідність запускати асинхронні методи іншими шляхами ніж await [27].

2.3 Огляд функцій системи оповіщення

Основне завдання що вирішуватиме система оповіщення це своєчасне отримання повідомлень із платформи дистанційного навчання. Для цього система оповіщення буде перевіряти наявність нових повідомлень із заданим проміжком часу.

Для доступу до бота у клієнті «Телеграм» слід ввести його назву у полі пошуку – «TNTU_Notification_Bot». Після чого можна перейти до функцій бота (рис. 2.1).

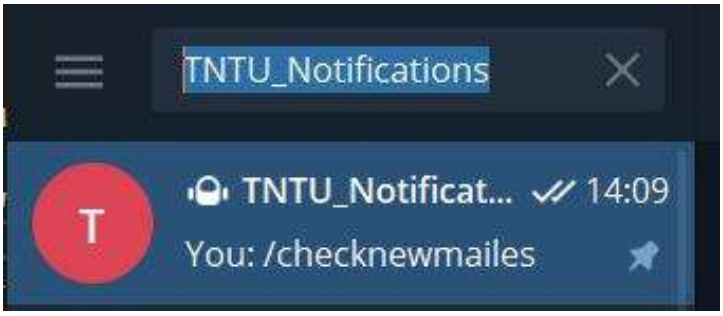

Рисунок 2.1 – Отримання нових повідомлень

Для доступу до функціоналу бота слід ввести символ «/» або натиснути кнопку з таким символом у полі вводу (рис. 2.2).

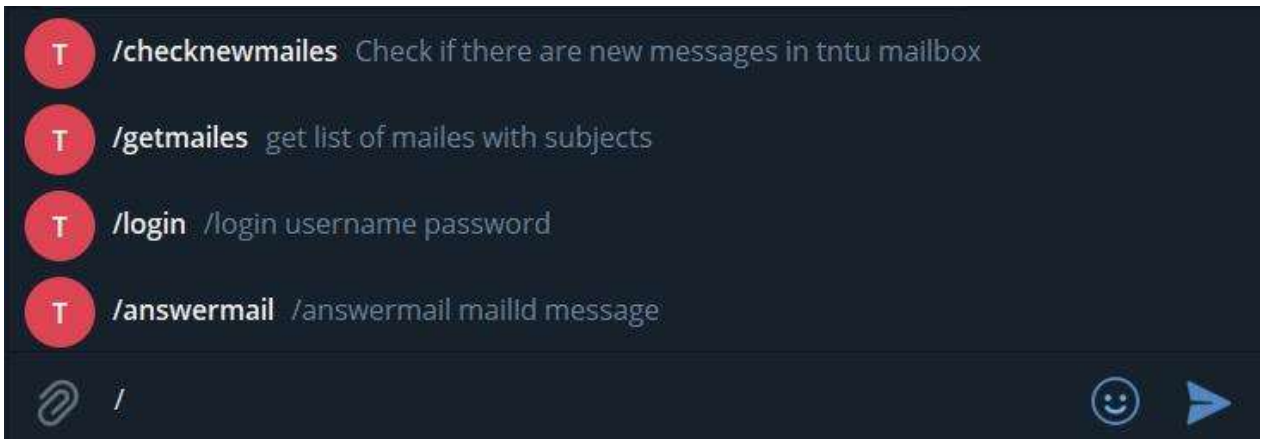

Рисунок 2.2 – Огляд доступних команд

Перш за все потрібно ввести дані облікового запису щоб бот мав можливість зайти до облікового запису користувача. Для цього слід ввести команду «/login» та дані входу разом з командою (рис. 2.3).

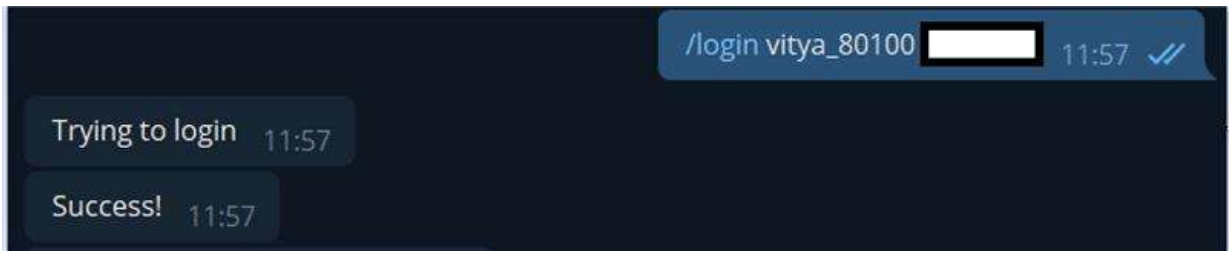

Рисунок 2.3 – Відправка даних для входу

Система сповіщення намагається увійти до облікового запису користувача та після успішного входу виводить відповідне повідомлення. Для кожного користувача бот зберігає його дані для входу для подальшого використання.

Для забезпечення швидкої відповіді на пошті розроблено функціонал для чат бота що дозволяє отримувати інформацію про нове повідомлення. Для цього користувачу потрібно відправити команду /checknewmailes до бота (рис. 2.4).

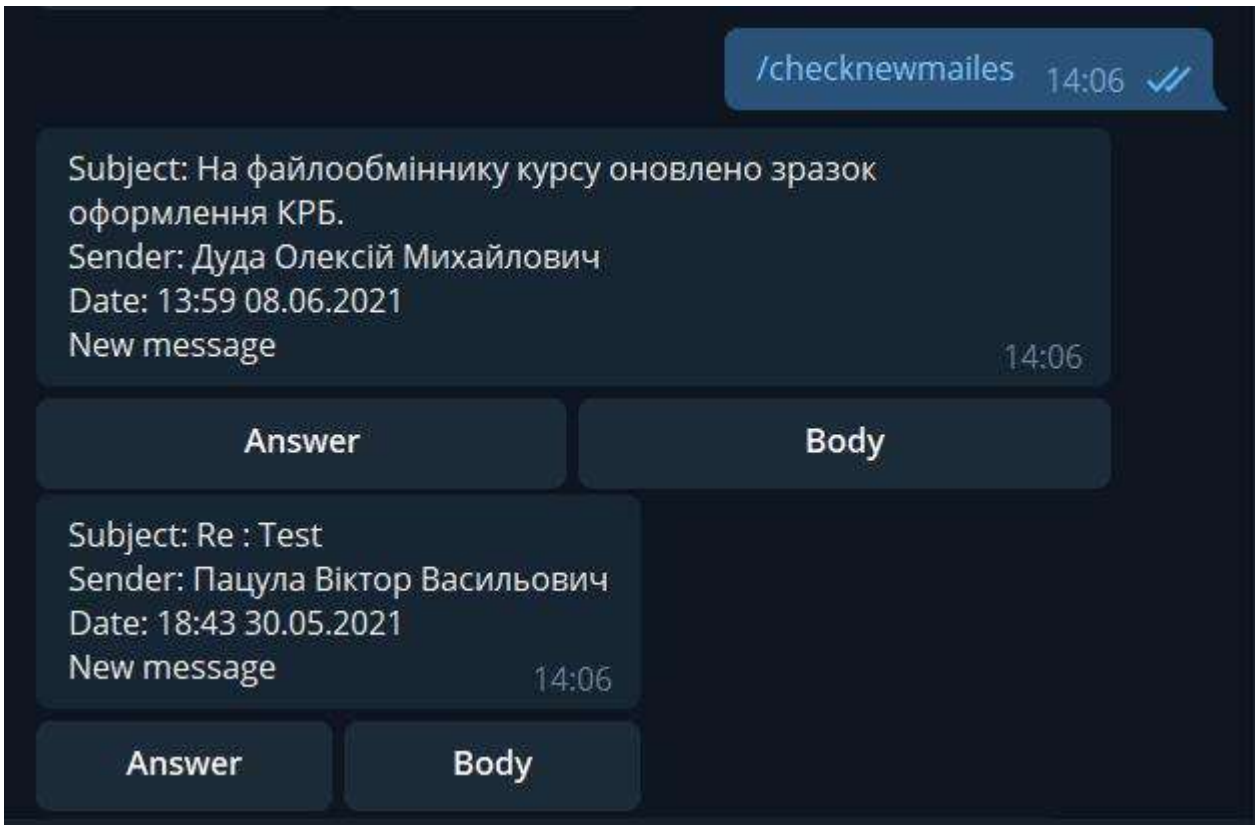

Рисунок 2.4 – Отримання нових повідомлень

Результат команди – список нових повідомлень або повідомлення про їх відсутність. Повідомлення містить наступні елементи:

– тема листа;

– відправник;

– дата та час отримання повідомлення;

– позначка про те що лист новий;

– кнопка «Answer»;

– кнопка «Body».

Після отримання списку з новими повідомленнями користувач може переглянути тіло повідомлення або відповісти на нього (рис. 2.5).

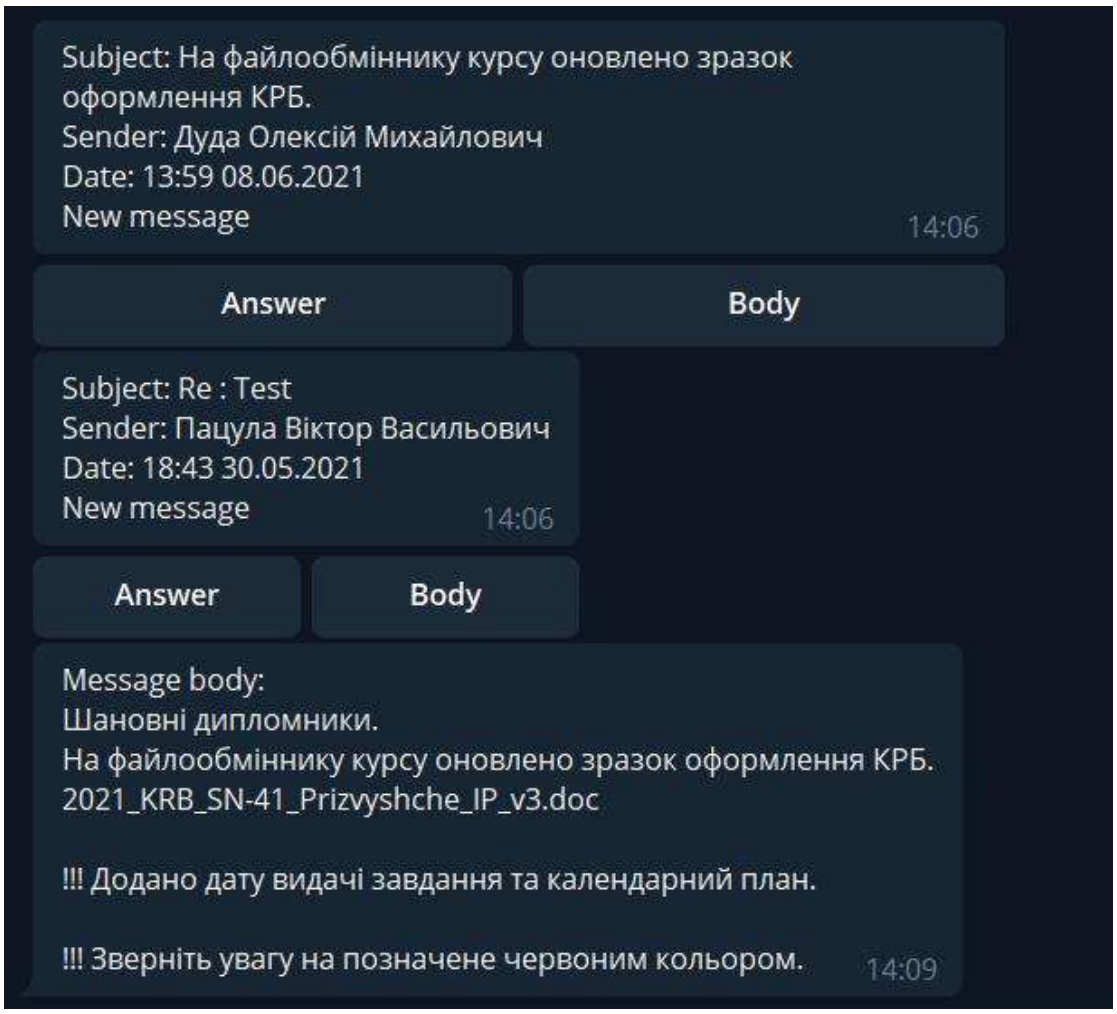

Рисунок 2.5 – Отримання тіла повідомлення

Для отримання тіла повідомлення потрібно натиснути кнопку «Body». Після чого бот відправить запит системі оповіщення на отримання тіла. Результатом буде повідомлення з тілом листа (рис. 2.5).

Для відповіді на повідомлення слід натиснути кнопку «Answer». Після чого бот запропонує написати відповідь для пересилання. Після успішного відправлення бот інформує користувача (рис. 2.6).

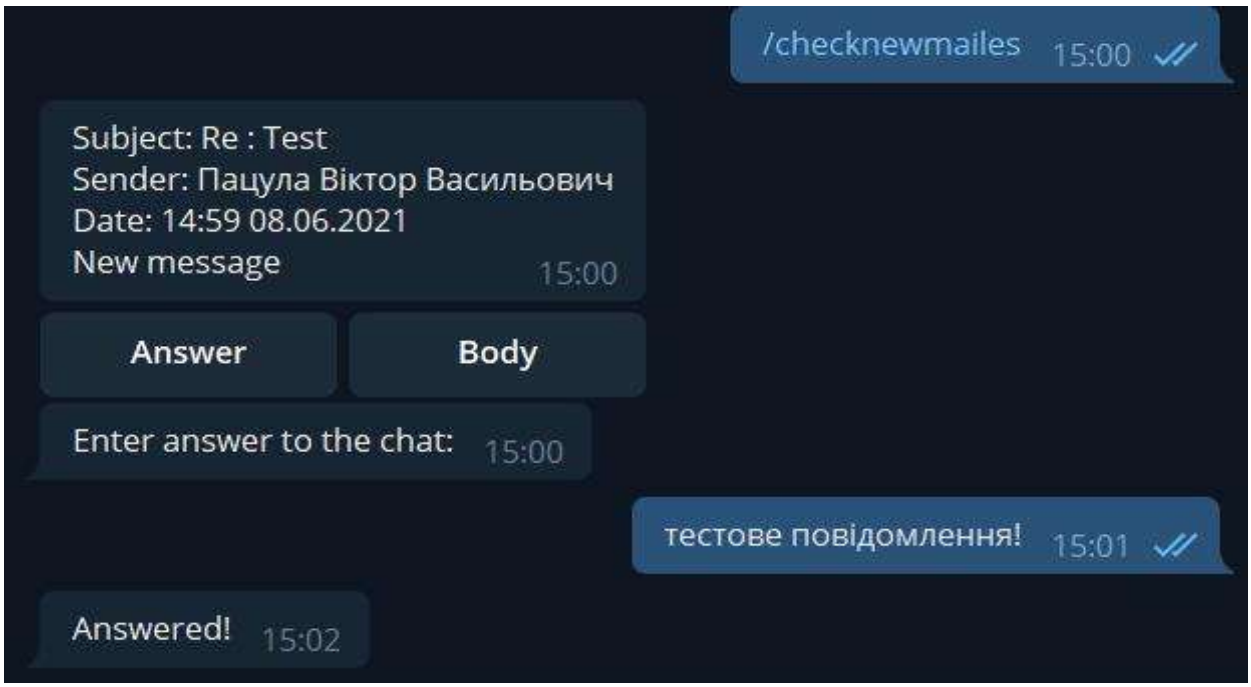

Рисунок 2.6 – Відповідь на повідомлення

Окрім перевірки наявності нових повідомлень можна отримати список усіх вхідних повідомлень. Для цього потрібно ввести команду «/getmailes». Після чого бот виведе список усіх повідомлень користувача у вкладці «Пошта» з дистанційного навчання (Рис. 2.7).

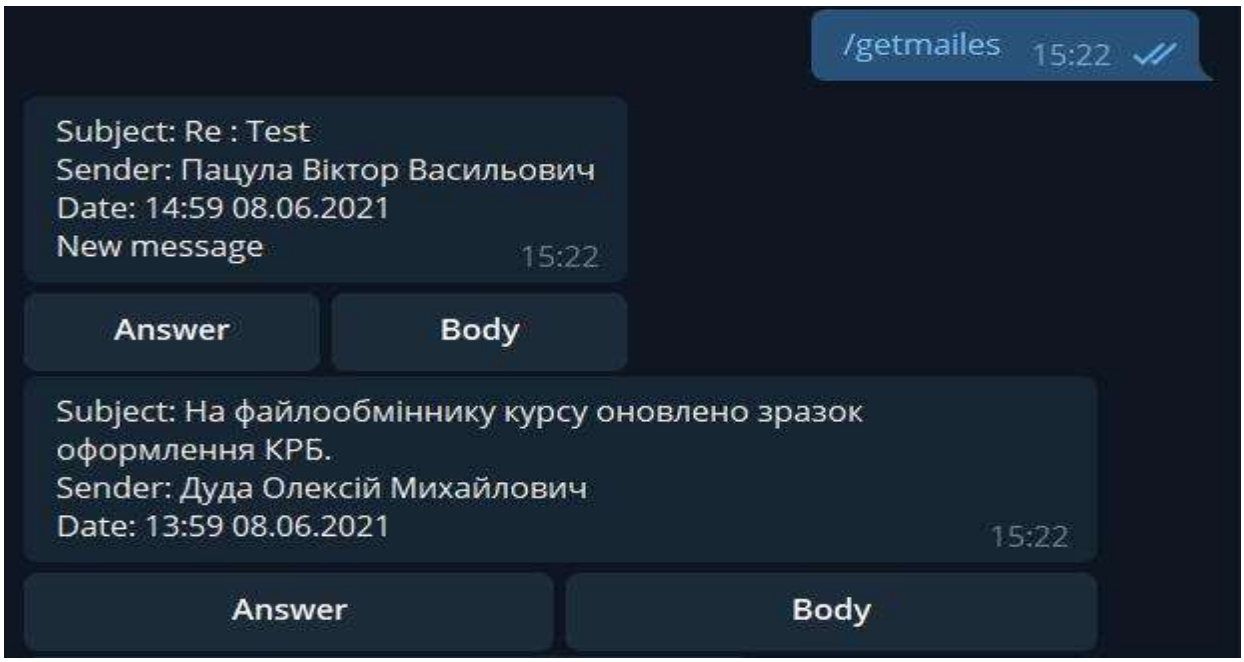

Рисунок 2.7 – Отримання усіх повідомлень

На отриманий список листів також можна відповідати та отримати їх вміст.

Ще одна функція чат бота – періодична перевірка нових повідомлень у скриньці дистанційного навчання. Для цього система оповіщення періодично заходить в обліковий запис користувача та перевіряє наявність листів. Якщо нових листів немає то система відправляє відповідне повідомлення користувачу у «Телеграм», але без сповіщення. Таким чином користувач не буде відволікатися на непотрібну інформацію. Якщо система оповіщення знайшла нові повідомлення то чат бот відправляє відповідне повідомлення користувачу, аналогічне результату команди «/checknewmailes».

2.4 Висновки до другого розділу

В другому розділі кваліфікаційної роботи було розглянуто інструменти для розробки системи оповіщення, а саме бібліотеки для взаємодії із дистанційним навчанням, маніпулювання HTML розміткою та керування чат бота.

Наступним пунктом розглянуто загальну архітектуру запитів починаючи від повідомлення користувача до входу системи в обліковий запис. Також розглянути загальний принцип перевірки наявності нових повідомлень у користувача.

Реалізацію системи було розглянуто на основі 2 модулів: «Crawler» та «TelegramWrapper». Після чого розглянуто функціонал системи оповіщення з вигляду користувача.

3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ

3.1 Стихійні лиха та їх класифікація

До природних антропогенних явищ, які називаються також стихійними лихами, що призводять до великих людських і матеріальних втрат, відносяться: землетруси, повені, бурі, тумани, заморозки, дощі і хуртовини, пожежі [29].

Рисунок 3.1 - Класифікація стихійних лих [33]

Землетруси - це струси земної кори, викликані рухами гірських мас в її надрах, виверженнями вулканів, обваленнями підземних карстових печер. Тільки в 20 столітті землетруси забрали близько 10 мільйонів життів по всьому світу [29].

Повені - це затоплення низинних територій, зазвичай прибережних, в результаті підйому річок після сильних дощів, під час відлиги. Повені призводять до численних людських жертв та економічних збитків.

Шторми - це сильні порушення атмосферної рівноваги. У поєднанні з сильним вітром, громом і блискавкою, а також рясними опадами вони є

причиною загибелі людей і пошкодження будівель, комунікацій та сільського господарства.

Тумани - це суспензії мікроскопічних водяних крапель, що летять низько над землею у вигляді легких хмар, що утворюються в результаті конденсації охолодженої водяної пари, що міститься в атмосфері Землі. Знижуючи видимість нижче одного кілометра, вони є причиною аварій літаків, кораблів і автомобілів.

Морози - це температура повітря нижче 0 ^{o}C, яка може викликати обмороження і навіть замерзання людей, комунікаційні та економічні колапси.

Дощі і хуртовини - це атмосферні опади у вигляді крапель води або снігу; рясні і тривалі, разом з вітрами або морозами, можуть бути причиною повеней або заметів, паралізують комунікації, зв'язок, постачання. Вони також можуть стати причиною загибелі людей.

Пожежі - це загоряння, в результаті яких горять будівлі, ліси, торф'яники, посіви. Через несподіване виникнення і швидке розповсюдження вони можуть викликати вторинні наслідки, наприклад, вибухи паливних баків, викиди отруйних парів і газів, обвалення будівель.

Тривалі посухи і небезпечні біологічні явища, наприклад, епідемії, різке зростання чисельності шкідників (сарани, гризунів), також набувають характеру стихійного лиха.

Стихійні лиха - це явища природи, які призводять до порушення нормальної життєдіяльності, які представляють загрозу для людей та їхнього майна, що викликають порушення в функціонуванні економіки, транспорту, зв'язку.

Виникнення небезпечних природних явищ викликає необхідність попередження населення про загрозу небезпеки на певній території і прийняття необхідних превентивних і оборонних заходів. Залежно від типу загрози і її масштабу мають бути задіяні відповідні служби, які діють в

звичайному режимі, або мобілізуються зі складу населення у разі наявної потреби.

3.2 Розрахунок місцевого освітлення для роботи з комп’ютером

Освітлення - дуже важливий фактор при виконанні будь-якої роботи, де ми проводимо більшу частину дня і сидіння перед монітором створює додаткове навантаження на зір. Кожне робоче місце, обладнане комп'ютером для безперервної роботи, має відповідати стандартам щодо освітлення. Щоб комфортно виконувати роботу і не перевантажувати зір, необхідно привести тип освітлення у відповідність зі специфікою роботи.

Рисунок 3.2. - HSK LEDY - Світильник Draco D-66 - стандартний панельний світильник 60x60 см [32]

Кращі умови - це ті, в яких у нас є доступ до природного світла, який є нашим найбільшим союзником. Однак реальність в офісах зазвичай зовсім інша, тому при виборі освітлення варто орієнтуватися на джерела світла, які будуть максимально наближені до сонячного світла. Світло навколо нас має

величезне значення для здоров'я і благополуччя наших очей, що також відображається на якості роботи. Світло монітора сильно напружує очі, тому не рекомендується працювати перед монітором з вимкненим світлом [32].

Ідеальне освітлення житлових та промислових приміщень в першу чергу залежить від вибору екологічно та технічно досконалих світильників.

Формула розрахунку освітлення приміщення наступна:

Світловий потік Лм = X*Y*Z (1.1) [30]

X – стандарт норми освітленості (Лк);

Y – площа приміщення (м²);

Z – коефіцієнт запасу.

Коефіцієнт висоти стелі Z становить:

– при висоті від 2,5 м до 2,7 м – коефіцієнт 1;
– від 2,7 до 3 м – коефіцієнт 1,2;
– від 3 до 3,5 м – коефіцієнт 1,5;
– від 3,5 до 4,5 м – коефіцієнт 2;

Таблиця 3.1. - Розрахунок освітленості приміщення [30]

Лампа	**Потужність, Вт**	**Світловий потік, Лм**	**Кількість ламп, шт.**
1	2	3	4
Накалювання	40	470	4500/470=9,5
Галогенна	32	470	4500/470=9,5
Енергозберігаюча	15	700	4500/700=6,4
Світлодіодна	10	750	4500/750=6.0

3.3 Санітарно-гігієнічні вимоги до умов праці

Положення про загальні правила охорони праці та техніки безпеки визначає тип і розмір гігієнічних і санітарних норм на підприємстві і

визначає вимоги, яким вони повинні відповідати, а саме:

- Приміщення повинні бути адаптовані до кількості зайнятих працівників, використовуваних технологій і видів робіт і умов праці;
- Вони повинні знаходитись в стані, що забезпечує безпечне і гігієнічне використання; це правило також відноситься до обладнання, розташованого в приміщеннях;
- Їхнє розташування повинне виключати необхідність проходження через приміщення, де використовуються отруйні речовини або потенційно інфіковані матеріали, або де виконуються особливо брудні роботи (за винятком працівників, які працюють з цими речовинами);
- Вони повинні опалюватися, освітлюватися і вентилюватися відповідно до технічних і будівельних норм і українських стандартів;
- Вільна висота приміщень повинна бути не менше 2,5 м (за винятком громадських , висота яких повинна бути не менше 3,0 м). Правила дозволяють зменшувати вільну висоту приміщень до 2,2 м відносно підвалів, льохів або горищ (за винятком загальнодоступних туалетів і ванних) [30].

Роботодавець повинен надати приміщення для переодягання в окремих або спеціально відведених кімнатах. Умови, яким вони повинні відповідати, такі:

- Вони повинні бути сухими і, по можливості, освітлюватись денним світлом. Вони можуть розташовуватися в підвалах або льохах за умови достатньої ізоляції зовнішніх стін і підлоги для захисту приміщень від вологи і надмірних тепловтрат і забезпечення умов для евакуації людей з цих приміщень;
- Має відбуватися не менше чотирьох змін повітря за годину, а в гардеробних з вікнами, які можуть бути відкриті не більше ніж на 10 працівників, зміна повітря повинна відбуватися не рідше двох разів на годину;

- Механічна вентиляція повинна бути передбачена в роздягальнях не менше ніж для 25 працівників [30];
- Роздягальні повинні забезпечувати сидячі місця не менше ніж для 50% працівників найбільш численної зміни.

Ширина проходів між двома рядами шафок і основними комунікаційними проходами повинна бути не менше 1,5 м. Ширина проходів між рядами шафок і стіною повинна бути не менше 1,1 м [30].

3.4 Висновки до третього розділу

Важливим фактором, що впливає на умови праці, є те, на скільки роботодавець і працівник піклуються про гігієну та безпеку праці. Перший елемент тісно пов'язаний з профілактикою професійних захворювань. Основна задача гігієни праці - контролювати ступінь загроз у процесі праці та прагнути до їх мінімізації у всіх можливих випадках. Для того, щоб це стало можливим, необхідно визначити всі важливі фактори, визначити ступінь їх інтенсивності та вказати дії, що ведуть до обмеження ризику, обумовленого ними [31].

На даний момент, тільки 1% проведених досліджень пов'язані з питаннями охорони праці. Економічні аналізи, пов'язані з невідповідними умовами праці та їх наслідками, проводилися протягом багатьох років [29].

У Європейському Союзі загальні витрати на невідповідні умови праці оцінюються в 2,8% від валового внутрішнього продукту (ВВП) [29].

Серед пріоритетних напрямків досліджень в області охорони праці є дослідження з розвитку культури безпеки шляхом впровадження систем управління охороною праці, посилення соціальної ролі, відповідальності бізнесу та поширення наукових відкриттів і прикладів передової практики.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було вирішено проблему несвоєчасного отримання повідомлень на платформі дистанційного навчання

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– Подано інформацію про можливість дистанційного навчання для студентів ТНТУ;

– Розглянуто можливості платформи дистанційного навчання, її зручності;

– Висвітлено проблему несвоєчасного отримання сповіщень про повідомлення на платформі дистанційного навчання та швидкої відповіді на них;

– Проаналізовано можливість вирішення проблеми підключивши можливість отримувати повідомлення та відповідати на них у месенджері «Телеграм».

В другому розділі кваліфікаційної роботи:

– Розроблено систему сповіщення для дистанційного навчання;

– Запропоновано створити чат бот що буде періодично перевіряти наявність нових повідомлень на платформі дистанційного навчання та можливість відповідати на них;

– Спроєктовано застосунок що складається із трьох модулів для вирішення проблеми несвоєчасного отримання повідомлень.

У розділі «Безпека життєдіяльності, основи хорони праці» Висвітлено питання про охорону праці, а саме:

– Стихійні лиха та їх класифікація;

– Розрахунок місцевого освітлення для роботи з комп’ютером;

– Санітарно-гігієнічні вимоги до умов праці.

ПЕРЕЛІК ДЖЕРЕЛ

1. Офіційні документи сервера ЕН [Інтернет ресурс] - Режим доступу https://dl.tntu.edu.ua/showpage.php?id=3 - Дата доступу 10.03.2021 - Назва з Екрану;

2. Офіційні документи сервера ЕН [Інтернет ресурс] - Режим доступу https://dl.tntu.edu.ua/showpage.php?id=4 - Дата доступу 11.03.2021 - Назва з Екрану;

3. ПЕРШІ КРОКИ В СИСТЕМІ ЕЛЕКТРОННОГО НАВЧАННЯ ATUTOR [Інтернет ресурс] - Режим доступу https://dl.tntu.edu.ua/downloads/Student-first-steps.pdf - Дата доступу 13.03.2021 - Назва з Екрану;

4. Головний сайт «Телеграм» [Інтернет ресурс] - Режим доступу https://www.telegram.org/ - Дата доступу 20.03.2021 - Назва з Екрану;

5. Відповіді на часті питання месенджера «Телеграм» [Інтернет ресурс] - Режим доступу https://telegram.org/faq - Дата доступу 25.03.2021 - Назва з Екрану;

6. Документація API «Телеграм» [Інтернет ресурс] - Режим доступу https://core.telegram.org/ - Дата доступу 01.04.2021 - Назва з Екрану;

7. HTTP протокол [Інтернет ресурс] - Режим доступу https://developer.mozilla.org/en-US/docs/Web/HTTP - Дата доступу 10.04.2021 - Назва з Екрану;

8. Welcome to the Visual Studio IDE [Інтернет ресурс] - Режим доступу https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019 - Дата доступу 15.04.2021 - Назва з Екрану;

9. What is .NET? [Інтернет ресурс] - Режим доступу https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet - Дата доступу 16.04.2021 - Назва з Екрану;

10. C# [Інтернет ресурс] - Режим доступу https://dotnet.microsoft.com/languages/csharp - Дата доступу 18.04.2021 - Назва з Екрану;

11. Visual Studio 2019 [Інтернет ресурс] - Режим доступу https://visualstudio.microsoft.com/vs/ - Дата доступу 20.04.2021 - Назва з Екрану;

12. The Selenium project and tools [Інтернет ресурс] - Режим доступу https://www.selenium.dev/documentation/en/introduction/the_selenium_project_and_tools/ - Дата доступу 25.04.2021 - Назва з Екрану;

13. ΛngleSharp makes .NET ❤ HTML5 [Інтернет ресурс] - Режим доступу https://anglesharp.github.io/ - Дата доступу 28.04.2021 - Назва з Екрану;

14. Types Of Software Testing: Different Testing Types With Details [Інтернет ресурс] - Режим доступу https://www.softwaretestinghelp.com/types-of-software-testing/ - Дата доступу 30.04.2021 - Назва з Екрану;

15. Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения;

16. Vaskaran Sarcar. Design Patterns in C# 2018. Паттерн «Наглядач»;

17. HTML розмітка [Інтернет ресурс] - Режим доступу http://htmlbook.ru/samhtml/vvedenie-v-html - Дата доступу 10.05.2021 - Назва з Екрану;

18. Christian Nagel. Professional C# 7 and .NET Core 2.0;

19. Beautiful people at Stack Overflow. CSS Notes For Professionals 2019;

20. C# Coding Conventions [Інтернет ресурс] - Режим доступу https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions - Дата доступу 12.05.2021 - Назва з Екрану;

21. Mark J. Price C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET;

22. Jamie Chan. C#: Learn C# in One Day and Learn It Well. C# for Beginners with Hands-on Project;

23. Timer Класс [Інтернет ресурс] - Режим доступу https://docs.microsoft.com/ru-ru/dotnet/api/system.timers.timer?view=net-5.0 - Дата доступу 16.05.2021 - Назва з Екрану;

24. Andrew Stellman, Jennifer Greene. Head First C#: A Learner's Guide to Real-World Programming with C# and .NET Core 4th Edition;

25. Joseph Albahari, Ben Albahari. C# 8.0 Pocket Reference: Instant Help for C# 8.0 Programmers 1st Edition;

26. Документація бібліотеки для чат бота Telegram [Інтернет ресурс] - Режим доступу https://github.com/TelegramBots/Telegram.Bot - Дата доступу 20.05.2021 - Назва з Екрану;

27. Alex Davies. Async in C# 5.0: Unleash the Power of Async 1st Edition;

28. von Bill Wagner. Effective C#;

29. Генрік Свенсен. Кінець світу близько: про стихійні лиха та їх вплив на суспільство;

30. Алексанян А.Н. Охорона праці. Підручник для вищих навчальних закладів;

31. iMaikinG. А. Здоров'я та безпека на робочому місці. Підручник для вищих навчальних закладів;

32. Славукіа. К. Безпека на роботі в сільському господарстві. Підручник для вищих навчальних закладів;

33. Класифікація стихійних лих [Інтернет ресурс] - Режим доступу https://ua-books.com.ua/prezentaciyi/5588-styhijni-lyha-meteorologichnogo-harakteru-19133 - Дата доступу 28.05.2021 - Назва з Екрану.

ДОДАТКИ

Додаток А

Лістинг файлів програмного коду

Лістинг 1.1 – Класс Crawler

```
using System;
using System.Collections.Generic;
using System.Linq;
using AngleSharp;
using AngleSharp.Html.Parser;
using Microsoft.Edge.SeleniumTools;
using OpenQA.Selenium;
using TNTUNotifications.Models.Mail;
using TNTUNotifications.Crawler.Constants;
using TNTUNotifications.Crawler.Extension;
using AngleSharp.Html.Dom;
using TNTUNotifications.Crawler.Events;
using System.Timers;

namespace TNTUNotifications.Crawler
{
    public class Crawler : IDisposable
    {
        #region Properties
        private EdgeDriver _driver;
        private Timer _timer;

        private readonly string Login;
        private readonly string Password;
        private const int TimerOffsetMinutes = 1;

        #endregion

        #region Events

        public delegate void NewMailesEventHandler(object sender,
NewMailesEventArgs e);
        public event NewMailesEventHandler OnCheckNewMails;

        #endregion

        public Crawler(string login, string password)
        {
            Login = login;
            Password = password;

            var options = new EdgeOptions();
            options.UseChromium = true;
            _driver = new EdgeDriver(options);

            SetUpTimer();
        }

        #region Public
        public IEnumerable<MailItem> GetNewMailes()
        {
            lock (_driver)
            {
                _driver.Login(Login, Password);

                int newMailes = CheckNewMailes();
```

```
                if (newMailes == 0)
                    return Array.Empty<MailItem>();

                _driver.GoToMailBox();
                var document = GetMailesHtml();
                var mailesNew = ParseMailesNew(document).ToList();

                _driver.Logout();

                return mailesNew;
            }
        }

        public IEnumerable<MailItem> GetAllMailes()
        {
            lock (_driver)
            {
                _driver.Login(Login, Password);

                _driver.GoToMailBox();
                var document = GetMailesHtml();
                var mailes = ParseMailesAll(document).ToList();

                _driver.Logout();

                return mailes;
            }
        }

        public string GetMailBody(string mailId)
        {
            lock (_driver)
            {
                _driver.Login(Login, Password);

                _driver.GoToMailItem(mailId);
                var                         bodyElem                                =
_driver.FindElementByCssSelector(Css.MailItemBody);
                var message = bodyElem.Text;

                _driver.Logout();

                return message;
            }
        }

        public void AnswerMail(string mailId, string message)
        {
            lock (_driver)
            {
                _driver.GoToMailItem(mailId);

_driver.FindElementByCssSelector(Css.AnswerMailItemLink).Click();
                var                         mailBody                                =
_driver.FindElementByCssSelector(Css.AnswerMailItemBody);

                mailBody.Click();
                var count = mailBody.Text.Count(x => x == '\n');
                for (int i = 0; i < count; i++)
                {
                    mailBody.SendKeys(Keys.ArrowUp);
                }
                mailBody.SendKeys(message);
```

```
                _driver.FindElementByCssSelector(Css.LoginButton).Click();
            }
        }

        public void TryLogin()
        {
            lock (_driver)
            {
                _driver.Login(Login, Password);
                _driver.Logout();
            }
        }

        public void Dispose()
        {
            lock (_driver)
            {
                _driver.Close();
                _driver.Dispose();
            }
        }

        #endregion

        #region Private

        // go to MailBox page first
        private IHtmlDocument GetMailesHtml()
        {
            IWebElement elem = null;
            lock (_driver)
            {
                elem = _driver.FindElementByCssSelector(Css.MailAnswerBody);
            }
            string mailesHtml = elem.GetAttribute("outerHTML");

            //Use the default configuration for AngleSharp
            var config = Configuration.Default;
            var context = BrowsingContext.New(config);
            var parser = context.GetService<IHtmlParser>();

            var document = parser.ParseDocument(mailesHtml);
            return document;
        }

        private IEnumerable<MailItem> ParseMailesAll(IHtmlDocument document)
        {
            var                              mailes                                  =
document.QuerySelectorAll(Css.MailBoxRow).ToMailItemList();
            return mailes;
        }

        private IEnumerable<MailItem> ParseMailesNew(IHtmlDocument document)
        {
            var                              mailes                                  =
document.QuerySelectorAll(Css.MailBoxRow).ToMailNewItemList();
            return mailes;
        }

        private int CheckNewMailes()
        {
            IReadOnlyCollection<IWebElement> alerts = null;
            lock(_driver)
```

```
                alerts                                                    =
_driver.FindElementsByCssSelector(Css.NewMailesAlert);

            if (alerts.Count == 0 )
                return 0;

            string count = alerts.First().Text;
            var newMailes = int.Parse(count);
            return newMailes;
        }

        private void SetUpTimer()
        {
            _timer                              =                          new
Timer(TimeSpan.FromMinutes(TimerOffsetMinutes).TotalMilliseconds);
            _timer.Elapsed += OnElapsed;
            _timer.Start();

            void OnElapsed(object sender, ElapsedEventArgs e)
            {
                var newMailes = GetNewMailes().ToList();
                OnCheckNewMails?.Invoke(sender,                           new
NewMailesEventArgs(newMailes));
            }
        }

        #endregion
    }
}
```

Лістинг 1.2 – Клас Css

```
namespace TNTUNotifications.Crawler.Constants
{
    public class Css
    {
        // Login page
        public const string LoginInput = "input#login";
        public const string PassInput = "input#pass";
        public const string LoginButton = ".btn-primary";

        // Logout
        public const string LogoutButton = @"[href='\/logout\.php']";

        // mailbox
        public const string MailboxButton = @"[href='\/inbox\/index\.php']";
        public const string MailAnswerBody = "table.data";
        public const string MailItemBody = ".forum-post-body > p";
        public const string MailReseivers = "ul#to2_feed > li";
        public const string NewMailesAlert = "span.badge";
        public const string AnswerMailItemLink = ".forum-post-ctrl > a:nth-
    of-type(1)";
```

```
        public const string AnswerMailItemBody = ".row textarea";
        public const string MailBoxRow = "tbody tr";
        public const string SendMessageButton =
@"[href='\/inbox\/send_message\.php']";
    }
}
```

Лістинг 1.3 – Клас Pages

```
namespace TNTUNotifications.Crawler.Constants
{
    public class Pages
    {
        public const string LoginPage = "https://dl.tntu.edu.ua/login.php";
        public const string MailItemPage =
"https://dl.tntu.edu.ua/inbox/index.php?view=";
    }
}
```

Лістинг 1.4 – Клас NewMailesEventArgs

```
using System;
using System.Collections.Generic;
using TNTUNotifications.Models.Mail;

namespace TNTUNotifications.Crawler.Events
{
    public class NewMailesEventArgs : EventArgs
    {
        public List<MailItem> MailItems { get; set; }
        public NewMailesEventArgs(List<MailItem> mailitems)
        {
            MailItems = mailitems;
        }
    }
}
```

Лістинг 1.5 – Клас MailExtensions

```
using System;
using System.Collections.Generic;
using System.Linq;
using AngleSharp.Dom;
```

```
using OpenQA.Selenium;
using TNTUNotifications.Models.Mail;

namespace TNTUNotifications.Crawler.Extension
{
    public static class MailExtensions
    {
        public static IEnumerable<MailItem> ToMailItemList(this
IEnumerable<IElement> elements)
        {
            return elements.Select(x => x.ToMailItem());
        }

        private const string Sender = "tr td:nth-of-type(3)";
        private const string Subject = "tr a";
        private const string Date = "tr td:nth-of-type(5)";
        private const string IsNew = "tbody tr td:nth-of-type(2) span";

        private const string Href = "href";
        public static MailItem ToMailItem(this IElement element)
        {
            var sender = element.QuerySelector(Sender).TextContent;
            var subjectElem = element.QuerySelector(Subject);
            var subject = subjectElem.TextContent;
            var viewId = subjectElem.GetAttribute(Href).Substring(22); //
skip text /inbox/index.php?view=XXXXXX
            var dateUkr = element.QuerySelector(Date).TextContent;
            var date =
DateTime.Parse(dateUkr.Substring(dateUkr.IndexOf(','))); // remove day name
in Ukraine because of parse error: П’ятниця, 14 травня 2021, 10:30
            var isNewElem = element.QuerySelector(IsNew);
            var isNew = isNewElem != null;

            return new MailItem
            {
                Id = viewId,
                Subject = subject,
                Sender = sender,
                Date = date,
                IsNew = isNew
            };
        }
```

```
        public     static     IEnumerable<MailItem>     ToMailNewItemList(this
IEnumerable<IElement> elements)
        {
            const int SkipUrl = 22;
            const char UkrDaySeparator = ',';

            foreach (var element in elements)
            {
                var isNewElem = element.QuerySelector(IsNew);
                var isNew = isNewElem != null;

                if (!isNew)
                    continue;

                var sender = element.QuerySelector(Sender).TextContent;
                var subjectElem = element.QuerySelector(Subject);
                var subject = subjectElem.TextContent;
                var                          viewId                          =
subjectElem.GetAttribute(Href).Substring(SkipUrl);      //      skip      text
/inbox/index.php?view=XXXXXX
                var dateUkr = element.QuerySelector(Date).TextContent;
                var                           date                           =
DateTime.Parse(dateUkr.Substring(dateUkr.IndexOf(UkrDaySeparator)));        //
remove day name in Ukraine because of parse error: П’ятниця, 14 травня 2021,
10:30

                yield return new MailItem
                {
                    Id = viewId,
                    Subject = subject,
                    Sender = sender,
                    Date = date,
                    IsNew = isNew
                };
            }
        }

        public      static      IEnumerable<string>      LiToStringList(this
IEnumerable<IWebElement> elements)
        {
            return elements.Select(x => x.LiToString());
```

```
            }

            public static string LiToString(this IWebElement li)
            {
                var test = li.GetAttribute("innerText");
                return test;
            }
        }
}
```

Лістинг 1.6 – Клас NavigateExtensions

```
using Microsoft.Edge.SeleniumTools;
using TNTUNotifications.Crawler.Constants;

namespace TNTUNotifications.Crawler.Extension
{
    public static class NavigateExtensions
    {
        public static void Login(this EdgeDriver driver, string login, string
password)
        {
            driver.Navigate().GoToUrl(Pages.LoginPage);
            var loginField = driver.FindElementByCssSelector(Css.LoginInput);
            loginField.Click();
            loginField.SendKeys(login);

            var passField = driver.FindElementByCssSelector(Css.PassInput);
            passField.Click();
            passField.SendKeys(password);
            driver.FindElementByCssSelector(Css.LoginButton).Click();
        }

        public static void Logout(this EdgeDriver driver)
        {
            driver.FindElementByCssSelector(Css.LogoutButton).Click();
        }
        public static void GoToMailBox(this EdgeDriver driver)
        {
            driver.FindElementByCssSelector(Css.MailboxButton).Click();
        }
```

```
        public static void GoToMailItem(this EdgeDriver driver, string
    mailId)
        {
            driver.Navigate().GoToUrl(Pages.MailItemPage + mailId);
        }

        public static void GoToSendMessage(this EdgeDriver driver)
        {
            driver.GoToMailBox();
            driver.FindElementByCssSelector(Css.SendMessageButton).Click();
        }
    }
}
```

Лістинг 1.7 – Клас MailItem

```
using System;

namespace TNTUNotifications.Models.Mail
{
    public class MailItem
    {
        public string Id { get; set; }
        public string Sender { get; set; }
        public string Subject { get; set; }
        public DateTime Date { get; set; }
        public bool IsNew { get; set; }
    }
}
```

Лістинг 1.8 – Клас TelegramWrapper

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Telegram.Bot;
using Telegram.Bot.Args;
using Telegram.Bot.Types;
using Telegram.Bot.Types.Enums;
using Telegram.Bot.Types.ReplyMarkups;
using TNTUNotifications.Bot.Extension;
using TNTUNotifications.Models.Mail;

namespace TNTUNotifications.Bot
{
    public class TelegramWrapper
    {
        #region Properties

        private TelegramBotClient _bot;
```

```
        private readonly Dictionary<long, Crawler.Crawler> _crawlers =
new Dictionary<long, Crawler.Crawler>();

        #endregion

        #region Public

        public async Task SetUp()
        {
            _bot = new TelegramBotClient(Constants.BotToken);

            var me = await _bot.GetMeAsync();
            Console.Title = me.Username;

            _bot.OnMessage += BotOnMessageReceived;
            _bot.OnCallbackQuery += BotOnCallbackQueryReceived;
            _bot.OnReceiveError += BotOnReceiveError;

            _bot.StartReceiving(Array.Empty<UpdateType>());
            Console.WriteLine($"Start listening for @{me.Username}");

            Console.ReadLine();
            _bot.StopReceiving();
        }

        #endregion

        #region Event Handlers

        private async void BotOnMessageReceived(object sender,
MessageEventArgs messageEventArgs)
        {
            var message = messageEventArgs.Message;
            if (message == null || message.Type != MessageType.Text)
                return;

            switch (message.Text.Split(' ').First())
            {
                case "/login": // Enter your credentials to
https://dl.tntu.edu.ua/ after /login command. Format: login password
                    await CheckCredentials(message);
                    break;
                case "/checknewmailes":
                    await CheckMailes(message);
                    break;
                case "/answermail":
                    await AnswerMail(message);
                    break;
                case "/getmailes":
                    await GetAllMailes(message);
                    break;
            }
        }

        private async void BotOnCallbackQueryReceived(object sender,
CallbackQueryEventArgs callbackQueryEventArgs)
        {
            var callbackQuery = callbackQueryEventArgs.CallbackQuery;
            var                    arguments                    =
callbackQuery.Data.Split(Constants.Separator);
            var command = arguments[0];
            var mailId = arguments[1];
```

```
                switch (command)
                {
                    case Constants.GetMessageBody:
                        await SendMailBody(callbackQuery, mailId);
                        break;
                    case Constants.Answer:
                        await SendAnswerInstructions(callbackQuery, mailId);
                        break;
                }
            }

            private       void       BotOnReceiveError(object       sender,
ReceiveErrorEventArgs receiveErrorEventArgs)
            {
                Console.WriteLine("Received error: {0} — {1}",
                    receiveErrorEventArgs.ApiRequestException.ErrorCode,
                    receiveErrorEventArgs.ApiRequestException.Message
                );
            }

            #endregion

            #region Private

            private async Task CheckCredentials(Message message)
            {
                // add user to dictionary
                var userId = message.Chat.Id;
                var creds = message.Text.Split(' ');
                var login = creds[1];
                var password = creds[2];

                var crawler = new Crawler.Crawler(login, password);
                crawler.OnCheckNewMails     +=     (sender,     e)     =>
AnswerCheckNewMailes(userId, e.MailItems).GetAwaiter().GetResult();

                _crawlers[userId] = crawler;

                await _bot.SendTextMessageAsync(
                        chatId: message.Chat.Id,
                        text: $"Trying to login"
                    );

                crawler.TryLogin();

                await _bot.SendTextMessageAsync(
                        chatId: message.Chat.Id,
                        text: $"Success!"
                    );
            }

            private async Task CheckMailes(Message message)
            {
                var userId = message.Chat.Id;
                var mailes = _crawlers[userId].GetNewMailes().ToList();

                await AnswerCheckNewMailes(userId, mailes);
            }

            private async Task AnswerMail(Message message)
            {
                var mailId = message.Text.Split(' ')[1];
```

```
                var                         mailBody                        =
message.Text.Substring(message.Text.IndexOf('\n')+1);

                var userId = message.Chat.Id;
                _crawlers[userId].AnswerMail(mailId, mailBody);

                await _bot.SendTextMessageAsync(
                        chatId: message.Chat.Id,
                        text: "Answered!"
                        );
            }

            private async Task GetAllMailes(Message message)
            {
                var userId = message.Chat.Id;
                var mailes = _crawlers[userId].GetAllMailes().ToList();

                if (mailes.Count == 0)
                {
                    await _bot.SendTextMessageAsync(
                        chatId: message.Chat.Id,
                        text: $"There is no messages"
                    );
                }
                else
                {
                    foreach (var mail in mailes)
                    {
                        var                  answerButton                   =
InlineKeyboardButton.WithCallbackData("Answer",       Constants.Answer       +
Constants.Separator + mail.Id);
                        var                  getBodyButton                  =
InlineKeyboardButton.WithCallbackData("Body",    Constants.GetMessageBody    +
Constants.Separator + mail.Id);

                        await _bot.SendTextMessageAsync(
                        chatId: message.Chat.Id,
                        text: mail.ToMailString(),
                        replyMarkup:    new    InlineKeyboardMarkup(new[]    {
answerButton, getBodyButton })
                        );
                    }
                }
            }

            private    async    Task    AnswerCheckNewMailes(long    chatId,
List<MailItem> mailes)
            {
                if (mailes.Count == 0)
                {
                    await _bot.SendTextMessageAsync(
                        chatId: chatId,
                        disableNotification: true,
                        text: $"There is no messages"
                    );
                }
                else
                {
                    await _bot.SendTextMessageAsync(
                        chatId: chatId,
                        text: $"New messages:"
                    );
                    foreach (var mail in mailes)
```

```
                    {
                        var                    answerButton                    =
InlineKeyboardButton.WithCallbackData("Answer",       Constants.Answer       +
Constants.Separator + mail.Id);
                        var                   getBodyButton                    =
InlineKeyboardButton.WithCallbackData("Body",    Constants.GetMessageBody    +
Constants.Separator + mail.Id);

                        await _bot.SendTextMessageAsync(
                        chatId: chatId,
                        text: mail.ToMailString(),
                        replyMarkup:   new   InlineKeyboardMarkup(new[]   {
answerButton, getBodyButton })
                        );
                    }
                }
            }

            private  async  Task  SendMailBody(CallbackQuery  callbackQuery,
string mailId)
            {
                var userId = callbackQuery.Message.Chat.Id;
                var mailBody = _crawlers[userId].GetMailBody(mailId);

                await _bot.AnswerCallbackQueryAsync(
                    callbackQueryId: callbackQuery.Id,
                    text: $"Get body done!"
                );

                await _bot.SendTextMessageAsync(
                    chatId: callbackQuery.Message.Chat.Id,
                    text: $"Message body: \n{mailBody}"
                );
            }

            private    async    Task    SendAnswerInstructions(CallbackQuery
callbackQuery, string mailId)
            {
                var chatId = callbackQuery.Message.Chat.Id;

                await _bot.AnswerCallbackQueryAsync(
                    callbackQueryId: callbackQuery.Id
                );

                await _bot.SendTextMessageAsync(
                    chatId: chatId,
                    text: $"Send command in format:"
                );

                await _bot.SendTextMessageAsync(
                    chatId: chatId,
                    text: $"/answermail {mailId} \nmessage"
                );
            }

            #endregion
        }
    }
```

Лістинг 1.9 – Клас Constants

```
namespace TNTUNotifications.Bot
{
    public class Constants
    {
        public         const         string         BotToken         =
"1060632510:AAFSqU8anAR6qGRtCAh1DqEb7cfVP1PsCOw";

        public const string Separator = "|";
        public const string Answer = "Answer";
        public const string GetMessageBody = "GetMessageBody";
    }
}
```

Лістинг 1.10 – Клас MailExtensions

```
using TNTUNotifications.Models.Mail;

namespace TNTUNotifications.Bot.Extension
{
    public static class MailExtensions
    {
        public static string ToMailString(this MailItem mailItem)
        {
            return $"Subject: {mailItem.Subject}" + "\n" +
                $"Sender: {mailItem.Sender}" + "\n" +
                $"Date: {mailItem.Date.ToString("HH:mm dd.MM.yyyy")}" +
"\n" +
                $"{(mailItem.IsNew ? "New message" : "")}";
        }
    }
}
```