МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_____ проф. Приходько С.Б.

«___» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка програмного забезпечення для надання розкладу занять студентам НУК

Виконав: студент групи 4151

_____ Шебалін Н.О.

(підпис, ПІБ)

Керівник роботи:

Завідувач кафедри ПЗАС, проф.

(посада, науковий ступень вчене звання)

_____ Приходько С.Б.

(підпис, ПІБ)

Миколаїв – 2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«___» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Шебаліну Нікіти Олександровичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: розробка програмного забезпечення для надання розкладу занять студентам НУК

Керівник роботи завідувач кафедри ПЗАС, проф. Приходько Сергій Борисович

Затверджені наказом ректора №257-уч від «17» березня 2023 р.

2. Термін подання роботи: 29.05.2023 р.

3. Вихідні дані по роботі: _____

_____

4. Перелік питань, що належать до розробки (найменування розділів) _____

Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів Титульний слайд, Мета та завдання кваліфікаційної роботи, Аналіз вимог до ПЗ, Архітектура ПЗ, Діаграма варіантів використання, Концептуальна модель даних, Логічна модель даних, Технічний проект ПЗ, Вибір інструментів розробки, Реалізація та Тестування ПЗ, Результати розробки, Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 20.03.2023 р. _____

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	24.03.2023	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	31.03.2023	ПП*
3.	Аналіз вимог до ПЗ	07.04.2023	ПП*
4.	Розробка архітектури ПЗ	21.04.2023	
5.	Розробка проекту ПЗ	05.05.2023	
6.	Реалізація та тестування ПЗ	12.05.2023	
7.	Підготовка розділу Результати розробки ПЗ	17.05.2023	
8.	Підготовка розділу з охорони праці	19.05.2023	ПП*
9.	Підготовка розділу ВИСНОВКИ	24.05.2023	
10.	Оформлення списку використаних джерел та додатків	26.05.2023	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	29.05.2023	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
12.	Підготовка презентації та доповіді	02.06.2023	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	05.06.2023	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	12.06.2023	

* - за результатами переддипломної практики (ПП), яка була з 01.02.2023 до 19.03.2023 р.

Студент _____ Шебалін Н.О.
(підпис) (ПІБ)

Керівник роботи _____ Приходько С.Б.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена вирішенню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці програмного забезпечення для надання розкладу занять студентам НУК.

Об’єктом кваліфікаційної роботи є процес розробки програмного забезпечення для надання розкладу занять студентам НУК. Проведено аналіз предметної галузі та проаналізовано існуючі програмні продукти та методи розробки веб-сервісів та Telegram ботів. Здійснено постановку задачі на розробку веб-сервісу для керування розкладом та розробку Telegram бота для надання розкладу занять та складено технічне завдання. Було розроблено програмне забезпечення для надання розкладу занять студентам НУК та представлено частину коду. Проведено опис аспектів охорони праці при роботі з екранними пристроями.

Розробку програмного забезпечення було виконано за допомогою середовища розробки Visual Studio Code, мов програмування TypeScript та Dart, фреймворку Flutter та СУБД Firebase.

Кваліфікаційна робота викладена на 121 сторінках друкованого тексту, містить 41 рисунок, 14 таблиць, 5 додатків та список використаних джерел із 15 найменувань.

ABSTRACT

The qualification work is devoted to solving the practical problem of software engineering, characterized by complexity and uncertainty of conditions, namely the development of software for providing class schedules to students of NUOS.

The object of the qualification work is the process of developing software for providing class schedules to students of the NUOS. The subject area was analyzed and existing software products and methods for developing web services and Telegram bots were analyzed. The task of developing a web service for managing the schedule and developing a Telegram bot for providing class schedules was set and the terms of reference were drawn up. Software for providing class schedules to students of the NUOS was developed and part of the code was presented. The aspects of labor protection when working with screen devices were described.

The software was developed using the Visual Studio Code development environment, TypeScript and Dart programming languages, the Flutter framework, and the Firebase database.

The qualification work is presented on 121 pages of printed text, contains 41 figures, 14 tables, 5 appendices and a list of references of 15 titles.

ЗМІСТ

ВСТУП 7

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ НАДАННЯ РОЗКЛАДУ ЗАНЯТЬ СТУДЕНТАМ НУК 10

1.1 Аналіз практичної задачі інженерії програмного забезпечення 10

1.2 Аналіз існуючих аналогів планування розкладу та обґрунтування необхідності розробки програмного забезпечення для надання розкладу занять студентам НУК 14

1.3 Постановка задачі на розробку програмного забезпечення для надання розкладу занять студентам НУК 20

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ НАДАННЯ РОЗКЛАДУ ЗАНЯТЬ СТУДЕНТАМ НУК 24

2.1 Побудова моделі варіантів використання програмного забезпечення для надання розкладу занять студентам НУК 24

2.2 Специфікації варіантів використання програмного забезпечення для надання розкладу занять студентам НУК 27

2.3 Розробка архітектури програмного забезпечення для надання розкладу занять студентам НУК 34

2.4 Технічний проект програмного забезпечення для надання розкладу занять студентам НУК 48

2.5 Реалізація програмного забезпечення для надання розкладу занять студентам НУК 61

2.6 Тестування програмного забезпечення для надання розкладу занять студентам НУК 69

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ НАДАННЯ РОЗКЛАДУ ЗАНЯТЬ СТУДЕНТАМ НУК 71

4 ОХОРОНА ПРАЦІ 80

4.1 Аналіз шкідливих та небезпечних факторів під час роботи з екранними пристроями 81

4.2 Методи нейтралізації або зменшення впливу негативних факторів під час роботи з екранними пристроями 84

ВИСНОВКИ 86

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 87

ДОДАТОК А – Технічне завдання на розробку програмного забезпечення для надання розкладу занять студентам НУК 89

ДОДАТОК Б – Вихідні тексти програмних модулів 95

ДОДАТОК В – Опис програмного забезпечення 107

ДОДАТОК Г – Інструкція користувача 114

ДОДАТОК Д – Програма та методика випробувань програмного забезпечення 117

ВСТУП

Розробка ефективних програмних додатків стала надзвичайно важливою в сучасну цифрову епоху, коли технології відіграють невід'ємну частину нашого життя, оптимізуючи процеси та підвищуючи ефективність. Оскільки інженерія програмного забезпечення є динамічною та складною сферою, інновації необхідні для задоволення зростаючих потреб різних галузей. Розробка сучасних програмних систем передбачає вирішення таких проблем, як складність, невизначеність і зміна вимог користувачів. Практична задача інженерії програмного забезпечення, яка розглядається у кваліфікаційної роботі, стосується розробки програмного забезпечення для надання розкладу занять студентам НУК. Ця задача характеризується складністю та невизначеністю умов, оскільки вимагає розробки ефективного та надійного рішення для покращення процесу планування та управління розкладом занять.

В сучасних університетах існує багато проблем з традиційними методами створення розкладів. Потреба в розробці такого програмного забезпечення виникає через зростаючий попит на більш ефективні та надійні способи надання студентам інформації про розклад. Зараз університети використовують традиційні методи, такі як дошки оголошень, електронні листи та ручні системи обміну повідомленнями, які часто повільні та схильні до помилок. Тому існує потреба в розробці сучасного рішення, яке використовує передові технології для створення більш ефективної та надійної системи планування. Розробка програмного забезпечення для надання розкладу занять студентам може здійснюватися шляхом застосуванням фреймворків, тестування, використання баз даних, хмарних технологій, залучення користувачів та постійного вдосконалення. Таким чином, для вирішення даної задачі потрібно розробити Telegram-бота на мові TypeScript, який буде основним інтерфейсом доступу до розкладу та сповіщень [1, 2]. Розклади будуть надсилатись підписаним користувачам у визначений час за допомогою сервісу Cronjob [3]. Також слід

створити базу даних на Firebase для зберігання розкладу, груп та інформації про користувачів [4]. Додатково, потрібно розробити веб-сервіс планування для адміністраторів та модераторів, використовуючи фреймворк Flutter для керування розкладом занять [5].

Отже, мета кваліфікаційної роботи полягає в вирішенні практичної задачі інженерії програмного забезпечення, а саме в розробці програмного забезпечення для надання розкладу занять студентам НУК. Об’єктом цієї роботи є процес розробки програмного забезпечення для надання розкладу занять студентам НУК. Для досягнення мети кваліфікаційної роботи треба вирішити наступні завдання:

– Проаналізувати вимоги та потреби студентів, викладачів та адміністраторів НУК щодо розкладу занять.

– Визначити основні функціональні можливості та функції, які необхідно включити в програмне забезпечення.

– Здійснити огляд існуючих аналогів систем управління розкладом та визначити їх переваги і недоліки.

– Розробити архітектуру та функціональні вимоги до програмного забезпечення на основі отриманих вимог і аналізу існуючих рішень.

– Визначити необхідні технології та інструменти для реалізації програмного забезпечення.

– Розробити та впровадити Telegram-бота з використанням мови TypeScript. Цей бот буде служити основним інтерфейсом для користувачів, які зможуть отримувати доступ до розкладу занять та отримувати сповіщення про наближення пар через сервіс Cronjob відповідно до заданого часу.

– Розробити базу даних для зберігання розкладу груп, занять та інформації про користувачів. Для цього буде використано Firebase, який надає різноманітні функції, такі як база даних, автентифікація та послуги хостингу. Всі ці функції будуть використовуватись під час розробки проекту.

– Розробити веб-сервіс планування для адміністраторів та модераторів для керування розкладом та групами. Для створення цього сервісу буде використано фреймворк Flutter, який дозволяє розробляти високопродуктивні та високоякісні програми для веб, iOS, Android та інших платформ з використанням однієї кодової бази.

– Провести тестування розроблених модулів.

Ця робота спрямована на вирішення практичної задачі інженерії програмного забезпечення яка пов’язана з розробкою програми для надання розкладу занять студентам НУК. В рамках кваліфікаційної роботи буде проведено детальний аналіз та розробка програмного забезпечення з використанням різноманітних інструментів, зокрема TypeScript, Flutter, Dart, Telegraf, Firebase та Cronjob. Використання цих інструментів дозволяє забезпечити ефективну та зручну функціональність для користувачів шляхом використання Telegram-бота та веб-сервісу для адміністраторів та модераторів. Також, буде проведено тестування для забезпечення функціональності розробленого програмного забезпечення. Результати тестування та деталі реалізації будуть задокументовані та представлені в наступних розділах кваліфікаційної роботи. Робота має на меті показати ефективність та придатність розробленого програмного забезпечення для вирішення конкретної задачі надання розкладу занять студентам НУК.

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ НАДАННЯ РОЗКЛАДУ ЗАНЯТЬ СТУДЕНТАМ НУК

1.1 Аналіз практичної задачі інженерії програмного забезпечення

Аналіз практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці програмного забезпечення для надання розкладу занять студентам НУК, включає детальне вивчення різних аспектів та вимог, пов'язаних з розробкою та експлуатацією програмного забезпечення.

Функціональний аналіз має пріоритет при визначенні основних функцій і можливостей програмного забезпечення. У контексті програмного забезпечення для надання розкладів студентам НУК стає необхідним надати пріоритет зручному та інтуїтивно зрозумілому інтерфейсу, який сприяє безперешкодному управлінню та перегляду розкладів. Це включає в себе вибір групи, перегляд розкладу на кожен день тижня та можливість підписатися на сповіщення про майбутні заняття. Викладачі повинні мати можливість додавати, редагувати та видаляти заняття та групи в розкладі. Це дозволяє оновлювати розклад і забезпечувати його точність. Інтерфейс програми має бути привабливим і забезпечувати комфортну роботу користувача.

З іншого боку, нефункціональний аналіз зміщує акцент на продуктивність, безпеку та стабільність програми. Програмне забезпечення повинно демонструвати похвальну ефективність і швидкість реагування на запити користувачів, забезпечуючи тим самим безперебійне використання і мінімальний час очікування. Мінімізація затримок та оптимізація процесів пошуку даних мають першорядне значення для забезпечення оперативного реагування на запити користувачів. Важливо, щоб програмне забезпечення

демонструвало стабільність і надійність, запобігаючи критичним помилкам або системним збоям, які потенційно можуть зробити розклад недоступним для студентів. Процедури тестування та забезпечення якості повинні бути чітко встановлені, щоб оперативно виявляти та виправляти будь-які помилки або проблеми, які можуть виникнути.

Враховуючи ці аспекти, виникає задача інженерії програмного забезпечення, яка пов’язана з розробкою програмного забезпечення для надання розкладу занять студентам НУК. На сьогоднішній день доступні різноманітні рішення для розробки програмного забезпечення, яке забезпечує надання розкладу занять студентам. Ці рішення можуть включати розробку веб-порталу, мобільного додатку, Telegram-бота та інтеграцію з існуючими платформами. Задачу розробки програмного забезпечення для надання розкладу занять можна вирішити шляхом розробки та використання Telegram-бота. Telegram-бот - це спеціальний програмний засіб, який забезпечує комунікацію між користувачами та платформою Telegram. Telegram-бот є підходящим рішенням для надання розкладу студентам НУК, оскільки він забезпечує зручний доступ до розкладу безпосередньо у месенджері Telegram. Це забезпечує швидкий та зручний спосіб отримання інформації про пари та нагадування про майбутні заняття. Telegram-боти також відрізняються високою швидкістю роботи та простим інтерфейсом, що спрощує взаємодію студентів з ним.

В процесі розробки програмного забезпечення для надання розкладу занять студентам НУК було вирішено використовувати Firebase як хмарний бекенд-сервіс. Firebase є популярним рішенням, що надає широкий спектр функцій для створення мобільних та веб-додатків. Один з ключових компонентів Firebase - це база даних в режимі реального часу, яка дозволяє миттєво оновлювати дані на всіх підключених пристроях. Це особливо корисно для розкладу студентів, оскільки зміни в розкладі можуть бути відразу відображені для всіх користувачів. Firebase простий у використанні і може значно скоротити час розробки та забезпечить безперебійну синхронізацію даних між Telegram-ботом та веб-сервісом управління.

Для розробки Telegram-бота було вирішено використати мову TypeScript та інтегрувати його з Firebase як функцію [6]. Firebase-функції - це безсерверні функції, які виконуються в хмарі та дозволяють писати код на стороні сервера. Загалом, розгортання Telegram-бота як функції Firebase пропонує перевагу безсерверної моделі, де управління інфраструктурою абстрагується, дозволяючи зосередитися на написанні коду без необхідності вирішувати проблеми масштабування та обслуговування серверів. Функції Firebase можуть легко масштабуватися залежно від попиту, забезпечуючи надійну роботу Telegram-бота. TypeScript, який є сумісним з середовищем виконання Node.js, використовується для написання коду функцій Firebase [7]. Він надає перевагу системи типів, дозволяючи забезпечити перевірку типів та полегшує читання та розуміння коду. Це допомагає зменшити ймовірність помилок та багів, прискорюючи розробку. Для роботи Telegram-бота використовується бібліотека telegraf, яка надає повний набір функцій і має чітку документацію для створення Telegram-ботів з використанням Node.js [8]. Бібліотека має модульну архітектуру, що сприяє легкому розширенню та налаштуванню функціональності Telegram-бота. Система типів TypeScript може допомогти ефективно використовувати бібліотеку telegraf, забезпечуючи перевірку типів та кращу інтеграцію з інструментами розробки. Загалом, використання TypeScript разом з Firebase і бібліотекою telegraf дозволяє забезпечити ефективний розвиток Telegram-бота з використанням безсерверної архітектури, зручною роботою зі змінним навантаженням та гарантованою стабільністю.

Надсилання розкладів підписаним користувачам за 10 хвилин до початку занять є важливою функцією Telegram-бота, оскільки вона забезпечує користувачам своєчасне отримання необхідної інформації. Часовий інтервал в 10 хвилин був обраний з урахуванням потреб користувачів вчасно отримувати розклади та їх зручного організованого використання. Хоча в окремих випадках можуть виникати ситуації, коли 10 хвилин недостатньо для доїхати до місця занять, цей інтервал вважається прийнятним компромісом, який враховує звичайний режим проведення занять та здатність студентів швидко

підготуватися до навчального процесу. Він дозволяє студентам вчасно організувати свій час та приготуватися до занять, знаходячись в університеті або підключитися до відеоконференцій або віртуальних платформ для участі в онлайн заняттях. Для реалізації цієї функціональності у Telegram-боті використовується розробка та інтеграція Cronjob для автоматичного запуску сценаріїв у визначений час. Cronjob викликає Firebase-функцію для надсилання повідомлень з розкладами. Ці кроки є важливими для забезпечення автоматизованого та своєчасного надсилання розкладів студентам, що сприяє зручності та організованості навчального процесу.

Також передбачається наявність веб-сервісу планування, який дозволяє оновлювати розклад та групи у реальному часі. Існують різні фреймворки та інструменти, які можуть допомогти вирішити цю задачу. Наприклад, Node.js з Express, Django або Flutter [9]. Ці фреймворки надають зручні засоби для створення серверної логіки, маршрутизації, роботи з базами даних та взаємодії з клієнтською частиною. Flutter, розроблений Google, був обраний як фреймворк для розробки веб-сервісу планування. Він є перспективним інструментом з легкою інтеграцією з Firebase, що дозволяє створити зручний та ефективний інтерфейс користувача зі знайомим дизайном Material Design [10]. Крім того, Flutter дозволяє швидко розробляти та створювати прототипи, а його функція гарячого перезавантаження дозволяє легко бачити зміни в реальному часі під час розробки коду. Dart – це мова програмування, яка використовується для розробки додатків Flutter, і вона має багато можливостей, які роблять її добре придатною для створення веб-застосунків [11]. Dart має потужну систему типів, яка може допомогти з надійністю коду та виявленням помилок під час розробки. Він також підтримує асинхронне програмування, що може бути корисним для створення адаптивних і масштабованих веб-додатків. Вибравши Flutter для розробки веб-сервісу, можливо скористатися перевагами багаторазового використання коду, швидкого циклу розробки, багатих можливостей інтерфейсу, продуктивності, розробки однією мовою, потужної підтримки спільноти та безперешкодної інтеграції з іншими сервісами. Ці

переваги роблять Flutter переконливим вибором для створення ефективної та візуально привабливої веб-сервісу керування розкладом.

На завершення, розробка програмного забезпечення є складним і багатогранним завданням, яке вимагає глибокого розуміння унікальних викликів, з якими стикаються університети, коли мова йде про розклад, надаючи ефективний і надійний сервіс розкладу через Telegram.

1.2 Аналіз існуючих аналогів планування розкладу та обґрунтування необхідності розробки програмного забезпечення для надання розкладу занять студентам НУК

Для аналізу існуючих аналогів планування розкладу та обґрунтування необхідності розробки програмного забезпечення для надання розкладу занять студентам НУК, можна вивчити різноманітні застосунки, що пропонують подібний функціонал.

My Study Life – це популярний планувальник для студентів і викладачів, покликаний спростити управління вашим навчальним життям [12]. My Study Life дає змогу зберігати домашні завдання та іспити в хмарі, що робить їх доступними на будь-якому пристрої, де б ви не знаходилися. Застосунок має зручний інтерфейс, за допомогою якого студенти можуть легко переглядати свій щоденний та тижневий розклад, включаючи такі деталі, як час занять, місце проведення та викладачів. Він також дозволяє користувачам додавати власні завдання та дедлайни, забезпечуючи організованість та виконання своїх академічних обов'язків. My Study Life ще більше покращує процес планування, надаючи такі функції, як синхронізація між кількома пристроями, хмарне резервне копіювання та сповіщення про майбутні події. Це гарантує, що

студенти матимуть доступ до свого розкладу та важливої інформації, де б вони не були.

Хоча My Study Life пропонує кілька корисних функцій, він також має деякі недоліки:

– Обмежений доступ до розкладу: My Study Life дозволяє зберігати розклад у хмарі, але доступ до нього може бути обмежений лише до власних пристроїв. Це може бути незручним, якщо студент хоче отримувати розклад на різних пристроях або ділитися ним з іншими людьми.

– Відсутність інтеграції з месенджерами: My Study Life не надає можливості отримувати розклад через популярні месенджери, такі як Telegram. Це означає, що студентам потрібно відкривати окремий застосунок або веб-сайт, щоб переглянути свій розклад.

– Відсутність автоматичних нагадувань: My Study Life дозволяє додавати завдання та дедлайни, але він не надає автоматичних нагадувань про майбутні заняття або дедлайни. Це може призвести до пропуску важливих занять або завдань, якщо студент забуває про них.

– Відсутність реального часу оновлення розкладу: My Study Life не забезпечує миттєве оновлення розкладу на всіх пристроях. Якщо розклад змінюється, студентам потрібно вручну оновлювати його на кожному пристрої, що може призвести до невідповідностей та незручностей.

Враховуючи ці недоліки розробка програмного забезпечення для надання розкладу занять студентам НУК, яке включатиме зручний доступ до розкладу у месенджері Telegram, автоматичні нагадування про майбутні заняття та реальний час оновлення розкладу на всіх пристроях, є необхідною для поліпшення досвіду студентів та забезпечення їх ефективності у навчанні. Це дозволило б забезпечити більш зручний та персоналізований досвід для студентів.

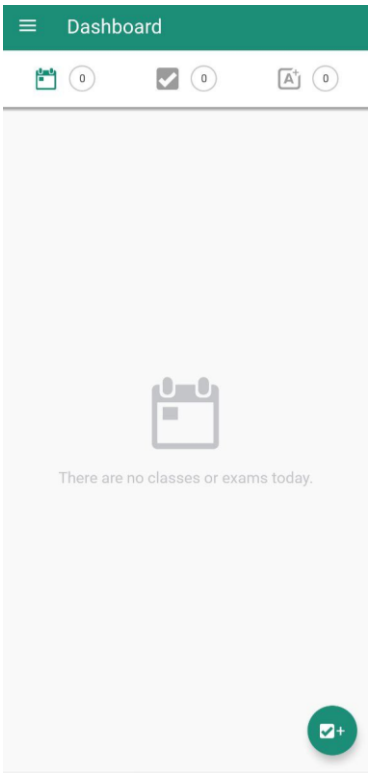

Рисунок 1.1 – Головна сторінка мобільного застосунку My Study Life

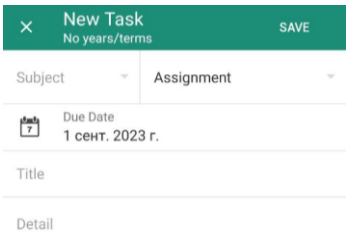

Рисунок 1.2 – Сторінка «Нове завдання» мобільного застосунку My Study Life

SchedulePro – це популярний застосунок для планування, який широко використовується в різних галузях, включаючи освіту, охорону здоров'я та

роздрібну торгівлю [13]. Застосунок має зручний інтерфейс і пропонує безліч функцій, які задовольняють потреби різних типів організацій. SchedulePro забезпечує планування в режимі реального часу, автоматичну оптимізацію розкладу та вирішення конфліктів, що допомагає заощадити час і підвищити продуктивність. Застосунок дозволяє працівникам отримувати доступ до свого робочого графіку, підписуватися на понаднормові роботи та запитувати відгули. Застосунок також надає сповіщення, щоб тримати працівників в курсі будь-яких змін чи оновлень у їхньому графіку.

Однак SchedulePro має деякі обмеження, які можуть зробити його менш придатним для потреб університетів. По-перше, SchedulePro розроблений для ширшого кола підприємств, тоді як власне програмне забезпечення спеціально пристосовано до потреб університетів. Крім того, SchedulePro вимагає передплати та мобільного доступу від організації, тоді як власне програмне забезпечення використовує можливості Telegram, який вже широко використовується студентами та викладачами університетів. Застосунок призначений для роботи на настільних і мобільних пристроях, що може бути незручним для користувачів, які віддають перевагу додаткам для обміну повідомленнями, таким як Telegram.

Тому розробка альтернативи, може забезпечити більш зручний та безперешкодний досвід для студентів та викладачів університетів. Крім того, власне програмне забезпечення може бути адаптоване до конкретних потреб і вимог університетів, пропонуючи такі функції, як оновлення в режимі реального часу, автоматичне планування, а також зручний та функціональний інтерфейс.

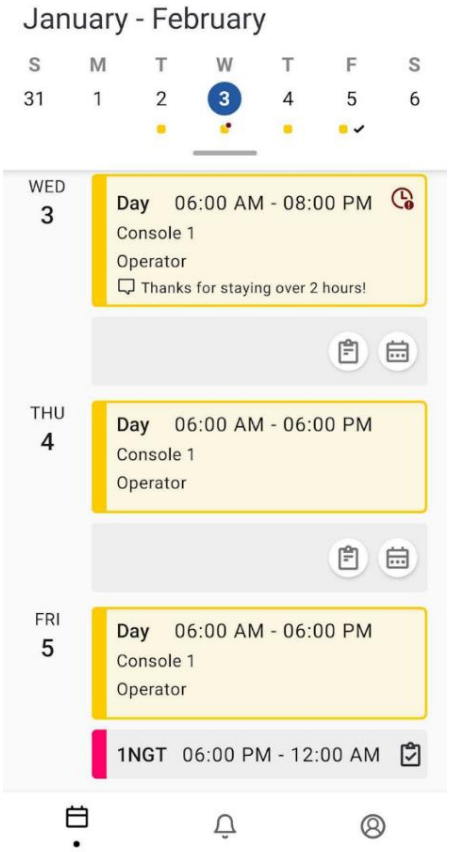

Рисунок 1.3 – Головна сторінка мобільного застосунку SchedulePro

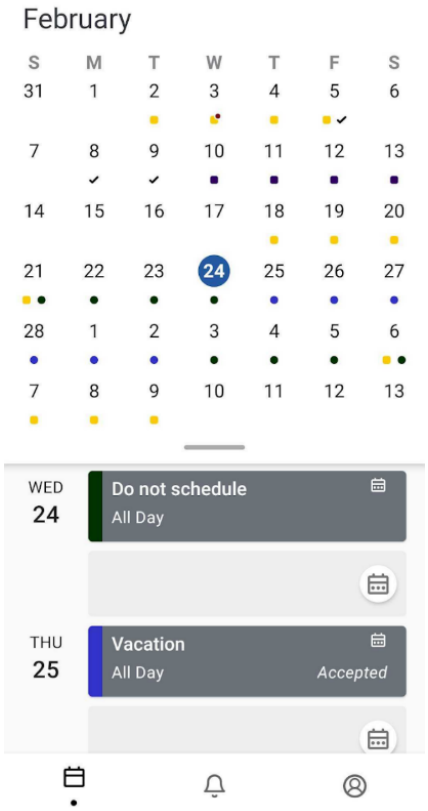

Рисунок 1.4 – Сторінка «Календар» мобільного додатку SchedulePro

Виходячи з вище перелічених недоліків можна зробити висновок, що створення власного програмного забезпечення наддасть кілька додаткових переваг використання індивідуального рішення, а не готових рішень.

Однією з ключових переваг розробки власного програмного забезпечення є те, що воно забезпечує більшу гнучкість і можливість кастомізації. Завдяки власному рішенню можливо адаптувати Telegram-бота і веб-сервіс до конкретних потреб і вимог. Це означає, що можливо створити більш персоналізований досвід для своїх користувачів, що може допомогти підвищити лояльність клієнтів і збільшити їхню залученість.

Ще одна перевага розробки полягає в тому, що вона дасть змогу використовувати новітні технології та інструменти. Залишаючись в курсі останніх тенденцій і технологічних досягнень, можливо переконатися, що Telegram-бот і веб-сервіс оптимізовані для підвищення продуктивності та покращення користувацького досвіду. Це допоможе підвищити задоволеність користувачів.

Крім того, розробка персонального рішення дозволяє забезпечити безпеку і надійність Telegram-бота і веб-сервісу. Готові рішення часто вразливі до кібератак і хакерських атак.

Насамкінець, розробка власного рішення пропонує широкий спектр переваг порівняно з використанням готових рішень. Від більшої гнучкості та кастомізації до можливості використовувати новітні технології та інструменти – власне рішення допоможе створити чудовий продукт, який відповідає потребам та очікуванням сучасних користувачів. Крім того, власне рішення може допомогти забезпечити безпеку та надійність Telegram-бота і веб-сервісу, що має вирішальне значення в сучасному цифровому ландшафті.

1.3 Постановка задачі на розробку програмного забезпечення для надання розкладу занять студентам НУК

Виходячи з аналізу задачі інженерії програмного забезпечення для надання розкладу занять студентам НУК, яку має розв’язувати кваліфікаційна робота, для предметної галузі університетського розкладу, сформулюємо наступну постановку задачі: потрібно розробити Telegram-бот «BingNUOS», яке забезпечить студентам легкий доступ до актуального розкладу та персоналізованих сповіщень та веб-сервіс «BingNUOS» для управління розкладом занять університету, яке би задовольняло наступним вимогам:

– Для користувача Telegram-бота, програма повинна виконувати наступні функції: перегляд та вибір груп, детальний перегляд розкладу із назвою предмета, кабінету та викладача на кожен будній день неділі та підписка або скасувати підписку на повідомлення о майбутніх парах.

– Для модератора веб-сервісу, програма повинна дозволяти додавати, редагувати та видаляти розклад із дозволених груп для модерації.

– Для адміністратора веб-сервісу програма повинна дозволяти додавати, редагувати, видаляти нові групи та розклад. Також повинна передбачатися можливість додавати модераторів та редагувати дозволені групи для модерації.

Функції користувача

– Користувач повинен мати можливість взаємодіяти з Telegram-ботом для вибору групи, підписки на сповіщення про майбутні заняття та перегляду розкладу для обраної групи на кожен робочий день.

– Користувач повинен отримувати сповіщення за 10 хвилин до початку кожного заняття.

– Користувач не повинен мати доступу до веб-сервісі планування.

Функції модератора

– Модератор повинен мати можливість авторизуватися в веб-сервісі планування за допомогою електронної пошти та пароля.

– Модератор повинен мати можливість переглядати розклад для дозволених груп модерації на кожен робочий день.

– Модератор повинен мати можливість додавати, редагувати та видаляти розклад для дозволених груп модерації на кожен робочий день.

– Модератор не повинен мати можливості додавати, редагувати або видаляти групи, реєструвати нових модераторів.

Функції адміністратора

– Адміністратор повинен мати можливість авторизуватися в веб-сервісі планування за допомогою електронної пошти та пароля.

– Адміністратор повинен мати можливість реєструвати нових модераторів і призначати їх до груп модерації.

– Адміністратор повинен мати можливість переглядати розклад усіх груп на кожен робочий день.

– Адміністратор повинен мати можливість створювати, редагувати та видаляти групи та розклади для кожного робочого дня.

Вимоги до організації вхідних та вихідних даних

Дані про розклад та користувачів зберігаються у базі даних. СУБД забезпечує розмежування прав доступу до даних – надає користувачу права на читання, а модератору та адміністратору – на читання та запис.

Вхідні дані:

– Користувач повинен вказати номер своєї групи натиснувши відповідну кнопку із номером групи у Telegram-боті.

– Користувач повинен підписатися або відписатися від сповіщень використовуючи відповідну команду у Telegram-боті (/subscribe або /unsubscribe).

– Модератор та Адміністратор повинен вказати свою електронну пошту та пароль у відповідні поля для авторизації у веб-сервісі планування.

– Модератор та Адміністратор повинні вказати номер групи та деталі розкладу у відповідні поля для додавання або редагування розкладу у веб-сервісі планування.

– Адміністратор повинен вказати ім’я, електронну пошту модератора та призначити його/її до груп модерації у відповідні поля для реєстрації модератора у веб-сервісі планування.

Вихідні дані:

– Телеграм-бот повинен показувати розклад для обраної групи на кожен будній день для любого користувача.

– Telegram-бот повинен надсилати сповіщення за 10 хвилин до початку кожного заняття підписаним користувачам.

– У веб-сервісі планування для модератора повинен відображатися розклад для дозволенних груп модерації на кожен робочий день.

– У веб-сервісі планування для адміністратора повинен відображатися розклад для усіх груп на кожен робочий день.

– У веб-сервісі планування для адміністратора повинен відображатися список модераторів та призначених їм груп модерації.

Вимоги до складу та параметрів технічних засобів

Для роботи із програмним забезпеченням необхідний наступний склад технічних засобів з мінімальними параметрами:

– Процесор Intel Pentium 1Ghz і вище.

– Оперативна пам'ять 512Мб і вище.

– Мінімум дискового простору 1 Гб.

– Екран з роздільною здатністю не менше ніж 800x600 пікселів

Вимоги до інформаційної та програмної сумісності.

Вихідні коди програми повинні бути реалізовані на мовах Dart та TypeScript. В якості інтегрованого середовища розробки програми буде використовуватися середовище Microsoft Visual Studio Code. Використовувати Flutter Framework версії не нижче 2.18.6.

Системні програмні засоби, які використовуються програмою, повинні бути представлені ліцензійною версією операційної системи Windows (версії не нижче Windows 7).

В повному обсязі, вимоги до програмного забезпечення наведенні в технічному завданні у Додатку А.

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ НАДАННЯ РОЗКЛАДУ ЗАНЯТЬ СТУДЕНТАМ НУК

2.1 Побудова моделі варіантів використання програмного забезпечення для надання розкладу занять студентам НУК

В ході аналізу предметної галузі та формуванні вимог було виявлено три актори, котрі взаємодіють з програмним забезпеченням, а саме студент, модератор та адміністратор. Telegram-бот орієнтований на студентів університету НУК для надання розкладу, надалі їх названо «Користувачем». Веб-сервіс планування орієнтований на модераторів та адміністраторів для керування розкладом.

Користувач може взаємодіяти з системою через Telegram-бот. Користувач може переглядати розклад обраної групи, підписатися на сповіщення про майбутні заняття та отримувати своєчасні нагадування перед кожним заняттям.

Модератор відповідає за управління розкладами своїх груп в веб-сервісі планування. Вони мають авторизований доступ до системи за допомогою електронної пошти та пароля. Модератори можуть переглядати розклад для авторизованих груп модерації, додавати, редагувати та видаляти розклади на кожен робочий день. Однак вони не мають повноважень додавати, редагувати чи видаляти групи або реєструвати нових модераторів.

Адміністратор має адміністративні привілеї у веб-сервісі планування. Він може увійти, використовуючи свою електронну пошту та пароль, і його роль включає управління групами, розкладами та користувачами. Адміністратори можуть переглядати розклад для всіх груп на кожен робочий день, створювати,

редагувати і видаляти групи і розклади, а також реєструвати нових модераторів і призначати їх до груп модерації.

Таблиця 2.1 – Короткий опис можливостей акторів

Актор	Можливості
1	2
Користувач	Перегляд та вибір груп, детальний перегляд розкладу із назвою предмета, кабінету та викладача на кожен будній день неділі та підписка або скасування підписки на сповіщення о майбутніх парах.
Модератор	Додавати, редагувати та видаляти розклад із дозволених груп модерації
Адміністратор	Додавати, редагувати, видаляти групи та розклад. Додавати, редагувати, видаляти модераторів та їх дозволені групи для модерації.

Загалом ці актори взаємодіють з програмним забезпеченням наступним чином:

– Користувач взаємодіє з Telegram-ботом для вибору групи, підписки на сповіщення та перегляду розкладу для обраної групи на кожен робочий день. Він отримує сповіщення за 10 хвилин до початку кожного заняття.

– Модератор входить до веб-сервісу планування розкладу за допомогою своєї електронної пошти та пароля. Переглядає розклад для авторизованих груп модерації на кожен робочий день. Додає, редагує та видаляє розклади для дозволених груп модерації на кожен робочий день.

– Адміністратор входить веб-сервісу планування розкладу за допомогою електронної пошти та пароля. Реєструє нових модераторів і призначає їх до груп модерації. Переглядає розклад для всіх груп на кожен робочий день. Створює, редагує та видаляє групи та розклади на кожен робочий день.

Виявлені варіанти використання програмного забезпечення для надання розкладу занять зведено до таблиці 2.2.

Таблиця 2.2 – Варіанти використання

Актор	Назва	Короткий опис
1	2	3
Користувач	Вибір групи	Користувач може обрати групу із доступних
Користувач	Підписатися на сповіщення	Користувач може підписатися на сповіщення про майбутні пари
Користувач	Відписатись від сповіщень	Користувач може відписатись від сповіщень про майбутні пари
Користувач	Меню з вибором дня тижня для отримання розкладу	Користувач може обрати будній день тижня для отримання детального розкладу.
Адміністратор, Модератор	Автентифікація	Адміністратор та модератор може пройти автентифікацію за допомогою електронної пошти та паролю
Адміністратор, Модератор	Відновити пароль	Адміністратор та модератор може відновити пароль за допомогою електронної пошти
Адміністратор	Керування модераторами	Адміністратор може керувати модераторами та призначати їх до груп модерації
Адміністратор	Керувати розкладом	Адміністратор може додавати, редагувати та видаляти нові групи та розклад до них
Модератор	Керувати розкладом із дозволених груп	Модератор може переглядати, редагувати та видаляти розклад із дозволених груп

На рисунку 2.1 представлено діаграму варіантів використання

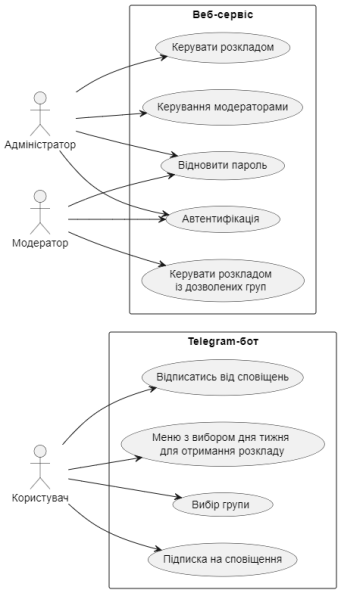

Рисунок 2.1 – Діаграма варіантів використання

Ці актори та їхні відповідні ролі і функції формують основу програмної системи, що дозволяє користувачам, модераторам та адміністраторам ефективно керувати розкладом занять в НУК та мати доступ до нього.

2.2 Специфікації варіантів використання

Специфікації варіантів використання Telegram-бота представлені в таблицях 2.3 – 2.6.

Таблиця 2.3 – Сценарій використання «Вибір групи»

Опис прецеденту	Вибір групи
Дійові особи	Користувач
Передумови	Немає
Потік подій	Користувач запускає Telegram-бот або вводить команду /group.
Базовий потік	1. Telegram-бот відображає список курсів. 2. Користувач обирає свій курс. 3. Telegram-бот відображає список доступних груп, з яких користувач може вибрати, або повернутися назад до вибору курсу. 4. Користувач переглядає список груп та шукає потрібну групу. 5. Користувач вибирає потрібну групу, натискаючи на неї. 6. Telegram-бот підтверджує вибір групи.
Альтернативні потоки	Відсутні.
Постумови	Користувач має можливість використовувати Telegram-бот з обраною групою та виконувати функції, пов'язані з вибраною групою.

Таблиця 2.4 – Сценарій використання «Підписка на сповіщення»

Опис прецеденту	Підписка на сповіщення
1	2
Дійові особи	Користувач
Поток подій	Користувач починає взаємодію з Telegram-ботом і активує опцію підписки, натискаючи на відповідну кнопку у меню бота або вводячи текстову команду /subscribe.
Передумови	Користувач повинен вибрати групу

Продовження таблиці 2.4

1	2
Базовий потік	Telegram-бот підтверджує, що користувач успішно підписаний на отримання сповіщень про майбутні пари.
Альтернативні потоки	Telegram-бот відправляє повідомлення, запитуючи користувача про вибір групи, якщо він ще не зробив цього
Постумови	За настанням майбутнього заняття, Telegram-бот автоматично надсилає сповіщення користувачеві, яке містить інформацію про деталі пари (назва, час, кабінет та викладач).

Таблиця 2.5 – Сценарій використання «Відписка від сповіщень»

Опис прецеденту	Відписка від сповіщень
Дійові особи	Користувач
Поток подій	Користувач починає взаємодію з Telegram-ботом і деактивує опцію підписки, натискаючи на відповідну кнопку у меню бота або вводячи текстову команду /unsubscribe.
Передумови	Користувач повинен вибрати групу
Базовий потік	Telegram-бот підтверджує, що користувач більше не підписан на отримання сповіщень про майбутні пари.
Альтернативні потоки	Telegram-бот відправляє повідомлення, запитуючи користувача про вибір групи, якщо він ще не зробив цього
Постумови	Відсутні

Таблиця 2.6 – Сценарій використання «Меню з вибором дня тижня для отримання розкладу»

Опис прецеденту	Меню з вибором дня тижня для отримання розкладу
1	2
Дійові особи	Користувач
Поток подій	Користувач починає взаємодію з Telegram-ботом і натискає на відповідну кнопку у меню бота або вводячи текстову команду /schedule.
Передумови	Користувач повинен вибрати групу

Продовження таблиці 2.6

1	2
Базовий потік	1. Користувач починає взаємодію з Telegram-ботом і вводить команду /schedule або натискає кнопку для виклику меню з днями тижня. 2. Telegram-бот відправляє повідомлення, пропонуючи користувачеві вибрати день тижня з доступних варіантів. 3. Користувач обирає день тижня, натискаючи на відповідну кнопку або вводячи текстову команду. 4. Telegram-бот відображає користувачеві детальний розклад занять для обраного дня. Розклад включає час початку пари, назву пари, кабінет та викладача.
Альтернативні потоки	Telegram-бот відправляє повідомлення, запитуючи користувача про вибір групи, якщо він ще не зробив цього
Постумови	Відсутні

Специфікації варіантів використання веб-сервісу планування представлені в таблицях 2.7 – 2.12

Таблиця 2.7 – Сценарій використання «Автентифікація»

Опис прецеденту	Автентифікація
1	2
Дійові особи	Адміністратор, Модератор
Поток подій	Адміністратор або модератор відкриває веб-сервіс, який вимагає автентифікації для доступу.
Передумови	Адміністратор або модератор мають бути зареєстрованими у веб-сервісі.
Базовий потік	1. Адміністратор або модератор відкриває веб-сервіс і система відображає сторінку автентифікації, де користувач повинен ввести свою електронну пошту та пароль. 2. Адміністратор або модератор вводить свою електронну пошту та пароль у відповідні поля. 3. Після введення облікових даних, адміністратор або модератор натискає кнопку «Увійти». 4. Система перевіряє введені дані на правильність. Якщо дані введені вірно, адміністратору або модератору надається доступ до системи з відповідними привілеями в залежності від його ролі.

Продовження таблиці 2.7

1	2
Альтернативні потоки	У разі неправильного введення облікових даних, система відображає повідомлення про помилку і користувач повинен повторити процедуру автентифікації з правильними даними.
Постумови	Після успішної автентифікації, адміністратору або модератору надається доступ до повного або обмеженого функціоналу системи в залежності від його ролі. Він/вона може виконувати необхідні дії або взаємодіяти з системою на основі своїх привілеїв.

Таблиця 2.8 – Сценарій використання «Відновлення пароля»

Опис прецеденту	Відновлення пароля
1	2
Дійові особи	Адміністратор, Модератор
Поток подій	Адміністратор або модератор відкриває сторінку автентифікації у веб-сервісі.
Передумови	Адміністратор або модератор мають бути зареєстрованими у веб-сервісі.
Базовий потік	1. Адміністратор або модератор натискає на кнопку «Забули пароль?» на сторінці автентифікації. 2. Система відображає сторінку відновлення паролю, де користувач повинен ввести свою електронну пошту, пов'язану з обліковим записом. 3. Адміністратор або модератор вводить свою електронну пошту у відповідне поле та натискає кнопку «Відновити пароль». 4. Система перевіряє, чи існує введена електронна пошта у системі. Якщо пошта знайдена, система генерує та надсилає на вказану електронну пошту лист із посиланням для відновлення паролю. 5. Адміністратор або модератор перевіряє свою електронну пошту і переходить за посиланням, надісланим системою. 6. Система відображає сторінку з формою для введення нового паролю де адміністратор або модератор вводить новий пароль у відповідні поля та натискають кнопку «Зберегти пароль». 7. Система перевіряє введений пароль та зберігає його як новий пароль для облікового запису користувача та повідомляє користувача про успішну зміну паролю. 8. Користувач відкриває сторінку автентифікації, де він може увійти зі своїм новим паролем.

Продовження таблиці 2.8

1	2
Альтернативні потоки	Якщо введена електронна пошта не знайдена у системі, система відображає повідомлення про помилку і надає користувачеві можливість повторити процедуру введення електронної пошти з правильними даними.
Постумови	Після відновлення паролю, користувач може використовувати новий пароль для доступу до системи і здійснювати відповідні дії в межах своїх повноважень.

Таблиця 2.9 – Сценарій використання «Керування модераторами»

Опис прецеденту	Керування модераторами
1	2
Дійові особи	Адміністратор
Поток подій	Адміністратор відкриває сторінку керування користувачами у веб-сервісі.
Передумови	Адміністратор увійшов в систему веб-сервісу планування за допомогою свого облікового запису та пароля.
Базовий потік	Адміністратор відкриває сторінку керування користувачами у веб-сервісі. Система відкриває сторінку для керування модераторами. Адміністратор додає нового модератора, вводячи його ім'я, пошту та доступні групи. Після натискання кнопки "Додати", система реєструє модератора та повідомляє адміністратора про успішне додавання.
Альтернативні потоки	Редагування налаштувань модератора – Адміністратор відкриває сторінку керування модераторами та вибирає модератора. – Система відображає деталі обраного модератора. – Адміністратор змінює ім'я, пошту або доступні групи модератора. – Після натискання кнопки "Зберегти", система оновлює налаштування модератора та повідомляє адміністратора про успішне збереження змін. Видалення модератора: – Адміністратор відкриває сторінку керування модераторами та вибирає модератора для видалення. – Система підтверджує видалення модератора та виконує відповідні дії для видалення облікового запису. – Система повідомляє адміністратора про успішне видалення модератора.

Продовження таблиці 2.9

1	2
Постумови	Модератор отримує повідомлення про створення облікового запису і може зайти в панель модератора, скинувши пароль, використовуючи свою пошту для входу в систему.

Таблиця 2.10 – Сценарій використання «Керувати розкладом»

Опис прецеденту	Керувати розкладом
1	2
Дійові особи	Адміністратор
Поток подій	Адміністратор відкриває головну сторінку веб-сервісу.
Передумови	Адміністратор увійшов в систему веб-сервісу планування за допомогою свого облікового запису та пароля.
Базовий потік	1. Система відображає список груп разом з розкладом на будні дні. 2. Адміністратор переглядає інформацію про кожну групу, включаючи назву, номер та розклад занять. 3. Для додавання нової групи адміністратор натискає на опцію додавання. 4. Система відкриває форму для введення деталей нової групи, таких як назва. 5. Адміністратор зберігає нову групу після введення інформації. 6. Для редагування розкладу та назви існуючої групи, адміністратор вибирає її зі списку та натискає на опцію редагування. 7. Система відкриває вікно редагування, де адміністратор змінює розклад занять. 8. Після внесення змін адміністратор зберігає оновлену інформацію про групу та розклад. 9. Для видалення групи та її розкладу, адміністратор вибирає її зі списку та натискає на опцію видалення. 10. Система підтверджує видалення групи та пов'язаного розкладу. 11. Після підтвердження група та розклад видаляються з системи.

Продовження таблиці 2.10

1	2
Альтернативні потоки	– Якщо адміністратор не вводить необхідну інформацію про нову групу, система вимагає повний ввід даних. – Якщо адміністратор вводить некоректну інформацію про нову групу, система вимагає ввести коректні дані. – Невнесені зміни в розклад після редагування не зберігаються системою. – Відміна видалення групи та розкладу залишає дані без змін.
Постумови	Зміни, внесені адміністратором до розкладу, зберігаються в системі і відображаються для користувачів, які переглядають розклад.

Таблиця 2.11 – Сценарій використання «Керувати розкладом із дозволених груп»

Опис прецеденту	Керувати розкладом
1	2
Дійові особи	Модератор
Поток подій	Модератор відкриває головну сторінку веб-сервісу.
Передумови	Модератор увійшов у веб-сервіс планування за допомогою свого облікового запису та пароля.
Базовий потік	1. Після входу модератора в систему відображаються його дозволені групи та відповідний розклад. 2. Модератор може переглядати розклад для кожної дозволеної групи для отримання інформації про заняття. 3. Модератор може редагувати розклад для дозволених груп, включаючи зміну часу, назви, кабінету та викладача. 4. Модератор вибирає групу, для якої потрібно змінити розклад. 5. Система відкриває вікно редагування розкладу для обраної групи. 6. Модератор вносить зміни до розкладу, такі як час, назва, кабінет або викладач. 7. Після внесення змін модератор зберігає оновлений розклад. 8. Зміни в розкладі або його видалення відображаються в системі для користувачів, які переглядають розклад.

Продовження таблиці 2.11

1	2
Альтернативні потоки	– Якщо модератор не має дозволу на керування розкладом жодної групи, система відображає повідомлення про відсутність доступу до керування розкладом. – Модератор може добавити розклад для дозволених груп. – Модератор може видалити розклад для дозволених груп.
Постумови	Модератор може змінювати розклад лише для груп, до яких він має дозвіл. Зміни, внесені модератором до розкладу, зберігаються в системі і відображаються для користувачів, які переглядають розклад.

Ці таблиці варіантів використання допомагають описати послідовність подій та дій користувачів при взаємодії з Telegram-ботом і веб-сервісом. Вони допомагають зрозуміти функціональні можливості системи і спрощують її аналіз та розробку.

2.3 Розробка архітектури програмного забезпечення для надання розкладу занять студентам НУК

Для розробки програмного забезпечення було обрано архітектуру MVC (Model-View-Controller). Вибір архітектури MVC для розробки веб-сервісу та Telegram-бота для надання розкладу занять студентам НУК обґрунтований її здатністю розділяти проблеми та покращувати супровідність, масштабованість та повторне використання коду. Архітектура MVC розділяє систему на три основні компоненти: модель (Model), представлення (View) та контролер (Controller). Ось як ці компоненти використовуються в архітектурі програмного забезпечення:

Модель (Model):

– Компонент Model представляє дані та бізнес-логіку системи.

– Він відповідає за отримання та маніпулювання даними, пов'язаними з розкладом, групами, модераторами та адміністраторами, з бази даних.

– Модель також виконує необхідні розрахунки і перевірки, необхідні для планування операцій.

– Вона надає інтерфейс для доступу до даних і їх модифікації, забезпечуючи цілісність і узгодженість даних.

Представлення (View):

Компонент View відповідає за логіку представлення системи.

– У веб-сервісі компонент View представляє користувацький інтерфейс, з яким взаємодіють студенти, модератори та адміністратори НУК.

– Він відображає розклад занять, інформацію про групи та сповіщення у зручному для користувача вигляді.

– Компонент View також обробляє дані, введені користувачем, і взаємодіє з контролером для виконання відповідних дій.

Контролер (Controller):

– Компонент Контролер діє як посередник між моделлю та представленням.

– Він отримує вхідні дані від користувача та ініціює відповідні дії на основі цих даних.

– Контролер обробляє запити, взаємодіє з моделлю для отримання або модифікації даних і відповідно оновлює подання.

– Він забезпечує розподіл завдань і організовує потік інформації між моделлю і представленням.

Взаємодія між цими компонентами може бути представлена за допомогою діаграм компонентів і макетів UML. Діаграма компонентів ілюструє структурну організацію системи, виділяючи компоненти «Модель», «Представлення» і «Контролер», а також їх залежності і взаємодію. Макети UML можуть бути

використані для надання більш детальної специфікації кожного компонента, демонструючи їх внутрішню структуру та інтерфейси.

Прийнявши архітектуру MVC, програмна система може досягти наступних переваг:

– Розділення проблем: Кожен компонент має чітко визначену сферу відповідальності, що сприяє модульності та полегшує обслуговування.

– Повторне використання коду: Компоненти моделі та контролера можна повторно використовувати на різних платформах, таких як веб-сервіс та Telegram-бот, мінімізуючи зусилля з розробки.

– Масштабованість: У міру зростання системи можна додавати нові функції, розширюючи модель, представлення або контролер, не впливаючи на інші компоненти.

– Покращене тестування: Поділ завдань полегшує модульне тестування окремих компонентів, забезпечуючи коректність кожної частини незалежно від інших.

Нижче, на рисунку 2.2 представлено діаграму компонентів в загальному випадку:

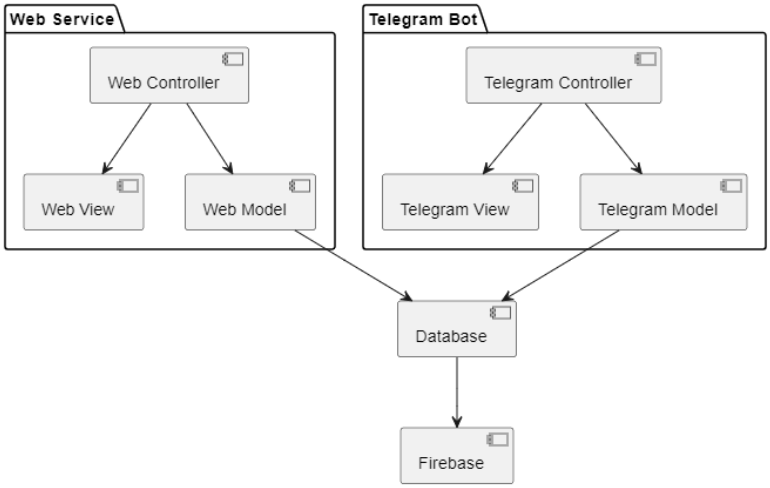

Рисунок 2.2 – Діаграма компонентів

На цій схемі архітектура програмного забезпечення розділена на два основні пакети: «Веб-сервіс» та «Telegram-бот». Кожен пакет складається з

трьох компонентів: Controller, Model та View. Компоненти всередині кожного пакета взаємодіють між собою та з компонентом Firebase (що представляє базу даних).

Пакет «Веб-сервіс» включає компоненти Веб-контролер, Веб-модель та Веб-представлення. Веб-контролер взаємодіє з веб-моделлю для отримання та оновлення даних, а також з веб-представленням для обробки даних, введених користувачем, та оновлення інтерфейсу. Веб-модель взаємодіє з компонентом Firebase для зберігання та отримання даних, пов'язаних з розкладом. Аналогічно, пакет «Telegram-бот» містить компоненти Telegram Controller, Telegram Model і Telegram View. Контролер Telegram Controller взаємодіє з Telegram Model і Telegram View для обробки користувацького вводу і відповідного оновлення інтерфейсу. Компонент Telegram Model взаємодіє з компонентом Firebase для зберігання та пошуку даних. Компонент Firebase являє собою базу даних, де зберігаються дані розкладу, інформація про групи та інші відповідні дані. Ця діаграма надає огляд компонентів та їх взаємозв'язків в архітектурі програмного забезпечення, демонструючи високорівневу структуру системи на основі MVC. На рисунку 2.3 представлено ланцюжок взаємодії компонентів.

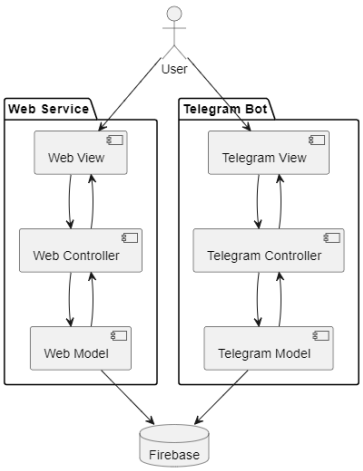

Рисунок 2.3 – Діаграма взаємодії компонентів

На цій схемі архітектура програмного забезпечення розділена на два основні пакети: «Веб-сервіс» та «Телеграм-бот». Кожен пакет складається з

трьох компонентів: Controller, Model та View. Компоненти всередині кожного пакета взаємодіють між собою та з компонентом Firebase (що представляє базу даних).

Для того, щоб візуально проілюструвати фізичне розгортання системи, рекомендується використовувати діаграму розгортання, яка є технікою побудови діаграм в рамках уніфікованої мови моделювання (UML), спеціально розробленої для представлення фізичної архітектури програмного забезпечення. Ця діаграма надає всебічний огляд фізичних компонентів системи, демонструючи взаємодію між програмними та апаратними елементами. Використовуючи діаграму розгортання, ви можете ефективно зобразити різні аспекти, включаючи програмні додатки, сервери, комп'ютери, бази даних, мережі та інші відповідні об'єкти, що складають систему. Основна мета діаграми розгортання - зобразити фізичне розташування цих компонентів та їх взаємозв'язок. Вона пропонує візуальне представлення їх фізичного розташування і демонструє спосіб, у який вони взаємопов'язані або взаємодіють один з одним. Ця діаграма слугує цінним інструментом для розуміння та комунікації фізичної інфраструктури системи, допомагаючи в аналізі, проектуванні та оцінці загального розгортання системи. На рисунку 2.4 представлено діаграму розміщення.

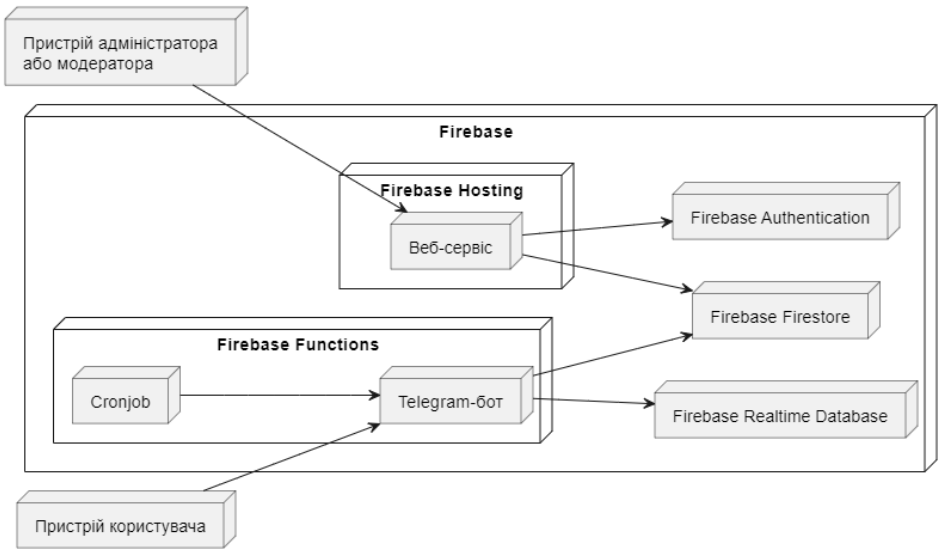

Рисунок 2.4 – «Діаграма розміщення»

На цій схемі пристрій адміністратора або модератора взаємодіє з веб-сервісом, розміщеним на хостингу Firebase. Веб-сервіс має автентифікацію яка взаємодіє з Firebase Authentication та базу даних користувачів та розкладу яка взаємодіє з Firebase Firestore. Пристрій користувача взаємодіє з Telegram-ботом, розміщеним на Firebase Functions. Telegram-бот взаємодіє з базою даних Firebase Realtime для зберігання даних користувача та Firebase Firestore для отримання даних розкладу. Функція Cronjob взаємодіє із Telegram-ботом для надсилання розкладу у певний час. Зв'язки між компонентами представляють комунікацію та потік даних між ними у фізичному розгортанні системи.

2.3 Проект програмного забезпечення для надання розкладу занять студентам НУК

Під час розробки програмного забезпечення для надання розкладу занять студентам НУК було створено декілька артефактів для формалізації поведінки та дизайну системи. Це включало побудову діаграм діяльності для представлення сценаріїв основних варіантів використання, а також розробку інформаційної моделі з використанням методології IDEF1X. Діаграми діяльності були використані для зображення послідовності дій і поведінки в системі або за участю акторів. Ці діаграми демонструють потік дій, умови, необхідні для виконання кожної дії, а також вхідні та вихідні переходи між кроками. Візуалізуючи ці сценарії, стає легше зрозуміти та проаналізувати поведінку системи і переконатися, що всі необхідні дії враховані. На додаток до діаграм діяльності була побудована концептуальна модель даних за допомогою методології IDEF1X. Ця модель відображає інформаційну структуру та взаємозв'язки всередині системи. Вона забезпечує чітке розуміння сутностей,

атрибутів та асоціацій, що допомагають забезпечити цілісність даних та сприяють ефективному управлінню даними.

Крім того, на етапі концептуального проектування було розроблено дизайн інтерфейсу користувача. Дизайн інтерфейсу користувача зосереджений на створенні інтуїтивно зрозумілого та зручного інтерфейсу для взаємодії з програмним забезпеченням. Він враховує такі аспекти, як макет, навігація та візуальні елементи для покращення користувацького досвіду та зручності використання. Загалом, ці артефакти відіграють вирішальну роль у процесі розробки програмного забезпечення, формалізуючи поведінку системи, інформаційну структуру та інтерфейс користувача. Вони надають візуальне представлення функціональності системи, забезпечуючи всебічне розуміння її роботи та сприяючи ефективній комунікації між зацікавленими сторонами.

Діаграми діяльності для Telegram-бота представлені на рисунках 2.5-2.8

Рисунок 2.5 – Діаграма діяльності «Вибір групи»

Ця діаграма дій ілюструє послідовні кроки користувача при виборі групи в Telegram-боті. Спочатку користувач запускає бота або вводить команду /group. Потім бот відображає список доступних курсів, з якого користувач обирає курс, що його цікавить. Бот представляє список груп, пов'язаних з обраним курсом, що дозволяє користувачеві знайти і вибрати потрібну групу. Після того, як користувач зробив вибір, бот підтверджує вибір групи.

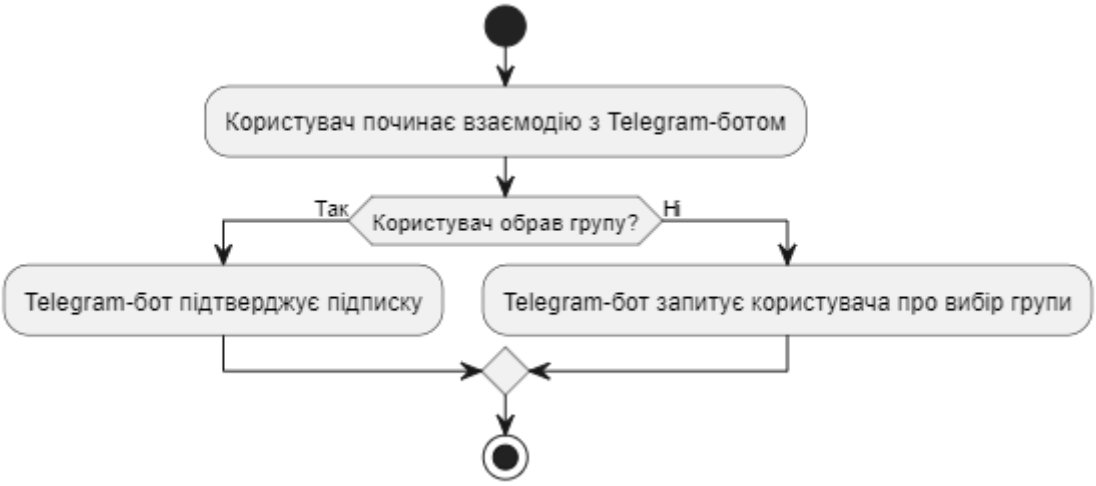

Рисунок 2.6 – Діаграма діяльності «Підписка на сповіщення»

Ця діаграма показує послідовні кроки, які користувач здійснює для підписки на сповіщення в Telegram-боті. Користувач розпочинає взаємодію з ботом, а потім, якщо він вже обрав групу, бот підтверджує підписку. У випадку, якщо користувач ще не обрав групу, бот запитує його про вибір групи.

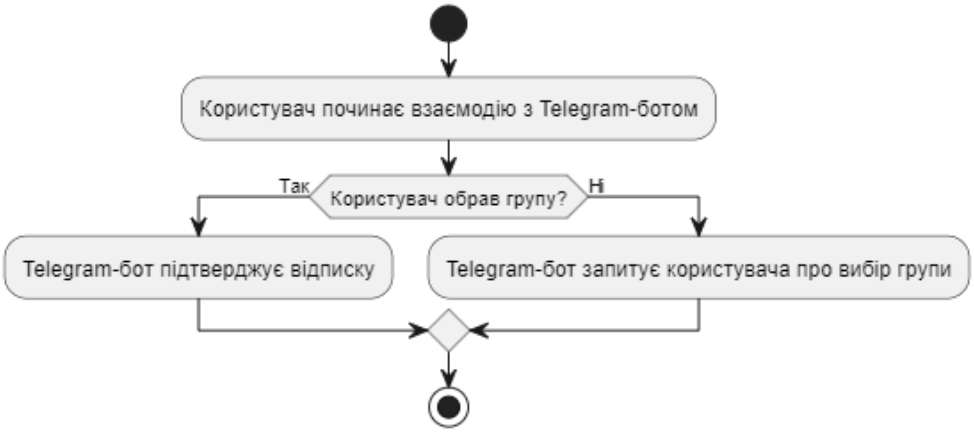

Рисунок 2.7 – Діаграма діяльності «Відписка від сповіщень»

Ця діаграма показує послідовні кроки, які користувач здійснює для відписки від сповіщень в Telegram-боті. Користувач розпочинає взаємодію з ботом, а потім, якщо він вже обрав групу, бот підтверджує відписку. У випадку, якщо користувач ще не обрав групу, бот запитує його про вибір групи.

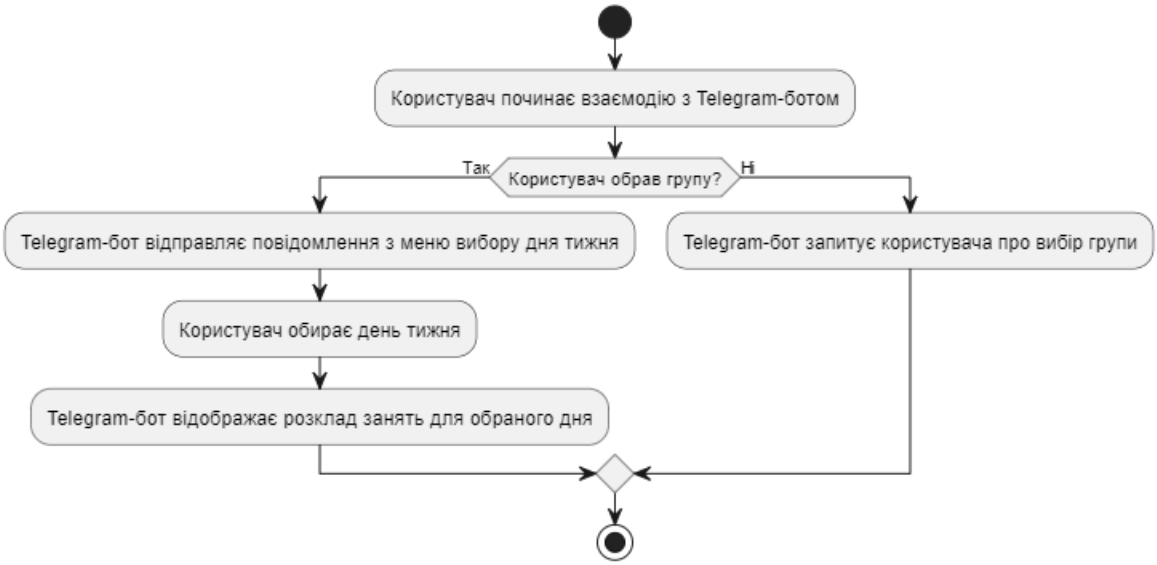

Рисунок 2.8 – Діаграма діяльності «Меню з вибором дня тижня для отримання розкладу»

Ця діаграма показує послідовні кроки, які користувач здійснює для використання меню з вибором дня тижня в Telegram-боті. Після початку взаємодії з ботом, якщо користувач вже обрав групу, бот відправляє повідомлення з меню вибору дня тижня. Користувач обирає день тижня, а потім бот відображає розклад занять для обраного дня. У випадку, якщо користувач ще не обрав групу, бот запитує його про вибір групи. Ця діаграма не містить альтернативних потоків або виключних ситуацій, які були вказані в наданій інформації.

Діаграми діяльності для веб-сервісу планування представлені на рисунках 2.9-2.13

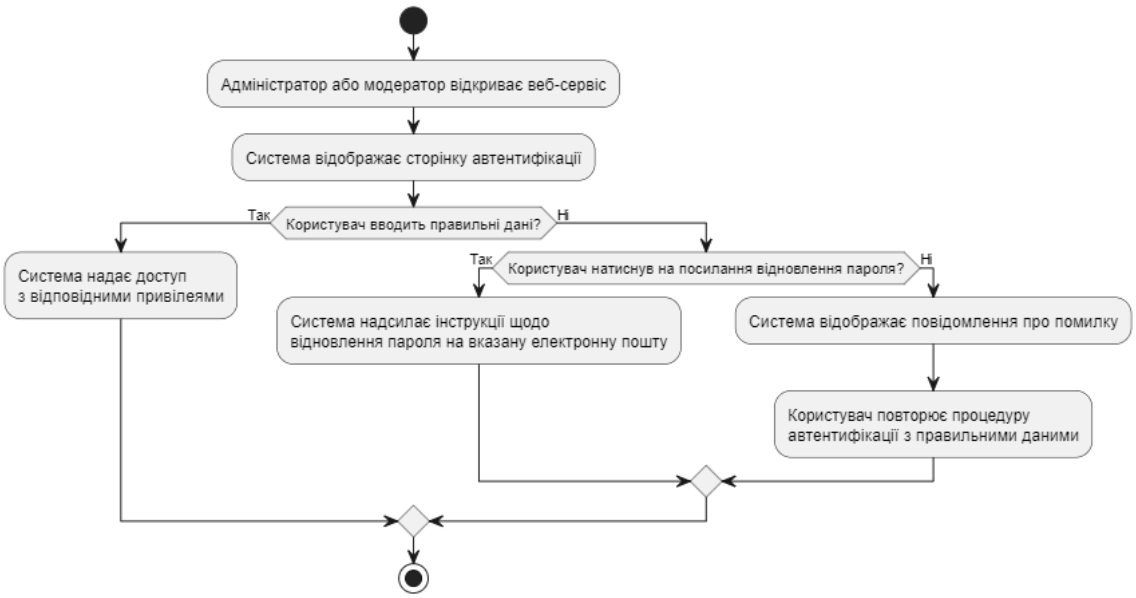

Рисунок 2.9 – Діаграма діяльності «Автентифікація»

Ця діаграма показує послідовні кроки, які адміністратор або модератор здійснюють для автентифікації в веб-сервісі.

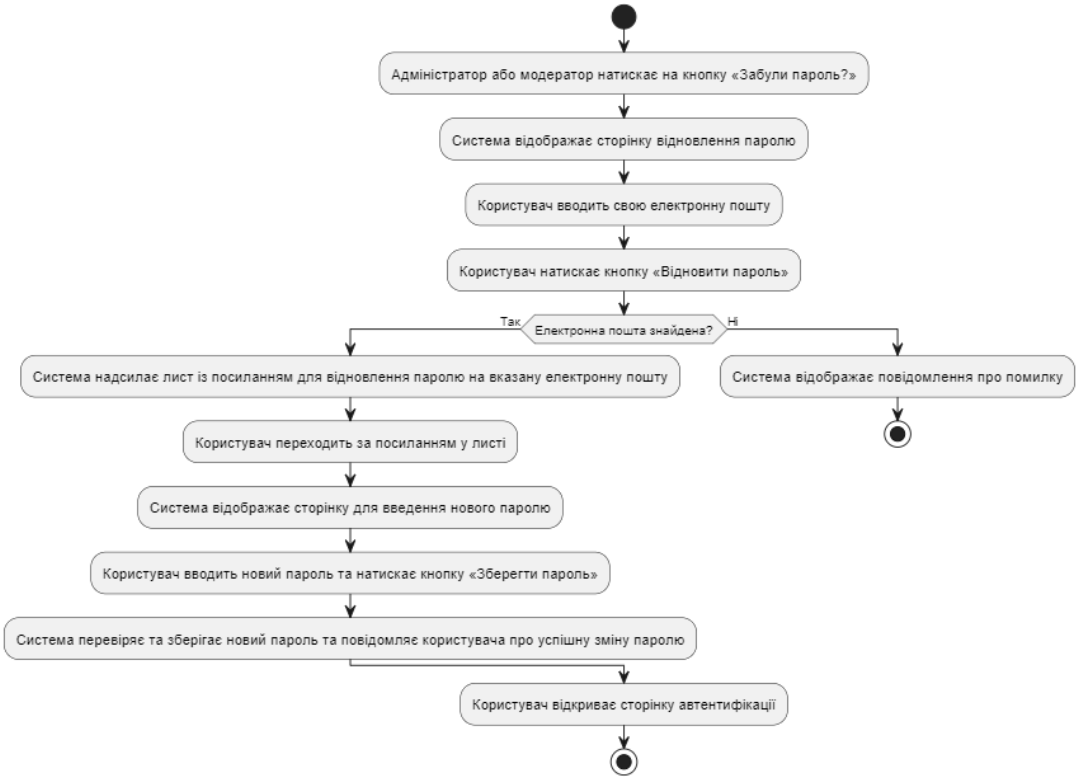

Рисунок 2.10 – Діаграма діяльності «Відновлення пароля»

Ця діаграма показує послідовні кроки, які адміністратор або модератор здійснюють для відновлення пароля в веб-сервісі.

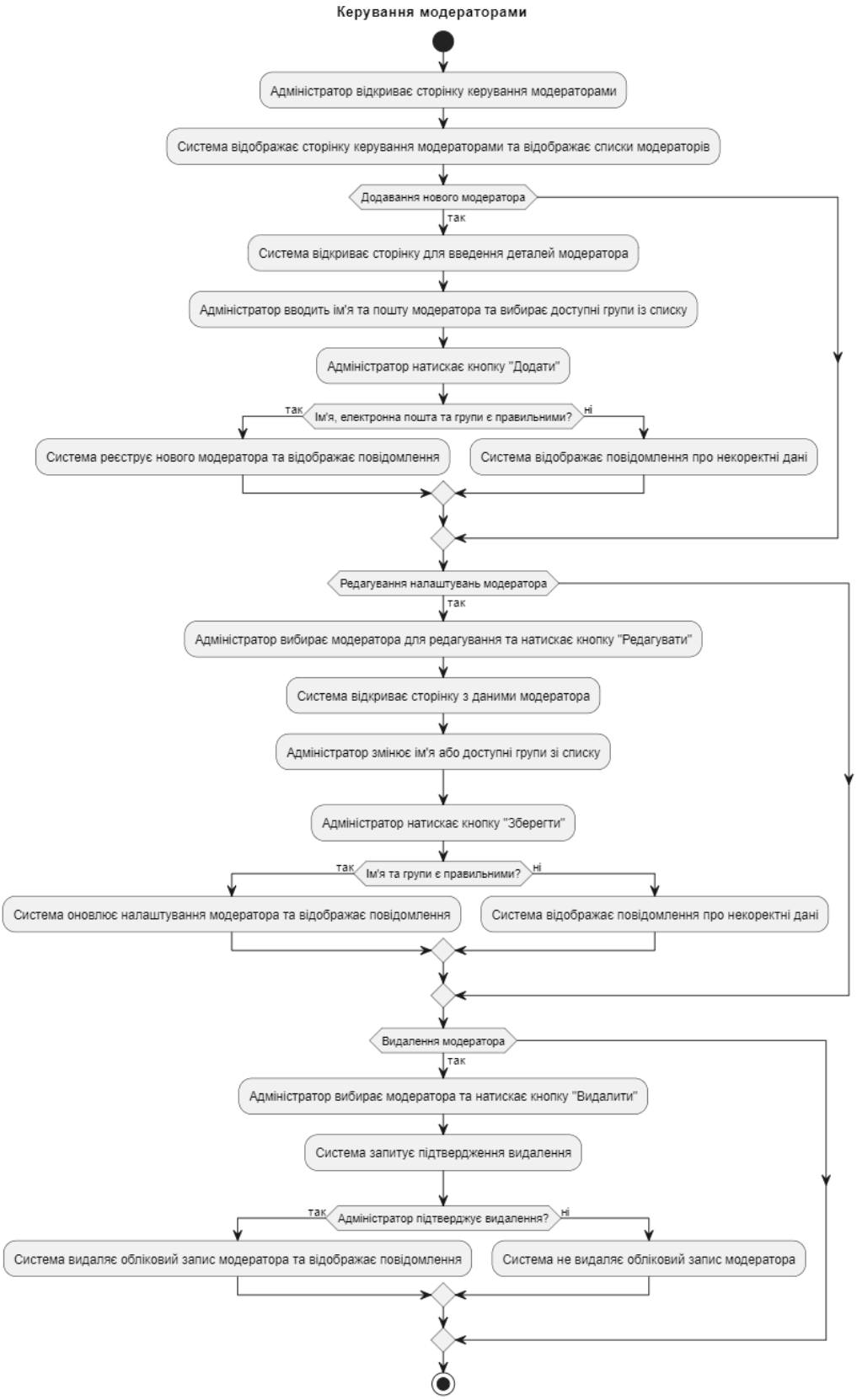

Рисунок 2.11 – Діаграма діяльності «Керувати модераторами»

Ця діаграма описує послідовність дій адміністратора при керуванні модераторів веб-сервісу.

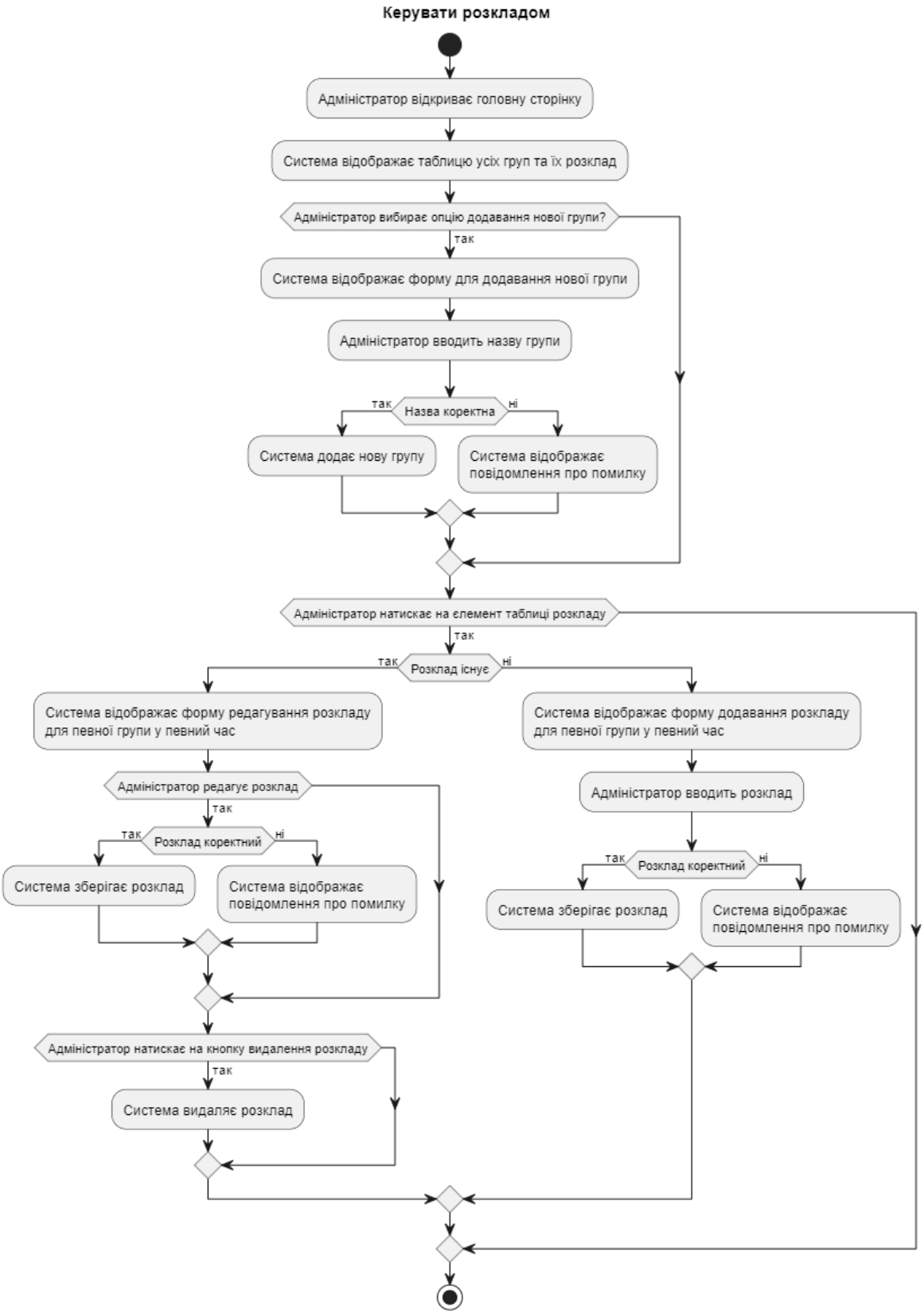

Рисунок 2.12 – Діаграма діяльності «Керувати розкладом»

Ця діаграма діяльності описує послідовність дій адміністратора при керуванні розкладом веб-сервісу.

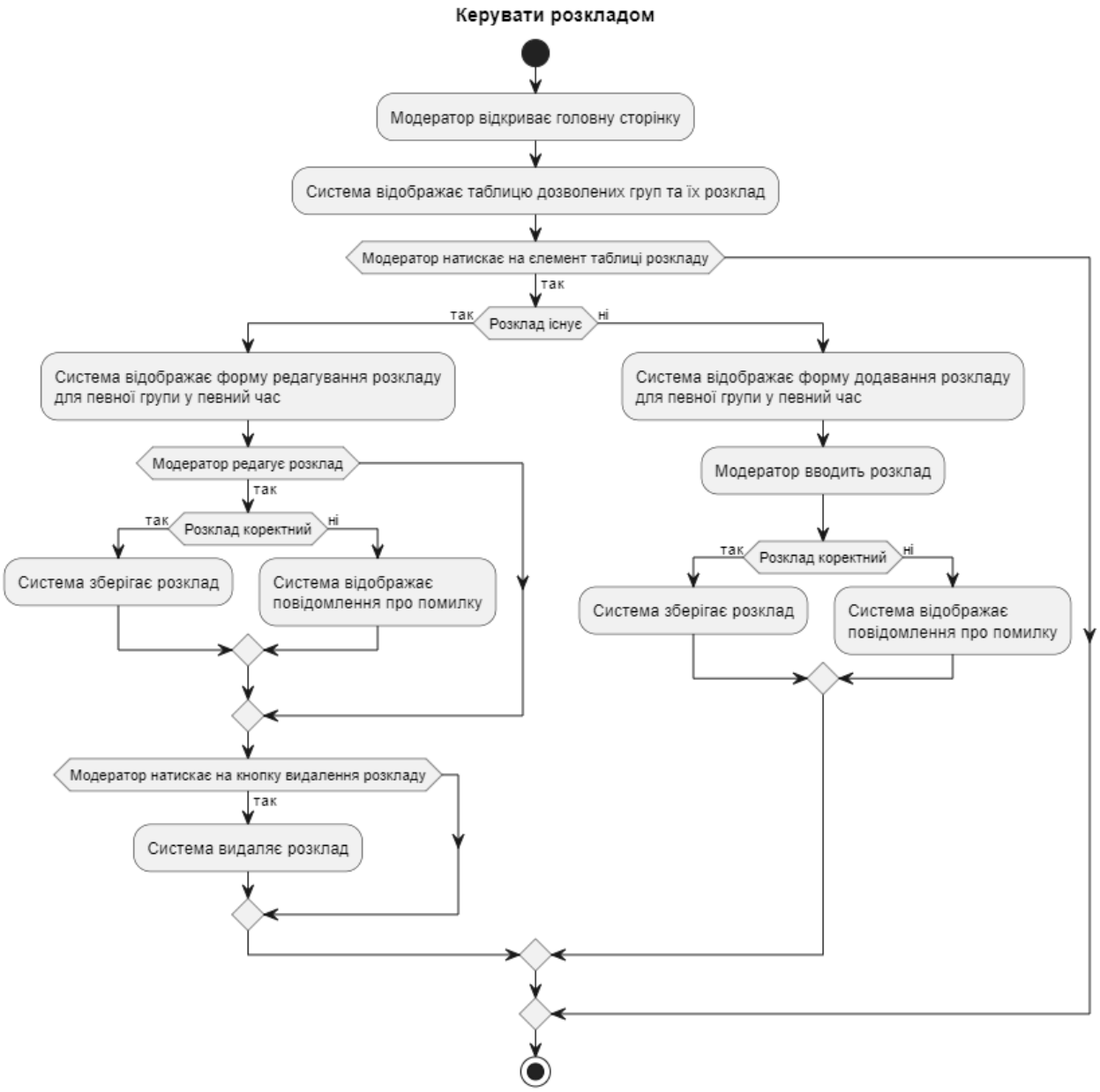

Рисунок 2.13 – Діаграма діяльності «Керувати розкладом із дозволених груп»

Ця діаграма демонструє описує послідовність дій модератора при керуванні розкладом веб-сервісу. Модератор відкриває головну сторінку веб-сервісу, де відображаються дозволені групи та розклади. Він може переглядати розклади для окремих груп, редагувати їх, зберігати зміни або видаляти розклад для обраної групи.

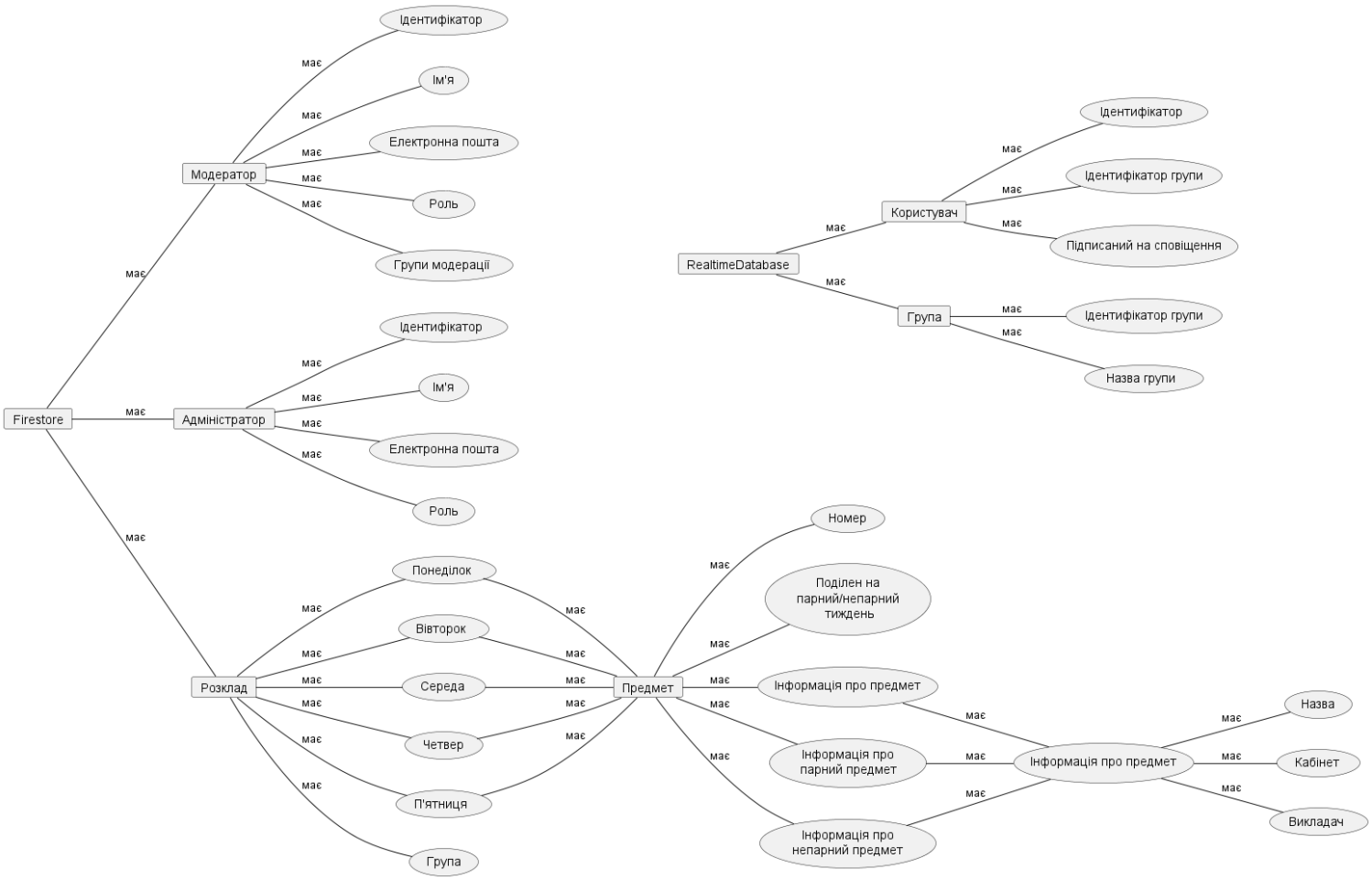

Рисунок 2.14 – Концептуальна модель даних

Ця концептуальна модель описує основні сутності, атрибути та зв'язки між ними. Модератор веб-сервісу має інформацію про ім'я, електронну пошту, ідентифікатор, роль та групи модерації. Адміністратор веб-сервісу має інформацію про ім'я, електронну пошту, ідентифікатор, роль. Розклад містить інформацію про групу та список предметів на кожен день тижня. Предмет із розкладу містить інформацію про його номер, поділ на парний/непарний тиждень та деталі предмету. Інформація про предмет відображає назву, кабінет та викладача.

2.4 Технічний проект програмного забезпечення для надання розкладу занять студентам НУК

Після ретельного аналізу початкового проекту було успішно створено комплексний технічний проект для веб-сервісу «BingNUOS», зосередившись на управлінні розкладом. Щоб точно відобразити функціональність системи, потрібно побудувати статичну та динамічну моделі. Статична модель представлена діаграмою класів, а динамічна - діаграмою послідовностей. На рисунку нижче зображено статичну модель Telegram-бота у вигляді діаграми класів.

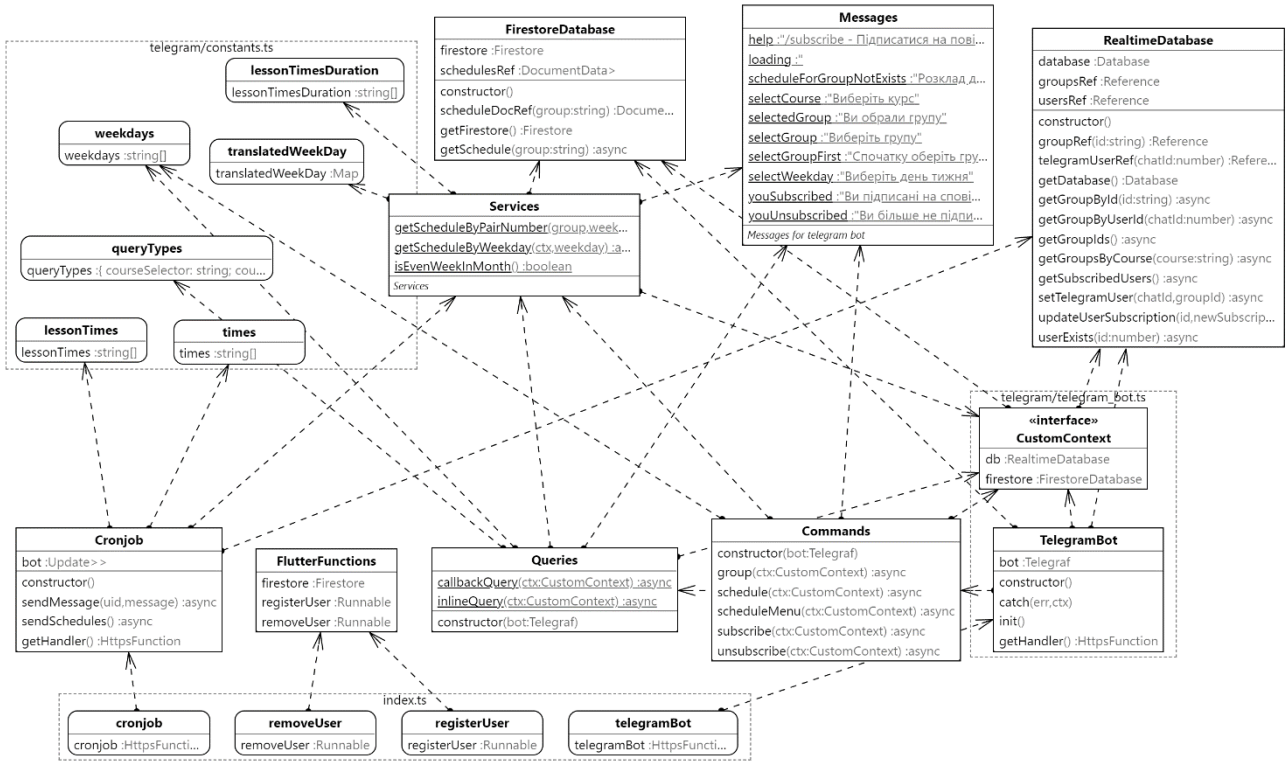

Рисунок 2.15 – Діаграма класів Telegram-бота

Специфікації основних файлів і класів Telegram-бота для надання розкладу студентам НУК:

«index.ts»: Цей файл є точкою входу в програму. Він імпортує необхідні залежності, ініціалізує Firebase функції та експортує різні функції та обробники, що використовуються Telegram-ботом.

«telegram_bot.ts»: Цей файл визначає клас TelegramBot який представляє Telegram-бота. Він ініціалізує бота, використовуючи наданий токен бота, налаштовує проміжне програмне забезпечення для додавання користувацького контексту до бота, а також ініціалізує обробники команд і запитів. Атрибути класу:

– bot: Telegraf<CustomContext> (private) - екземпляр бібліотеки Telegraf, що представляє Telegram-бота. Відповідає за обробку вхідних повідомлень та виконання відповідних дій.

– commands: Commands (private) - екземпляр класу Commands, який керує та обробляє різні команди, які може отримувати Telegram-бот.

– queries: Queries (private) - екземпляр класу Queries, який обробляє запити до бота, наприклад нажата кнопка.

– services: Services (private) - екземпляр класу Services (або його підкласу), який обробляє різні сервіси, пов'язані з Telegram-ботом.

«services.ts» Цей файл містить клас «Services», який надає різні сервіси, що використовуються Telegram-ботом. Він включає методи для отримання розкладу та перевірки парності чи непарності поточного тижня.

«queries.ts»: Цей файл визначає клас «Queries», який обробляє запити зворотного дзвінка та вбудовані запити, що надходять до Telegram-бота. Атрибути:

– db: RealtimeDatabase (protected) - екземпляр класу RealtimeDatabase, що надає доступ до бази даних реального часу. Використовується для запиту даних з бази даних.

– «commands.ts»: Цей файл містить клас «Commands», який обробляє різні команди, отримані Telegram-ботом, такі як запуск, групування, розклад, підписка та відписка. Атрибути:

– db: RealtimeDatabase (protected) - екземпляр класу RealtimeDatabase, що надає доступ до бази даних реального часу. Використовується для читання та запису даних до бази даних.

– firestore: FirestoreDatabase (protected) - екземпляр класу FirestoreDatabase, який використовується для взаємодії з базою даних Firestore.

«firestore_database.ts»: Цей файл визначає клас «FirestoreDatabase», який надає методи для взаємодії з Firestore, зокрема для отримання даних розкладу.

– firestore: Firestore (private) - екземпляр бібліотеки Firestore, що представляє базу даних Firestore. Використовується для виконання різних операцій з базою даних, таких як читання та запис даних.

– schedulesRef: CollectionReference (private) - Посилання на колекцію "schedules" у базі даних Firestore. Використовується для доступа та маніпулювання даними, пов'язаними з розкладом.

«realtime_database.ts»: Цей файл визначає клас «RealtimeDatabase», який надає методи для взаємодії з базою даних реального часу Firebase.

– db: FirebaseAdmin.db (private) - екземпляр модуля бази даних бібліотеки FirebaseAdmin, що надає доступ до бази даних у режимі реального часу. Використовується для виконання операцій з базою даних, таких як читання та запис даних.

–groupsRef: Посилання (private) - Посилання на вузол "groups" в базі даних реального часу. Використовується для доступа та маніпулювання даними, пов'язаними з групами.

Діаграма класів веб-сервісу представлено нижче на рисунку 2.16. На цій діаграмі продемонстровано взаємозв’язок класів у веб-сервісі. Ця діаграма класів представляє веб-сервіс планування з використанням мови програмування Dart та фреймворку Flutter для керування розкладом занять.

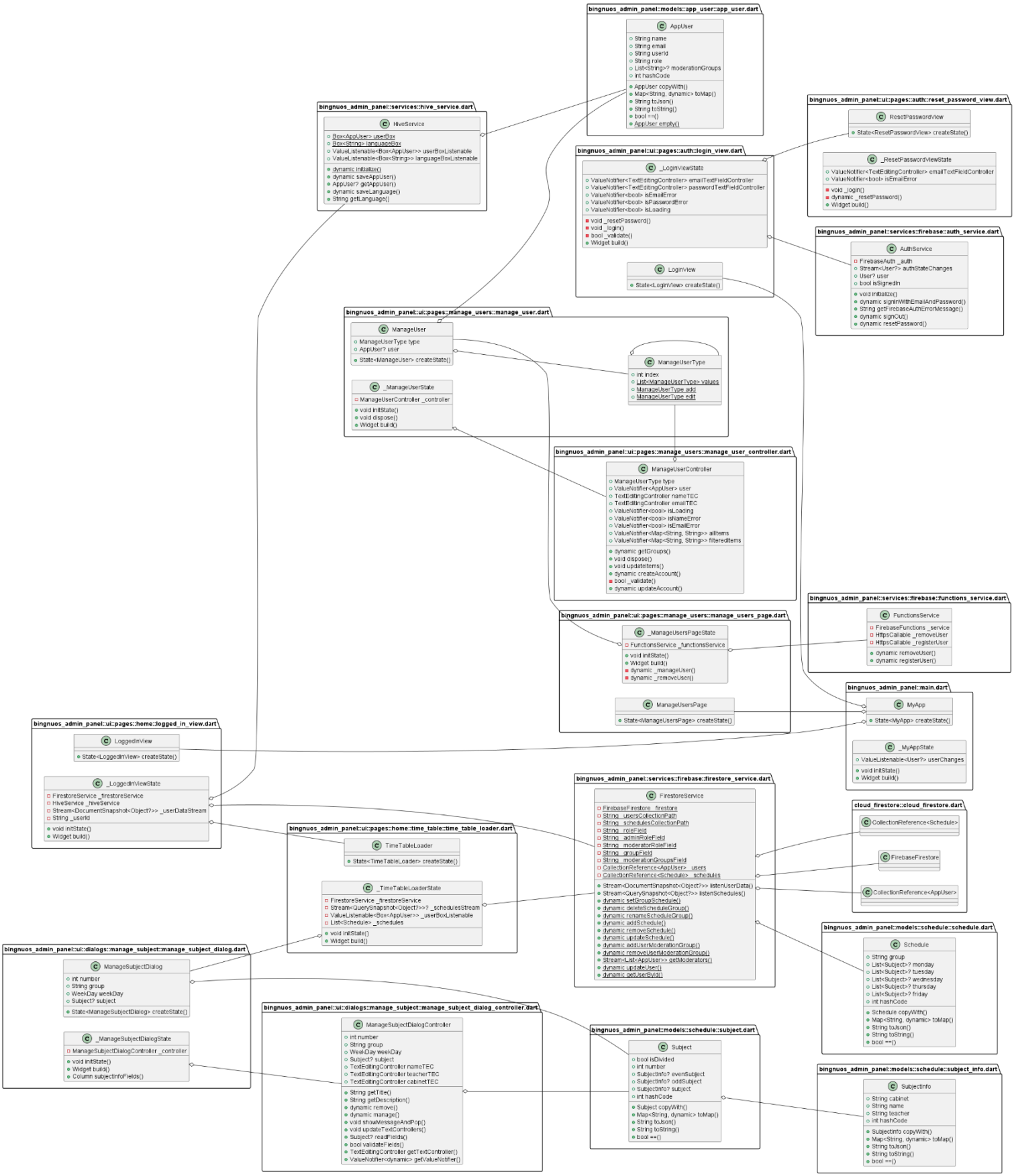

Рисунок 2.16 – Діаграма класів веб-сервісу

Специфікації основних класів веб-сервісу для керування розкладом приведено нижче:

Клас MyApp – Основна точка входу в застосунок Flutter. Створює кореневий віджет і керує станом програми. Загалом, цей клас налаштовує необхідних провайдерів, ініціалізує служби, обробляє автентифікацію, а також надає підтримку теми і локалізації для панелі адміністратора «BingNUOS». Атрибути:

– Клас MyApp: key: Об'єкт Key, який використовується для унікальної ідентифікації віджета. Він не визначений явно у фрагменті коду, але успадкований від ключового слова super у конструкторі.

– Клас _MyAppState (стан для MyApp): userChanges: Об'єкт ValueListenable<User?>, який представляє зміни у стані автентифікації користувача.

Клас LoginView: Сторінка, що представляє вікно входу в систему. Дозволяє користувачам аутентифікуватися та увійти до веб-сервісу планування. Атрибути:

– emailTextFieldController: Нотифікатор ValueNotifier, який містить контролер TextEditingController для текстового поля імейлу.

– passwordTextFieldController: Нотифікатор ValueNotifier, який містить TextEditingController для текстового поля пароля.

– isEmailError: Нотифікатор ValueNotifier, який вказує, чи є помилка при введенні адреси електронної пошти.

– isPasswordError: Нотифікатор ValueNotifier, який вказує, чи є помилка при введенні паролю.

– isLoading: ValueNotifier, який вказує, чи відбувається процес входу в систему.

Клас ResetPasswordView: Сторінка, що представляє вікно скидання пароля. Дозволяє користувачам скинути свої паролі, якщо вони їх забули. Атрибути:

– emailTextFieldController: Нотифікатор ValueNotifier, який містить контролер TextEditingController для текстового поля листа.

– isEmailError: Нотифікатор ValueNotifier, який вказує, чи є помилка при введенні імейлу.

Клас LoggedInView: Сторінка, що представляє основне подання для користувачів, які увійшли в систему. Відображає вміст і функціональність веб-сервісу планування. Атрибути:

– _firestoreService: Екземпляр FirestoreService, який використовується для взаємодії з Firestore для отримання даних користувача.

– _hiveService: Екземпляр HiveService, який використовується для збереження даних користувача у Hive.

– _userDataStream: Потік DocumentSnapshot, що представляє дані користувача, отримані з Firestore.

– _userId: Рядкова змінна, що представляє ідентифікатор користувача, отриманий від AuthService.

Ці атрибути використовуються для отримання та обробки даних користувача з Firestore, збереження їх у Hive та відображення сторінки.

Клас ManageUsersPage: Сторінка, що представляє сторінку керування користувачами. Він дозволяє адміністраторам додавати, редагувати та видаляти облікові записи користувачів. Атрібути:

– _functionsService: Екземпляр FunctionsService, який використовується для взаємодії з Firebase Cloud Functions для реєстрації або видалення модераторів.

Клас ManageUser: Сторінка, що представляє окремий обліковий запис користувача на сторінці управління користувачами. Його можна використовувати для додавання або редагування інформації про користувача. Атрібути:

– type: Значення зчислення ManageUserType, що вказує на те, чи користувач додається, чи редагується.

– user: Екземпляр AppUser, що представляє користувача, яким ви керуєте. Може бути нульовим при додаванні нового користувача.

Клас ManageSubjectDialog: Сторінка діалогу для керування темами в розкладі. Дозволяє адміністраторам додавати, редагувати або видаляти теми. Атрибути:

– number: Ціле число, що представляє номер предмета.

– group: Рядок, що представляє групу суб'єкта.

– weekDay: Зчислення, що представляє день тижня суб'єкта.

– subject: Необов'язковий об'єкт Subject, що представляє дані суб'єкта.

– nameTEC: TextEditingController для назви предмету.

– teacherTEC: TextEditingController для імені вчителя.

– cabinetTEC: TextEditingController для номера кабінету.

– isLoading: ValueNotifier<bool>, що вказує на те, чи завантажується діалогове вікно.

– isDivided: ValueNotifier<bool>, який вказує, чи предмет поділено на парні та непарні тижні.

Клас TimeTableLoader: Сторінка, що відповідає за завантаження та відображення розкладу. Атрибути:

– _firestoreService: Екземпляр класу FirestoreService.

– _schedulesStream: Потік QuerySnapshot, що представляє дані розкладу.

– _userBoxListenable: ValueListenable Box<AppUser>, що представляє дані користувача, які зберігаються в Hive.

– _schedules: Список об'єктів Schedule, що представляють дані розкладу.

Клас AppUser: модельний клас, що представляє обліковий запис користувача з такими властивостями, як ім'я, електронна пошта, userId, роль і групи модерації (moderationGroups). Атрибути:

– name: Рядок, що представляє ім'я користувача.

– email: Рядок, що представляє електронну пошту користувача.

– userId: Рядок, що представляє ідентифікатор користувача.

– role: Рядок, що представляє роль користувача.

– moderationGroups: Nullable List<String>, що представляє групи модерації, до яких належить користувач.

Клас Schedule: Модельний клас, що представляє розклад з такими властивостями, як список груп та предметів для кожного дня тижня. Атрибути:

– group: Рядок, що представляє групу, пов'язану з розкладом.

– monday: Nullable List<Subject>, що представляє предмети, заплановані на понеділок.

– tuesday: Nullable List<Subject>, що представляє предмети, заплановані на вівторок.

– wednesday: Nullable List<Subject>, що представляє предмети, заплановані на середу.

– thursday: Nullable List<Subject>, що представляє предмети, заплановані на четвер.

– friday: Nullable List<Subject>, що представляє предмети, заплановані на п'ятницю.

Клас Subject: Типовий клас, що представляє предмет у розкладі. Він містить таку інформацію, як назва предмета, вчитель, кабінет і те, чи розклад поділений. Атрибути:

– isDivided: Стовпчик, який вказує на те, чи розділений об'єкт на парні та непарні екземпляри.

– number: Ціле число, що представляє номер предмета.

– evenSubject: Об'єкт SubjectInfo з нульовим значенням, що представляє інформацію про парний екземпляр суб'єкта.

– oddSubject: Об'єкт SubjectInfo з нульовим значенням, що представляє інформацію про непарний екземпляр суб'єкта.

– subject: Об'єкт SubjectInfo, який може бути змінений на нуль, що представляє інформацію про суб'єкта, коли він не розділений.

Клас SubjectInfo: Типовий клас, що представляє додаткову інформацію про предмет, таку як назва предмета, вчитель та кабінет. Атрибути:

– cabinet: Рядок, що представляє інформацію про кабінет, пов'язану з предметом.

– name: Рядок, що представляє назву предмета.

– teacher: Рядок, що представляє ім'я викладача для даного предмету: Рядок, що представляє ім'я викладача для даного предмету.

Клас AuthService: Сервісний клас, що відповідає за операції, пов'язані з автентифікацією, такі як вхід, вихід та скидання паролів. Атрибути:

– _auth: Екземпляр FirebaseAuth, що використовується для автентифікації.

– authStateChanges: Потік<User?>, який випромінює зміни стану автентифікації.

Клас FirestoreService: Сервісний клас, який взаємодіє з базою даних Firestore. Він надає методи для отримання та оновлення даних про користувачів та розклад. Атрибути:

– _firestore: Екземпляр FirebaseFirestore, який використовується для взаємодії з базою даних Firestore.

– _usersCollectionPath: Рядок, що представляє шлях до колекції "users" у Firestore.

– _schedulesCollectionPath: Рядок, що представляє шлях до колекції "schedules" у Firestore.

– _roleField: Рядок, що представляє назву поля для поля "role" у Firestore.

– _adminRoleField: Рядок, що представляє значення для ролі "admin" у Firestore.

– _moderatorRoleField: Рядок, що представляє значення ролі "модератор" у Firestore.

– _groupField: Рядок, що представляє назву поля для поля "група" у Firestore.

– _moderationGroupsField: Рядок, що представляє назву поля для поля "moderationGroups" у Firestore.

– _users: CollectionReference<AppUser>, що представляє посилання на колекцію "users" у Firestore.

– _schedules: CollectionReference<Schedule>, що представляє посилання на колекцію "schedules" у Firestore.

Клас FunctionsService: Сервісний клас для виклику хмарних функцій Firebase Cloud Functions. Він надає функції для видалення та реєстрації модераторів. Атрибути:

– _service: Екземпляр FirebaseFunctions, який використовується для взаємодії з Firebase Cloud Functions.

– _removeUser: HttpsCallable, що представляє хмарну функцію для видалення користувача.

– _registerUser: HttpsCallable, що представляє хмарну функцію для реєстрації користувача.

Клас HiveService: Клас сервісу для локального зберігання даних за допомогою Hive. Він керує зберіганням поточного користувача та інших налаштувань. Атрибути:

– get userBox: Повертає екземпляр Box<AppUser>, пов'язаний з даними користувача.

– get languageBox: Повертає екземпляр Box<String>, пов'язаний з мовними даними.

Основні вихідні тексти програмних модулів програмного забезпечення наведенні у Додатку Б.

Наступний етап розробки технічного проекту програмного забезпечення, полягає в побудові динамічної моделі для розроблюваного програмного забезпечення. Використовуючи раніше розроблені моделі, зокрема модель варіантів використання та діаграму класів, будується динамічна модель. На рисунках 2.17 - 2.19 представлені діаграми послідовності для основних варіантів використання, які втілюють цю модель.

Діаграма послідовності для прецеденту «Вибір групи» представлена на рисунку 2.17.

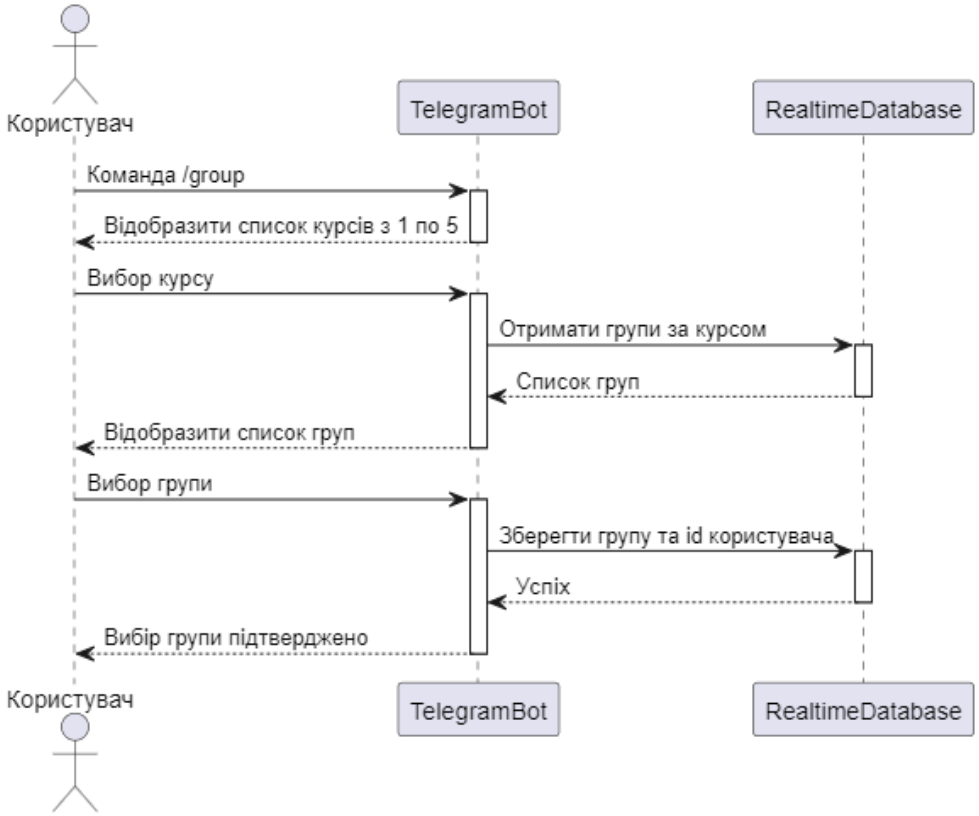

Рисунок 2.17 – Діаграма послідовності для прецеденту «Вибір групи»

Діаграма послідовності для прецеденту «Підписка на сповіщення» представлена на рисунку 2.18.

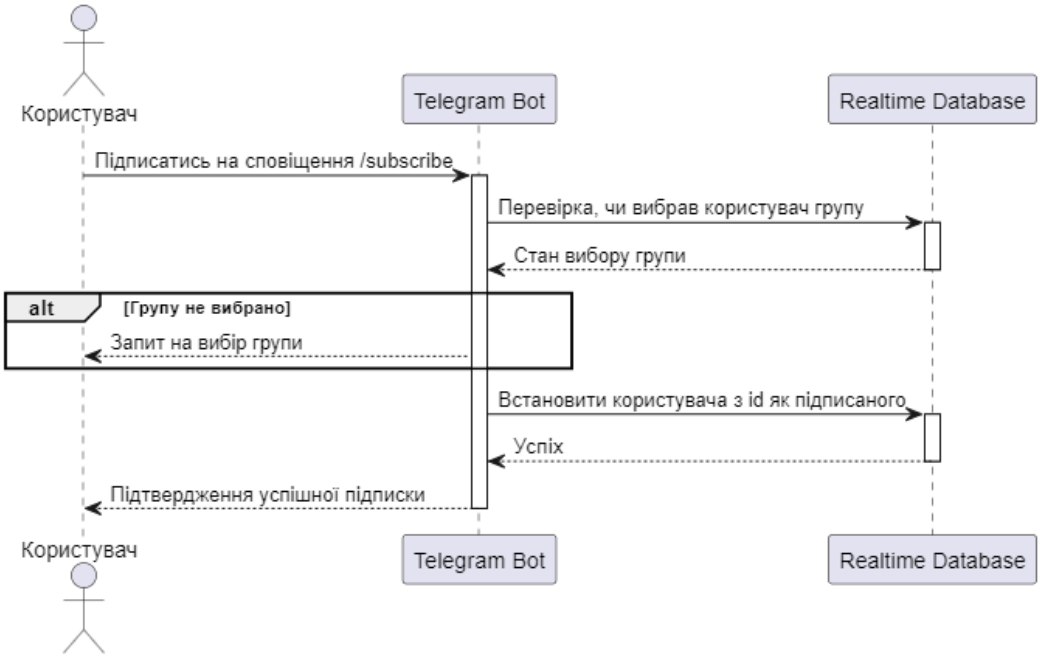

Рисунок 2.18 – Діаграма послідовності для прецеденту «Підписка на сповіщення»

Діаграма послідовності для прецеденту «Меню з вибором дня тижня для отримання розкладу» представлена на рисунку 2.19.

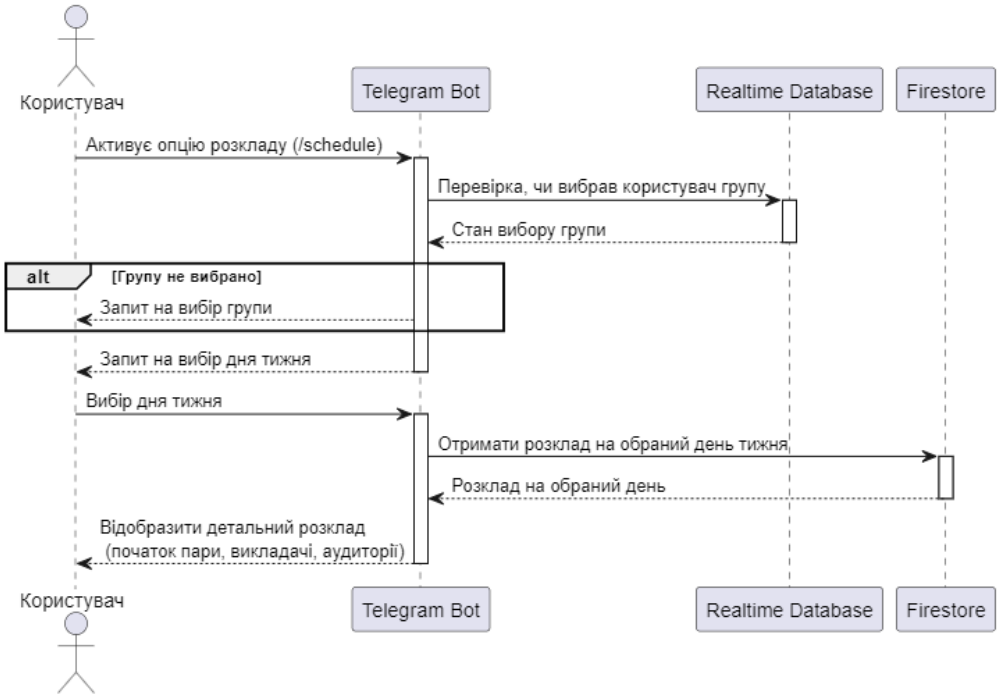

Рисунок 2.19 – Діаграма послідовності для прецеденту «Меню з вибором дня тижня для отримання розкладу»

Логічна модель даних програмного забезпечення представлена на рисунку 2.20.

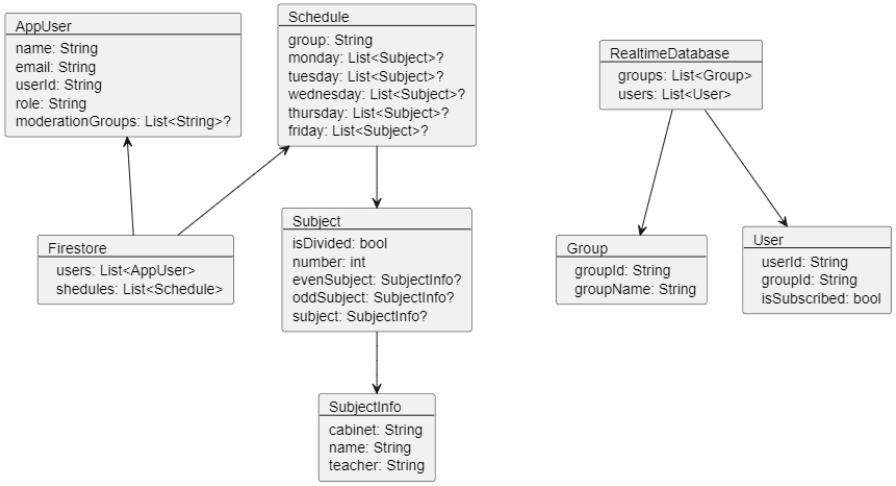

Рисунок 2.20 – Логічна модель даних

Створена логічна модель даних програмного забезпечення ґрунтується на концептуальній моделі даних. Ця модель відображає структуру бази даних.

2.5 Реалізація програмного забезпечення для надання розкладу занять студентам НУК

При розробці програмного забезпечення «BingNUOS» для надання розкладу занять студентам НУК будуть використані наступні технології та інструменти:

– Visual Studio Code буде використовуватися як середа розробки. Це легкий та потужний редактор коду, розроблений компанією Microsoft. Він підтримує різні мови програмування та забезпечує широкі можливості для редагування, налагодження і розробки програмного коду.

– TypeScript буде використовуватися як основна мова програмування для розробки Telegram-бота, який буде розгорнуто як функція Firebase. Файли TypeScript (.ts) будуть написані для визначення функціональності бота.

– Telegram-бот буде розроблений за допомогою Node.js, який виступатиме середовищем виконання для виконання коду на TypeScript. Бібліотека telegraf для Node.js буде встановлена як залежність за допомогою npm (Node Package Manager). Бібліотека надає набір функцій та утиліт для створення Telegram-ботів за допомогою Node.js. Вона буде використана для реалізації функціональності Telegram-бота, включаючи обробку команд користувача, надсилання сповіщень та взаємодію з базою даних Firebase в режимі реального часу.

– Firebase буде використаний як хмарний бекенд-сервіс для проекту. Firebase CLI (інтерфейс командного рядка) буде використаний для налаштування та управління проектом Firebase.

– Платформа Firebase Functions, яка дозволить створювати та розгортати функції в хмарному середовищі Firebase.

– База даних Firebase Realtime Database буде використана для зберігання та отримання даних щодо користувачів Telegram-бота та списку груп.

– Сервіс cron-job.org буде використаний для виклику певної Firebase функції для надсилання розкладу у заданий час.

– Веб-сервіс для планування розкладів буде розроблений з використанням фреймворку Flutter. Буде встановлено Flutter SDK, а команди Flutter будуть використані для створення, побудови та запуску проекту Flutter.

– Мова програмування Dart буде використана в поєднанні з Flutter для розробки веб-сервісу планування. Файли Dart (.dart) будуть написані для визначення компонентів інтерфейсу користувача, обробки взаємодії з користувачем і зв'язку з базою даних Firebase в режимі реального часу.

– База даних Firebase Firestore буде використана для зберігання та отримання даних розкладу та користувачів веб-сервісу в режимі реального часу.

– Firebase Hosting буде використано для розгортання веб-сервісу планування.

– Firebase Authentication буде використано для автентифікації та авторизації користувачів веб-сервісу.

Проект «BingNUOS» використовує дві бази даних - Firestore і Realtime Database - для різних компонентів системи. Firestore використовується для веб-сервісу планування розкладів, де зберігаються дані про розклади, групи та користувачів. Firestore надає швидкий доступ до даних та масштабованість для багатьох запитів. З іншого боку, Realtime Database використовується для зберігання та отримання даних, пов'язаних з користувачами Telegram-бота. Ця база даних працює в режимі реального часу і забезпечує миттєве оновлення даних. Розділення баз даних дозволяє оптимізувати та ефективно використовувати ресурси для кожного компонента системи, знижуючи витрати та забезпечуючи оптимальну продуктивність та ефективність роботи обох компонентів.

Telegram-бот «BingNUOS» повинен бути створений за допомогою бота BotFather у платформі Telegram, яка надає унікальний токен для бота. На рисунку 2.21 представлено створення бота.

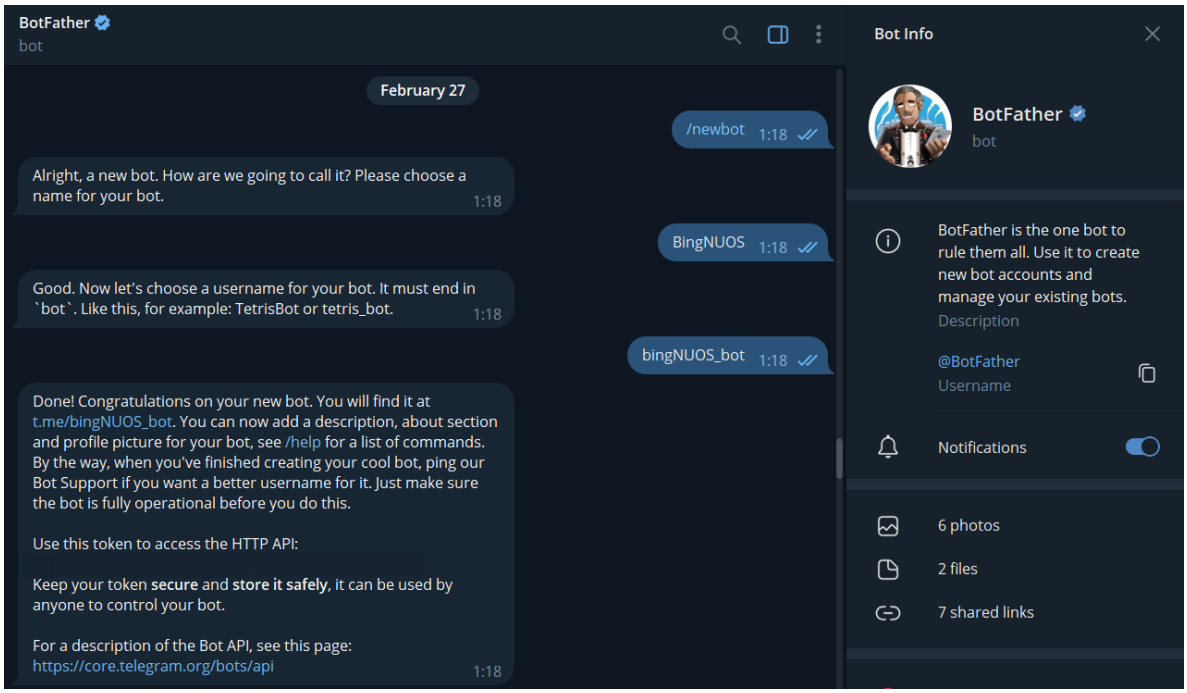

Рисунок 2.21 – Створення бота «BingNUOS»

Токен бота, отриманий від BotFather, потрібно налаштувати у коді TypeScript наступним чином: const bot = new Telegraf(токен_бота);

За допомогою інтерфейсу командного рядка (CLI) Firebase потрібно ініціалізувати проект «BingNUOS» використовуючи команду firebase init. Ця команда ініціалізує проект Firebase, де потрібно вибрати «Налаштувати каталог Cloud Functions», створити новий проект Firebase із назвою BingNUOS та вибрати мову функцій як TypeScript. Бібліотека Telegraf, яку потрібно попередньо встановити як залежність, використовується для створення нового екземпляра Telegram-бота з використанням наданого токена бота. Для інтеграції Telegram-бота з функціями Firebase було реалізовано наступний фрагмент коду:

```
functions.region("europe-west1").https.onRequest((req, res) => {{
   this.bot.handleUpdate(req.body, res);
});
```

Цей код встановлює HTTP-функцію Firebase в регіоні "europe-west1". Коли до цієї функції робиться HTTP-запит, вона запускає бота для обробки оновлення, що міститься в тілі запиту. Для розгортання коду TypeScript у

функції Firebase, код було транскрибовано в JavaScript за допомогою команди збірки компілятора TypeScript (tsc). Після того, як код JavaScript був згенерований, він був розгорнутий у функції Firebase за допомогою команди Node.js – firebase deploy. Після розгортання функції URL-адреса функції була скопійована з Firebase, а веб-хук Telegram-бота був налаштований за допомогою наступної команди curl:

curl -F "url=https://europe-west1-<ід_проекту>.firebaseapp.com/telegramBot" https://api.telegram.org/bot<токен_бота>/setWebhook

Ця команда налаштовує Telegram-бота на надсилання оновлень на вказану URL-адресу, яка є кінцевою точкою HTTP-функції Firebase. Далі потрібно налаштувати команди для вибору групи та отримання розкладу із Firestore. Розклад повинен оновлюватися за допомогою веб-сервісу планування, який буде розроблено надалі. Виконавши ці кроки, Telegram-бота було успішно інтегровано з функціями Firebase і налаштовано на обробку вхідних оновлень від користувачів.

База даних Realtime Database для Telegram-бота має вигляд JSON та повинна складається з двох основних розділів: «groups» та «users».

Розділ «groups»:

– «docId» Рядковий унікальний ідентифікатор документа у базі даних Firestore, що представляють різні групи.

– «group» Рядок, що містить номер групи.

Розділ «users»:

– «chatId» Рядковий ідентифікатор користувача в чаті Telegram-бота.

– «group» Рядок, що містить групу, до якої належить користувач, посилаючись на відповідний ідентифікатор документа групи в розділі «groups».

– «subscribed» Логічне значення, яке вказує, чи користувач підписаний (true) або не підписаний (false) на сповіщення.

На рисунку 2.23 представлено базу даних Firebase Realtime Database

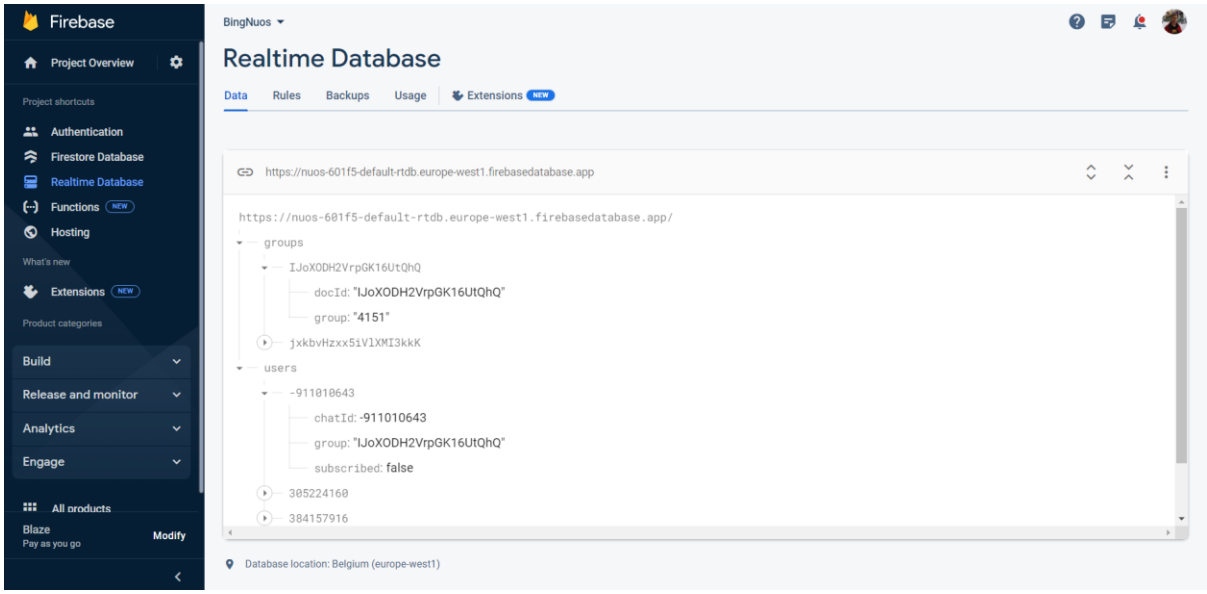

Рисунок 2.22 – База даних Realtime Database у Firebase

Також повинно розробити Firebase-функцію яка викликається веб-сервісом cron-job.org для надсилання розкладу підписаним користувачам за 10 хвилин до початку занять. На рисунку 2.22 представлено веб-сервіс cron-job.org із налаштованими завданнями.

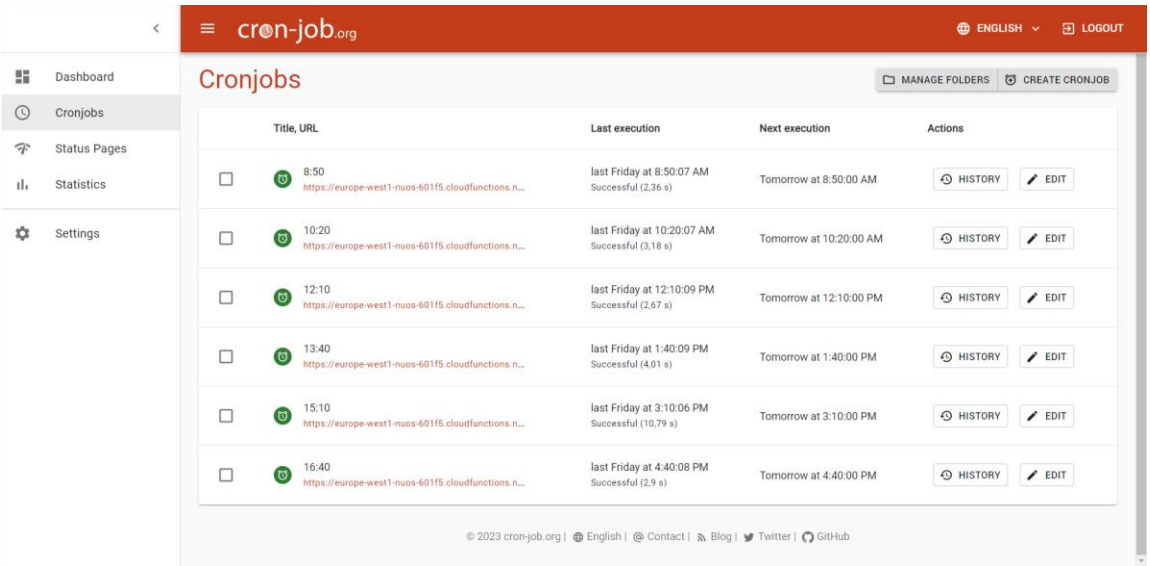

Рисунок 2.23 – Веб-сервіс cron-job.org

У цьому веб-сервісі було налаштовано виклики Firebase-функції за певним URL-адресом у 8:50, 10:20, 12:10, 13:40, 15:10, 16:40 кожен будній день.

Для розробки веб-сервісу планування «BingNUOS» повинен бути використаний фреймворк Flutter. Для ініціалізації проекту Flutter та налаштування необхідних залежностей будуть виконані наступні кроки:

– Flutter SDK повинен бути завантажений та встановлений для розробки.

– Необхідно створити новий проект Flutter за допомогою команди «flutter create». Це створить базову структуру проекту та файли.

– До файлу «pubspec.yaml» проекту повіни бути додані необхідні залежності для інтеграції з Firebase. Сюди увійдуть пакети Firebase Core, Firebase Authentication, Firebase Firestore.

– Проект Firebase повинен бути налаштований у веб-сервісі Firebase. Конфігураційні файли проекту необхідно завантажити та додати до проекту Flutter.

– Потрібно увімкнути необхідні служби Firebase, такі як Firebase Authentication та Firebase Firestore.

– Потрібно написати код використовуючи мову програмування Dart для реалізації веб-сервісу планування. Це буде включати створення компонентів інтерфейсу користувача, інтеграцію служб Firebase та реалізацію функціональності веб-сервісу, такої як автентифікація, керування розкладом, групами та модераторами.

– Після завершення розробки веб-сервісу наступним кроком буде його розгортання у хостингу Firebase. Для розгортання потрібно буде ініціалізувати проект Firebase за допомогою Firebase CLI, використовуючи команду firebase init. Ця команда ініціалізує проект Firebase, де потрібно буде вибрати «Налаштування файлів для хостингу Firebase» та вибрати проект Firebase із назвою "BingNUOS". Це створить необхідні конфігураційні файли та налаштування.

– У проекті Flutter будуть створені оптимізовані веб-ресурси у каталозі збірки проекту за допомогою команди flutter build web.

– Веб-ресурси будуть розгорнуті на хостингу Firebase за допомогою команди Firebase CLI firebase deploy. Це завантажить веб-ресурси до Firebase і зробить веб-службу планування доступною через загальнодоступну URL-адресу.

База даних Firestore веб-сервісу повинна складатися із двох основних колекцій: «schedules» та «users».

Колекція «users»:

– «userId»: Рядковий ідентифікатор користувача у веб-сервісі планування

– «name»: Рядок, ім'я користувача

– «email»: Рядок, електронна адреса користувача

– «role»: Рядок, роль користувача

– «moderationGroups»: Список груп, до яких модератор має права.

Колекція «schedules»:

– «group»: Рядок, назва групи розкладу

– «monday» – «friday»: Список предметів для понеділка – п’ятниці (може бути порожнім)

Кожен елемент списку предметів повинен містити наступну інформацію:

– «isDivided»: Логічне значення яке позначає, чи є предмет поділеним на парні та непарні тижні

– «number»: Число, номер пари

– «evenSubject»: Інформація про предмет у парні тижні (може бути відсутньою, якщо предмет не поділено)

– «oddSubject»: Інформація про предмет у непарні тижні (може бути відсутньою, якщо предмет не поділено)

– «subject»: Інформація про предмет (використовується, якщо предмет не поділено)

Інформація про предмет повинна містити наступну інформацію:

– «name»: Рядок, назва предмету
– «cabinet»: Рядок, назва аудиторії, де проходить предмет
– «teacher»: Рядок, ім'я викладача, який веде предмет

На рисунку 2.24 продемонстрована база даних Firestore із колекцією «schedules» та документом для групи 3151.

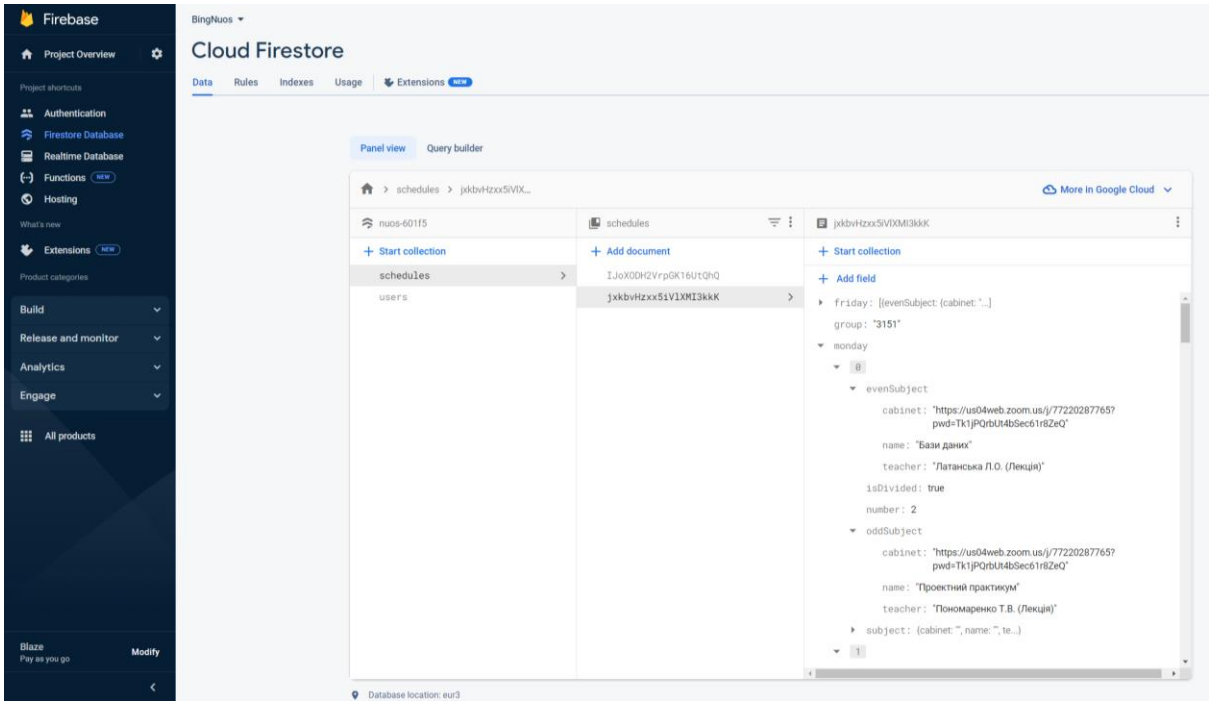

Рисунок 2.24 – База даних Firestore у Firebase

Загалом, використання цих технологій та інструментів дозволяє створити функціональне та зручне програмне забезпечення «BingNUOS», яке надає студентам НУК доступ до розкладу занять та особистих повідомлень, а також дозволяє адміністраторам та модераторам ефективно керувати розкладом занять та іншими функціями.

2.6 Тестування програмного забезпечення для надання розкладу занять студентам НУК

Під час тестування програмного забезпечення «BingNUOS» для надання розкладу занять студентам НУК були проведені наступні тести для перевірки вимог до функціональних характеристик:

– Перевірка коректності вибору групи.

– Перевірка перегляду розкладу для обраної групи.

– Перевірка підписки та відписки від сповіщень.

– Перевірка доставки сповіщень за 10 хвилин до початку кожного заняття.

– Перевірка авторизації модератора та адміністратора в веб-сервісі.

– Перевірка правильності відображення розкладу для дозволених груп для модератора.

– Перевірка додавання, редагування та видалення розкладу модератором та адміністратором.

– Перевірка правильності відображення розкладу усіх груп для адміністратора.

– Перевірка створення, редагування та видалення груп та розкладів адміністратором.

– Перевірка створення, редагування та видалення модераторів та їх груп модерації адміністратором.

У таблиці 2.12 представлено тестування програмного забезпечення.

Таблиця 2.12 – Тестування програмного забезпечення.

№	Дія	Очікуваний результат	Результат
1	2	3	4
1	Введення команди /group та вибір конкретної групи із меню Telegram-бота	Telegram-бот підтверджує вибір групи	Telegram-бот успішно підтвердив вибір групи

Продовження таблиці 2.12

1	2	3	4
2	Введення команди /schedule та вибір пункту меню із робочим днем.	Telegram-бот відображає розклад занять для вибраної групи на вибраний робочий день	Розклад занять для вибраної групи на вибраний робочий день відображається правильно
3	Введення команди /subscribe /unsubscribe для підписки або відписки від сповіщень	Telegram-бот змінює статус підписки на сповіщення	Статус підписки на сповіщення успішно змінено
4	Наявність підписки на сповіщення перед початком заняття	Telegram-бот надсилає розклад занять за 10 хвилин до початку кожного заняття	Розклад занять надсилається за 10 хвилин до початку кожного заняття
5	Введення коректних облікових даних модератора або адміністратора	Веб-сервіс авторизує модератора або адміністратора	Модератора або адміністратора успішно авторизовано
6	Авторизація модератора та відкриття головної сторінки веб-сервісу	Веб-сервіс відображає розклад для дозволених груп модератора	Розклад для дозволених груп модератора відображається правильно
7	Дії модератора або адміністратора з розкладом (додавання, редагування, видалення)	Веб-сервіс виконує відповідні дії з розкладом і зберігає зміни	Додавання, редагування, видалення розкладу виконано успішно
8	Авторизація адміністратора та відкриття головної сторінки веб-сервісу	Веб-сервіс відображає розклад для всіх груп	Розклад для всіх груп відображається правильно
9	Дії адміністратора з групами та розкладами (створення, редагування, видалення)	Веб-сервіс виконує відповідні дії з групами та розкладами і зберігає зміни	Створення, редагування, видалення груп та розкладів виконано успішно
10	Дії адміністратора з модераторами та їх групами (створення, редагування, видалення)	Програма виконує відповідні дії з модераторами та їх групами і зберігає зміни	Створення, редагування, видалення модераторів та їх груп модерації виконано успішно

В повному обсязі тестування програмного забезпечення наведено у програмі та методиці випробувань у Додатку Д.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ НАДАННЯ РОЗКЛАДУ ЗАНЯТЬ СТУДЕНТАМ НУК

У результаті проведення кваліфікаційної роботи було успішно розроблено програмне забезпечення для надання розкладу занять студентам НУК із назвою «BingNUOS». Веб-сервіс та Telegram-бот повністю відповідає в потребам та вимогам користувачів, забезпечуючи зручний та швидкий доступ до розкладу занять. Основні результати розробки програмного забезпечення «BingNUOS» включають:

– Веб-сервіс: Було розроблено веб-інтерфейс, який дозволяє користувачам зручно переглядати та отримувати розклад занять через Інтернет. Користувачі можуть вибирати курси, групи та дні тижня, щоб переглянути розклад занять. Веб-сервіс також підтримує можливість підписки на сповіщення про наближення до пари.

– Telegram-бот: Бот забезпечує студентам зручний доступ до розкладу занять через популярну платформу Telegram. Користувачі можуть вибирати курси, групи та дні тижня, отримувати детальний розклад занять, а також підписуватися на сповіщення про наближення до пари.

– Інтеграція з Firebase: Для збереження та управління даними були використані Firebase Realtime Database та Firestore. Це забезпечує надійне та швидке збереження розкладу занять, а також дозволяє оновлювати дані в реальному часі.

– Тестування та перевірка: Програмне забезпечення було піддане ретельному тестуванню для перевірки його функціональності, надійності та відповідності вимогам. Всі основні функції та сценарії використання були успішно протестовані, і відсутність помилок підтверджена.

– Успішна імплементація: Розроблене програмне забезпечення «BingNUOS» було успішно впроваджено та використовується студентами НУК. Користувачі оцінюють його зручність, доступність та надійність.

Розмір програмного забезпечення

Програмне забезпечення «BingNUOS» складається з наступних компонентів:

– Веб-сервіс складається з 5044 рядків коду, написаних мовою Dart. Розмір коду на диску становить приблизно 284 КБ. Після компіляції та зборки, розмір веб-сервісу становить 22 МБ. Головним вихідним файлом є index.html, який виконується в браузері, а також flutter.js, який містить необхідний код для ініціалізації та запуску Flutter веб-сервісу.

– Telegram-бот складається з 1003 рядків коду, написаних мовою TypeScript. Розмір коду на диску становить приблизно 60.0 КВ. Після компіляції коду TypeScript у код JavaScript розмір Telegram-бота становить 144 КВ. Головним вихідним файлом є файл ініціалізації index.js, який виконується у середовищі Node.js як Firebase функція.

Результати випробування програмного забезпечення

Програмне забезпечення «BingNUOS» було піддане випробуванням згідно програми та методикою випробувань наведених у додатку Д. Нижче наведені результати випробувань:

На рисунках 3.1 – 3.7 продемонстровано випробування Telegram-бота

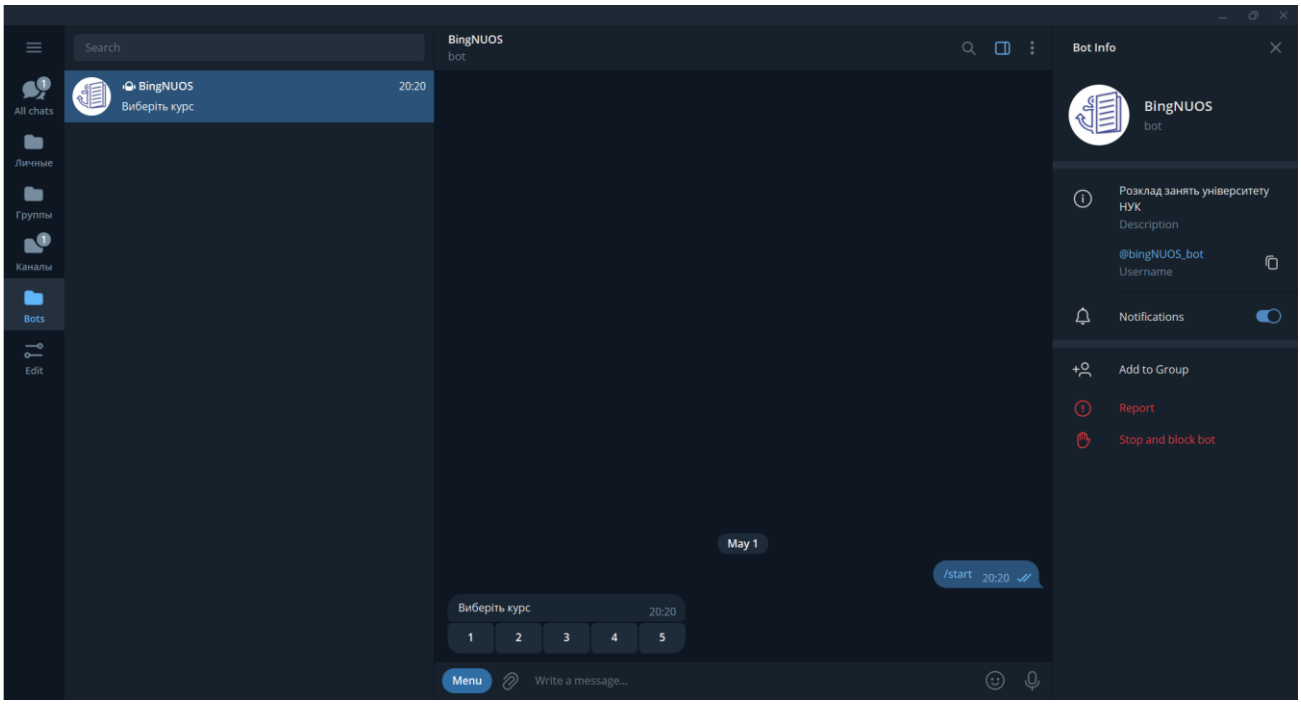

Рисунок 3.1 – Після запуску бота, бот надсилає меню з вибором курсу.

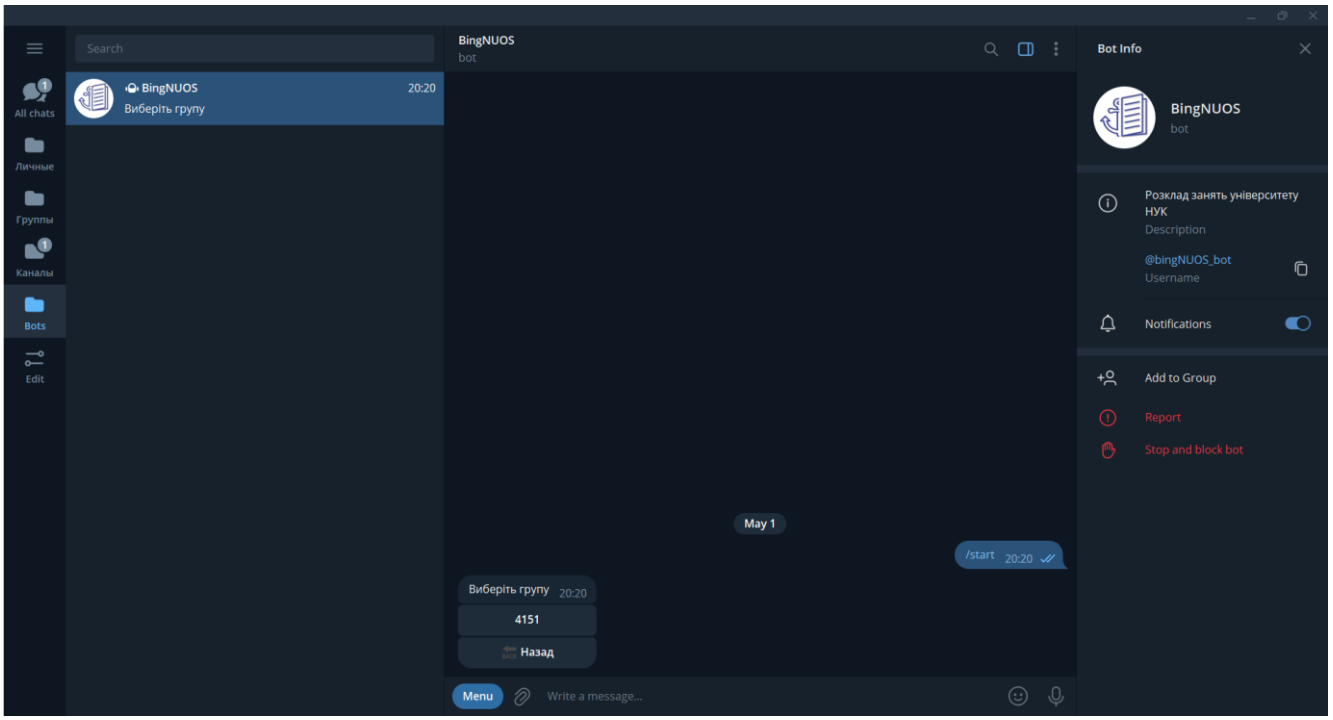

Рисунок 3.2 – Після вибору 4 курсу, бот надсилає меню з варіантами вибору групи.

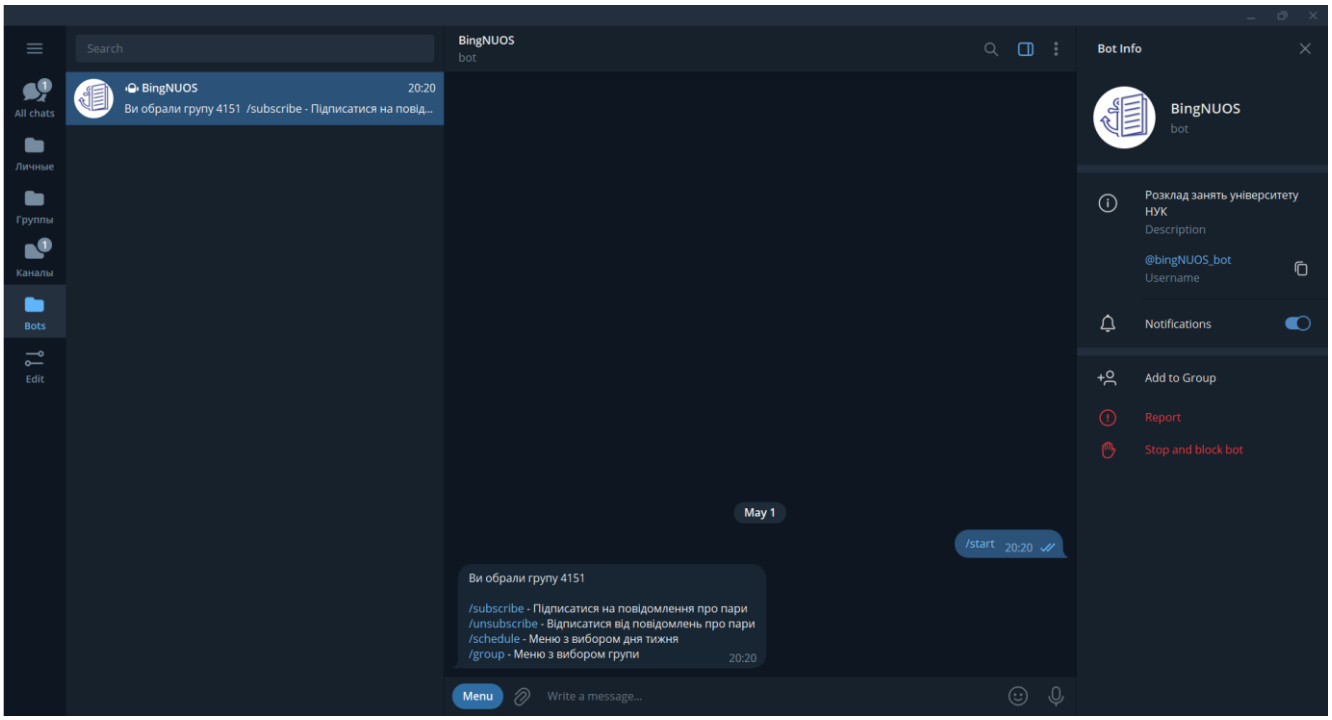

Рисунок 3.3 – Після вибору групи 4151, бот надсилає підтвердження вибору та список основних команд.

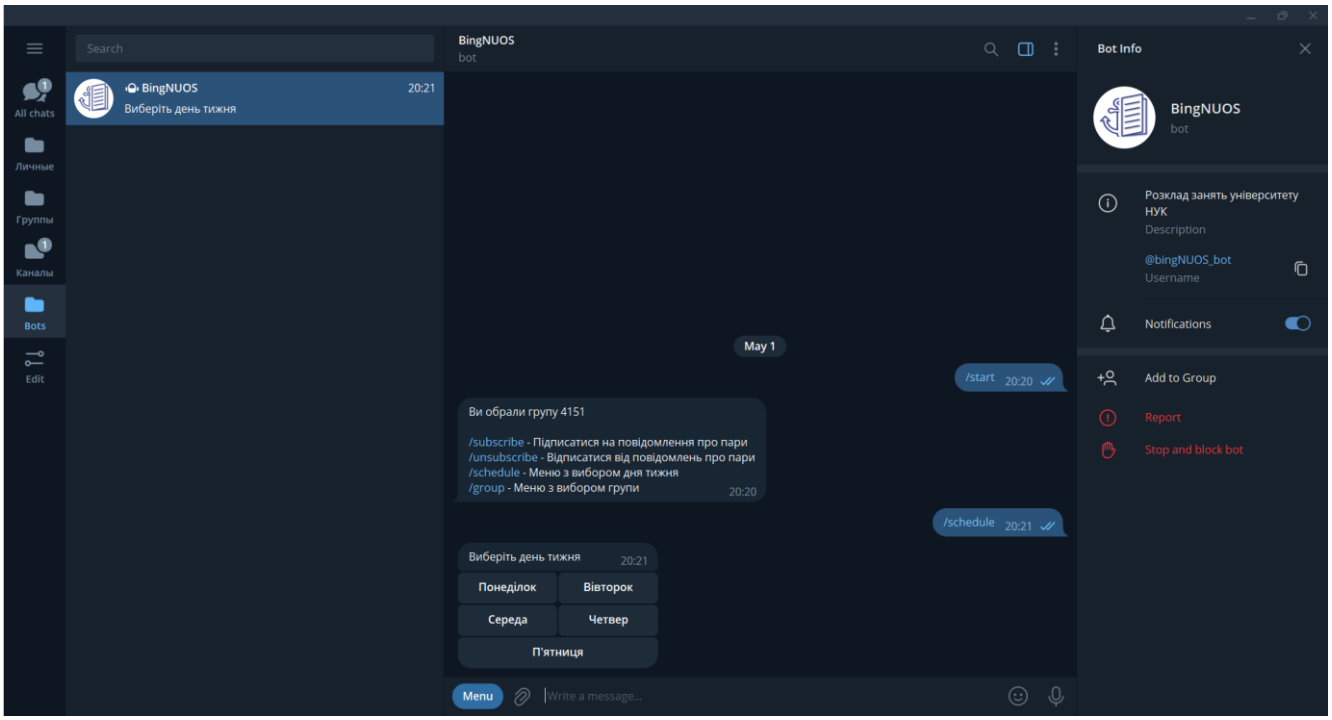

Рисунок 3.4 – Після виклику меню з вибором дня тижня за допомогою команди /schedule, бот надсилає меню із списком днів тижня.

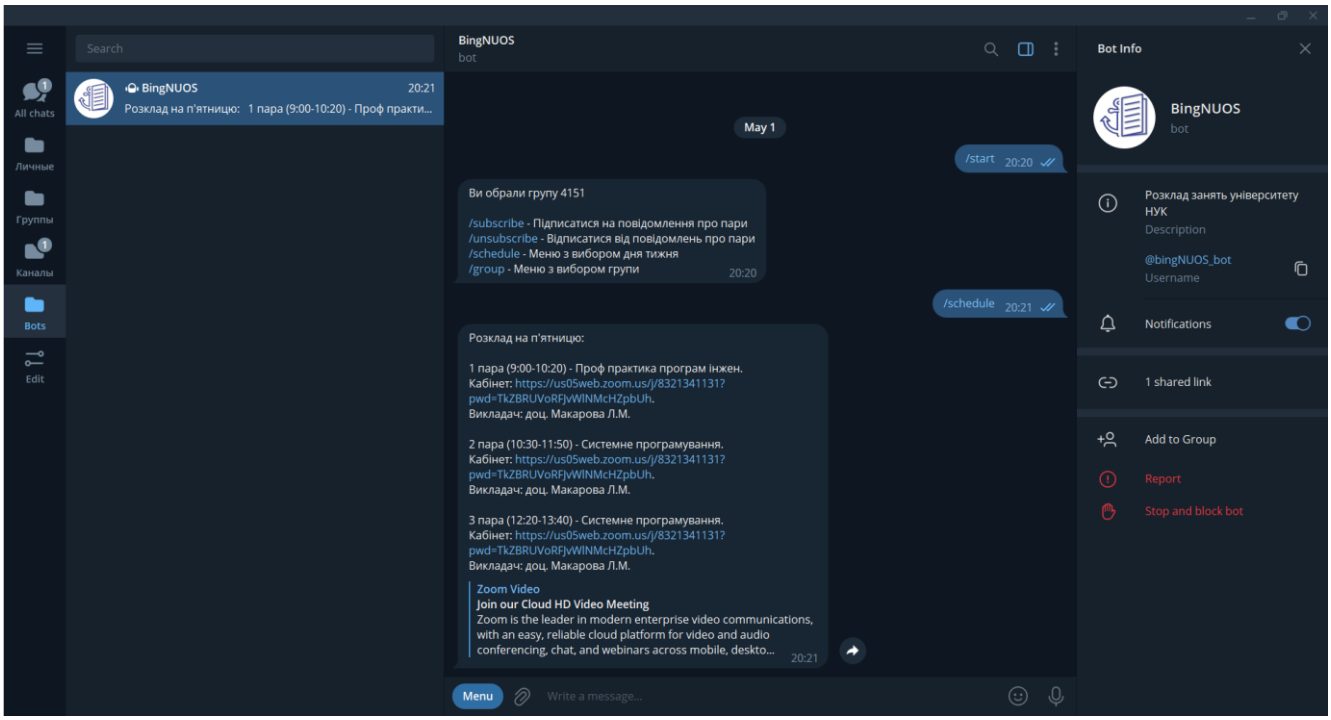

Рисунок 3.5 – Після вибору п'ятниці у меню, бот надсилає детальний розклад занять.

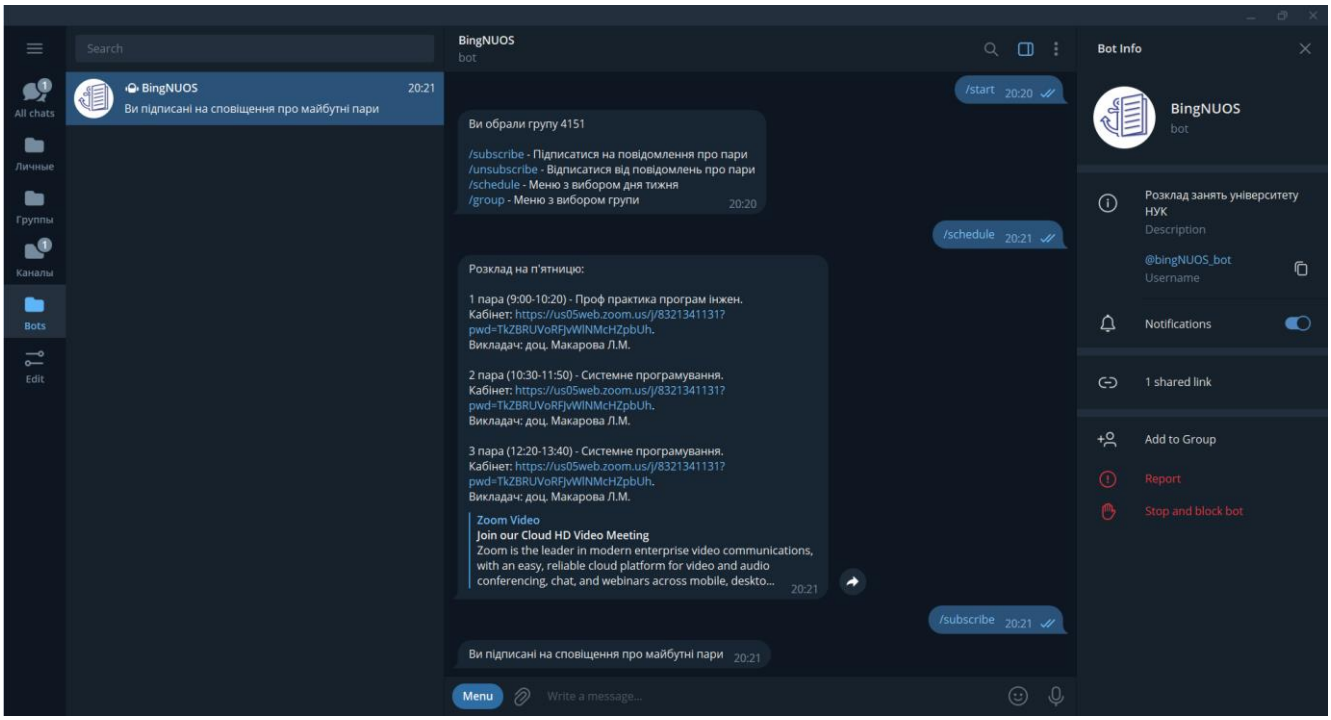

Рисунок 3.6 – Після підписки на сповіщення за допомогою команди /subscribe, бот надсилає повідомлення з підтвердженням.

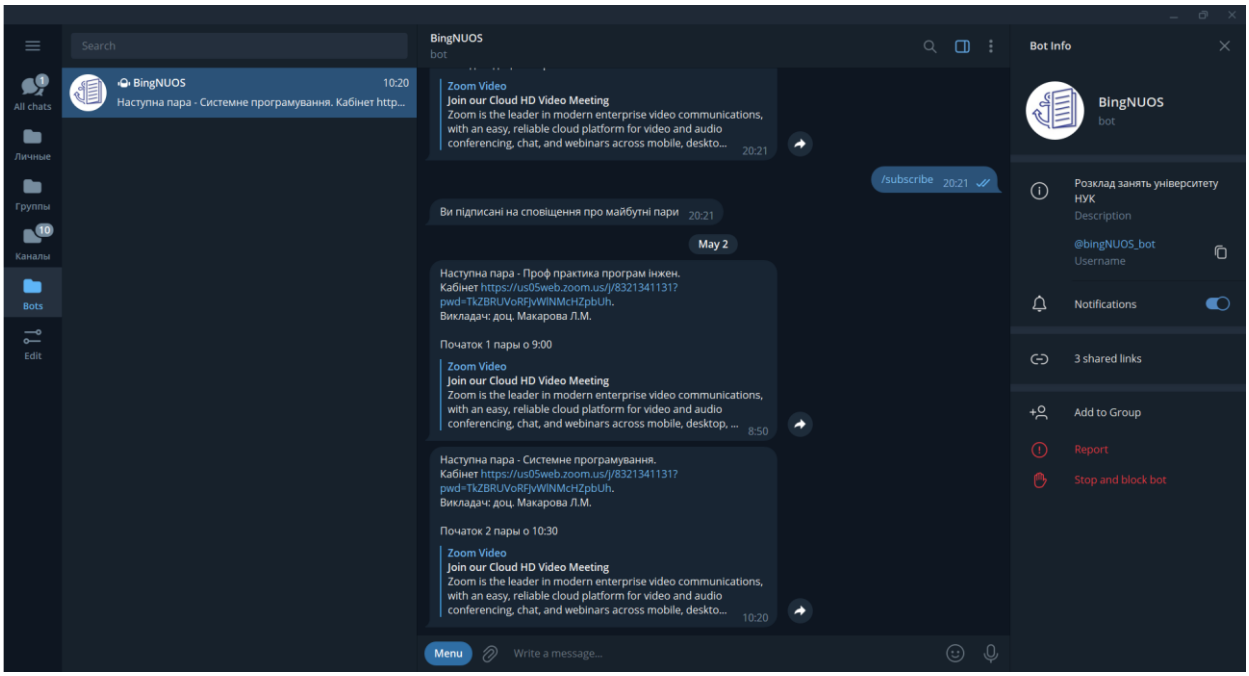

Рисунок 3.7 – На наступний день, за 10 хвилин до початку наступної пари, бот надсилає повідомлення з інформацією про цю пару.

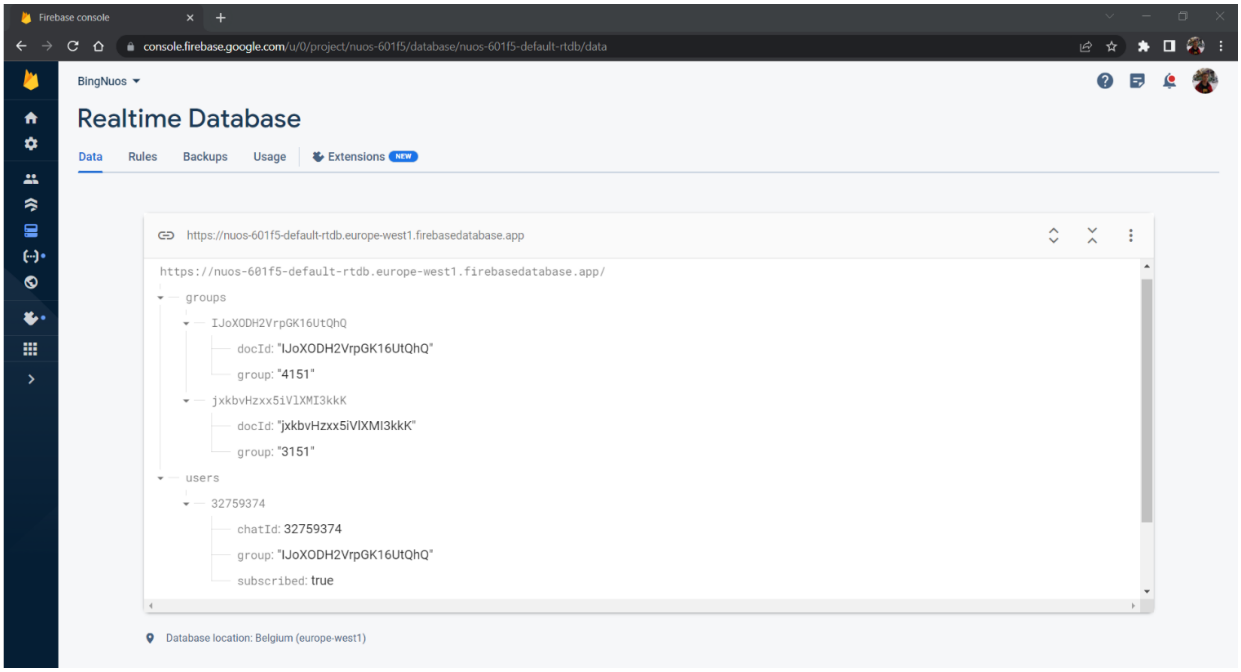

Рисунок 3.8 – Після певної дії, база даних Realtime Database оновлює користувача.

У результаті випробування функціональності програмного забезпечення «BingNUOS» підтверджено, що всі основні функції, такі як вибір курсів, груп,

днів тижня, перегляд розкладу занять та підписка на сповіщення про наближення до пари, працюють належним чином.

На рисунках 3.9 – 3.13 продемонстровано результати випробування веб-сервісу планування.

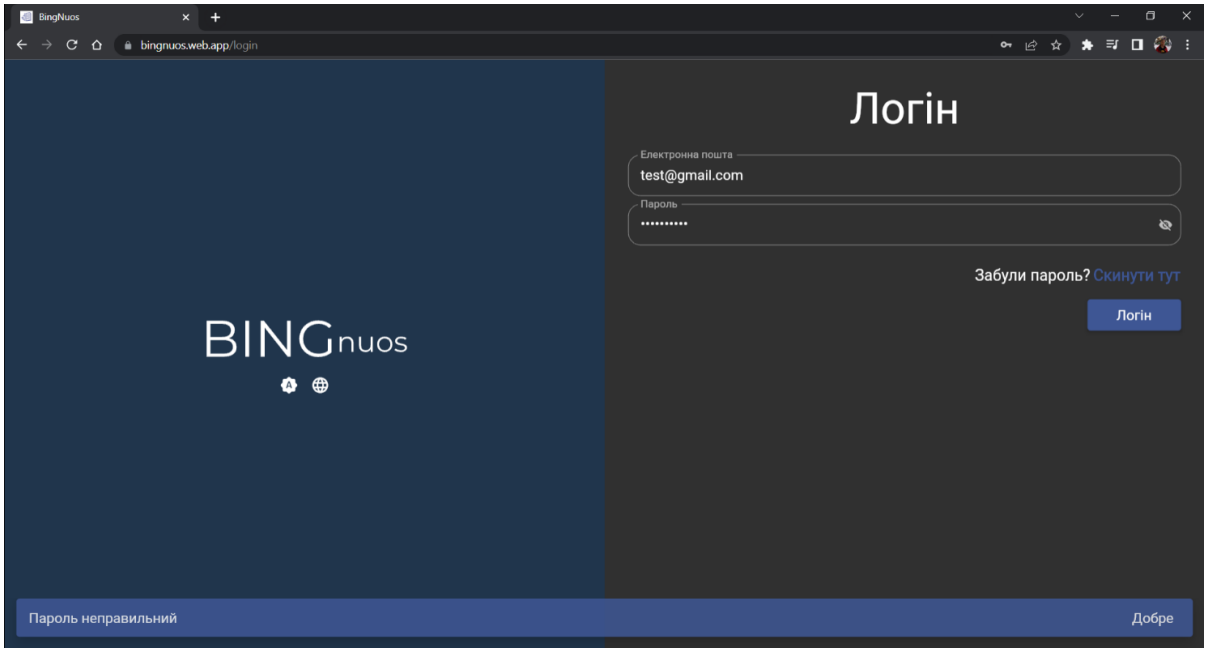

Рисунок 3.9 – Після неправильного вводу електронної пошти або паролю на сторінці "Логін" веб-сервісу, з'являється відповідне повідомлення.

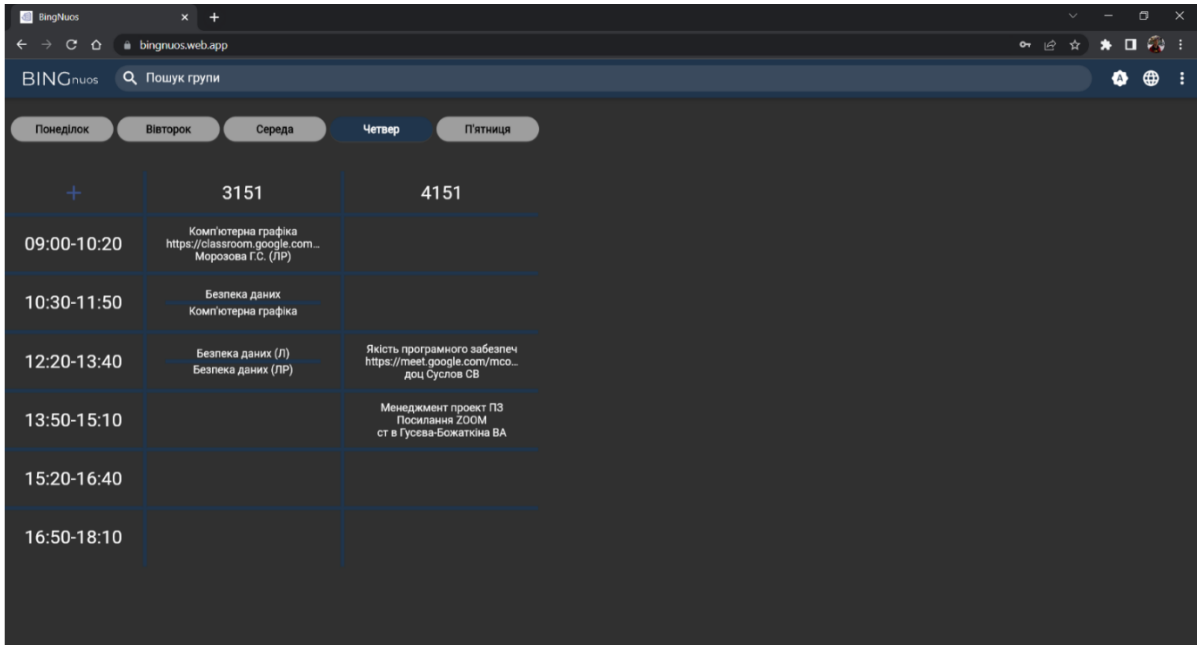

Рисунок 3.10 – На головній сторінці веб-сервісу відображаються наступні елементи: розклад груп, рядок пошуку груп, перемикач дня тижня.

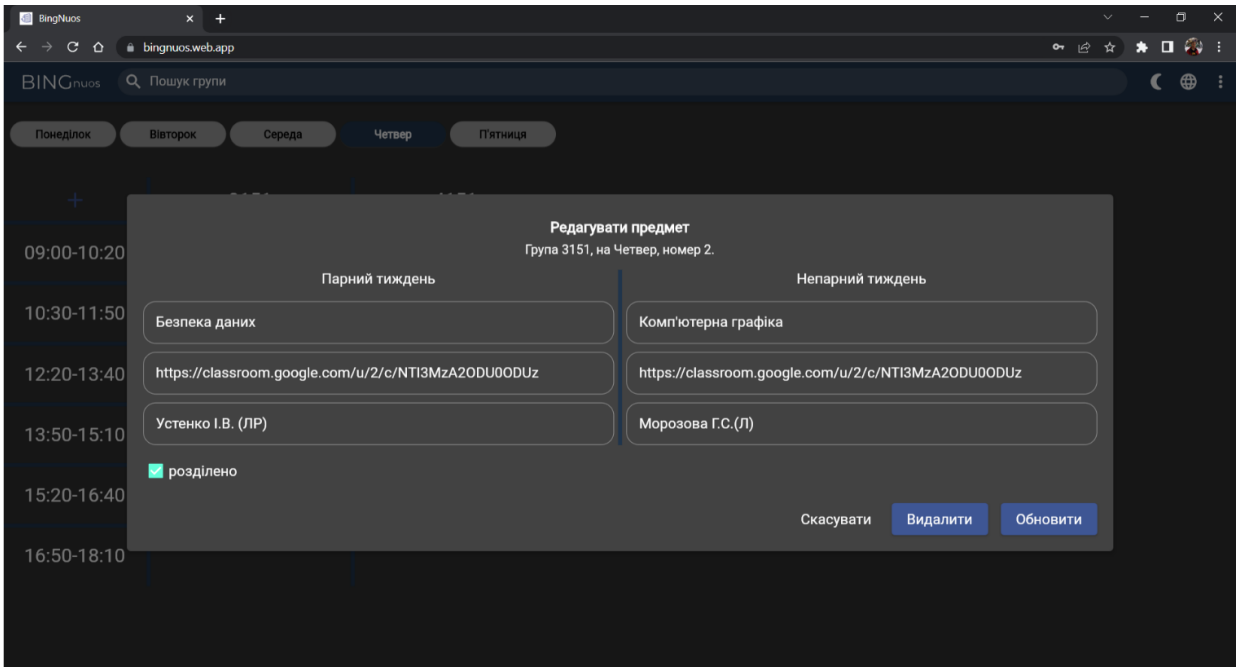

Рисунок 3.11 – При натисканні на предмет на головній сторінці веб-сервісу, відображається вікно редагування цього предмета.

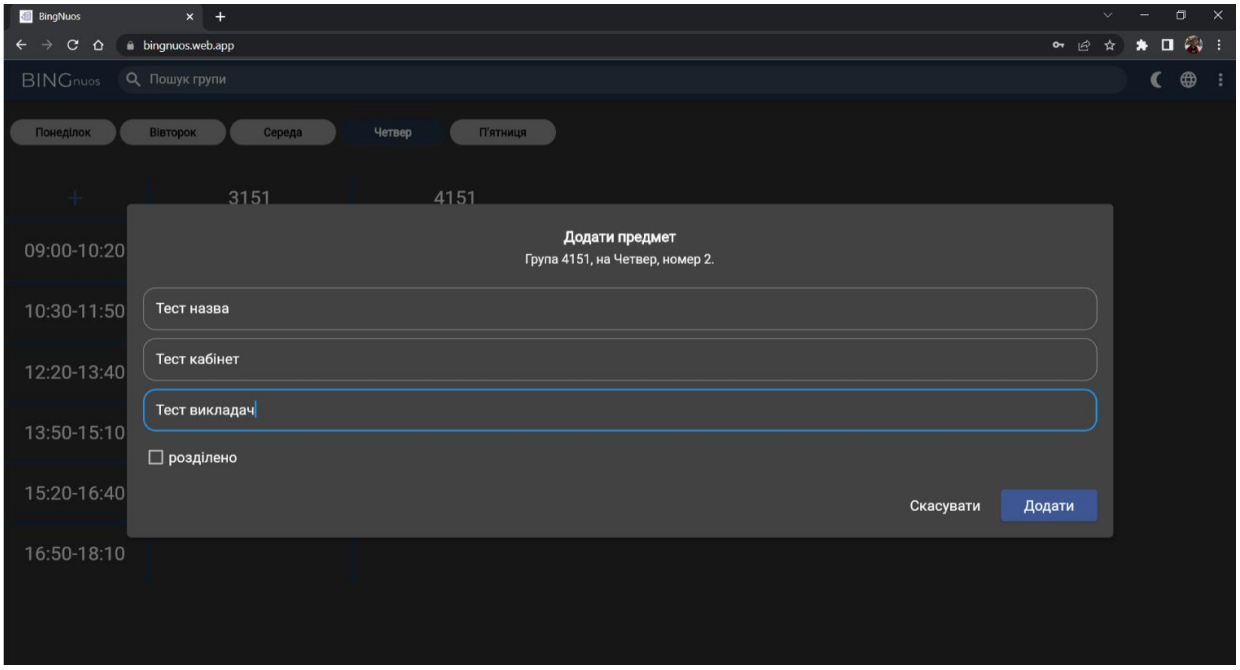

Рисунок 3.12 – При натисканні на пусте поле на головній сторінці веб-сервісу, відображається вікно додавання нового предмета.

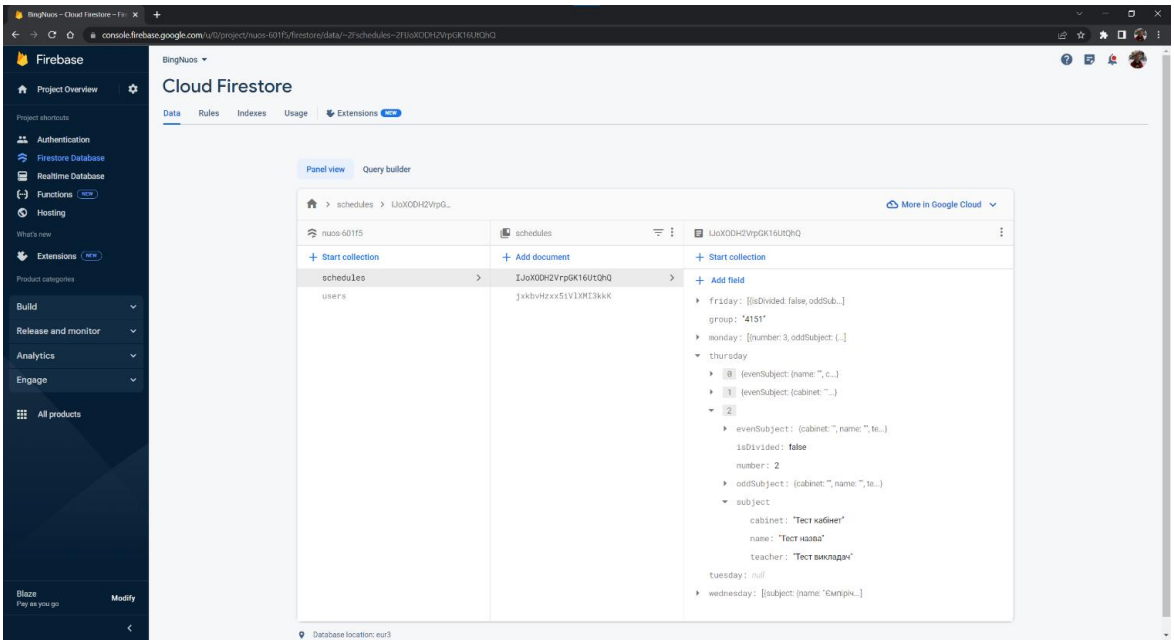

Рисунок 3.13 – Після додавання предмету, база даних Firestore оновлює дані.

Під час випробування функціональності веб-сервісу «BingNUOS» підтверджено:

– Відображення відповідного повідомлення при неправильному вводі електронної пошти або паролю на сторінці "Логін" веб-сервісу.

– Правильне відображення розкладу груп, рядка пошуку груп, перемикача дня тижня, тем та мови, а також кнопки меню на головній сторінці.

– Відображення вікна редагування предмета при натисканні на нього.

– Відображення вікна додавання предмета при натисканні на пусте поле.

– Оновлення бази даних Firestore з відповідними змінами після додавання предмета.

Виходячи з вищесказаного, можна зробити висновок, що було досягнуто мети розробки програмного забезпечення для надання розкладу занять студентам НУК, а саме створення Telegram-бота та веб-сервісу планування для редагування розкладу.

4 ОХОРОНА ПРАЦІ

Охорона праці є важливим аспектом трудової діяльності, який спрямований на забезпечення безпеки та здоров'я працівників. Вона включає в себе різні аспекти, такі як правові, соціально-економічні, організаційно-технічні, санітарно-гігієнічні та лікувально-профілактичні заходи та засоби. Основні завдання охорони праці можна розділити на інженерно-технічні та соціальні аспекти.

Інженерно-технічне завдання охорони праці спрямоване на зменшення ризиків, пов'язаних з трудовою діяльністю. Це досягається шляхом застосування нових технологій, розробки безпечних робочих процесів, використання спеціальних устаткувань та засобів захисту. Такі заходи допомагають запобігати нещасним випадкам, професійним захворюванням та іншим ризикам, пов'язаним з роботою.

Соціальне завдання охорони праці включає компенсаційні та захисні заходи, спрямовані на відшкодування шкоди, запобігання несправедливості та захист прав працівників. Це включає компенсацію за шкоду, завдану під час трудового процесу, виплату страхових виплат в разі нещасних випадків, розробку правил та норм, що стосуються безпеки праці, а також захист працівників від дискримінації та несправедливого поводження.

Охорона праці регулюється законом України «Про охорону праці», який визначає права та обов'язки працівників і роботодавців у сфері охорони праці [14]. Цей закон містить норми та вимоги, які стосуються організації робочого процесу, професійної підготовки та інструктажу працівників, використання засобів індивідуального захисту, медичного обстеження та інших аспектів, що стосуються охорони праці.

Забезпечення безпеки та охорони праці є спільною відповідальністю роботодавців, працівників та держави, і вимагає постійного вдосконалення,

навчання та контролю для забезпечення безпечних умов праці та попередження можливих ризиків.

4.1 Аналіз шкідливих та небезпечних факторів під час роботи з екранними пристроями

Аналіз шкідливих та небезпечних факторів під час роботи з екранними пристроями є важливою частиною охорони праці, оскільки тривалий час, проведений перед екраном, може викликати різні проблеми зі здоров'ям. Основні шкідливі та небезпечні фактори, пов'язані з роботою з екранними пристроями, включають:

– Вплив візуального навантаження: Постійна фокусування на екрані може призвести до втоми та напруги очей, сухості та подразнення очей, а також до розвитку офтальмологічних захворювань, таких як комп'ютерний синдром.

– Погіршення позиції тіла: Некоректна позиція тіла під час роботи з екраном може призвести до розладів опорно-рухового апарату, наприклад, до болей у шиї, плечах, спині, а також до м'язових напруг і захворювань суглобів.

– Вплив електромагнітного випромінювання: Екранні пристрої випромінюють електромагнітне випромінювання, яке може мати вплив на здоров'я працівників, включаючи можливість розвитку електромагнітної гіперчутливості.

– Погіршення психоемоційного стану: Тривала робота з екраном може спричиняти стрес, збільшену втомлюваність, розлади сну та інші психоемоційні проблеми.

– Регулювання робочого часу: Важливо встановити оптимальну тривалість роботи перед екраном і враховувати перерви для відпочинку та очей. Рекомендується регулярно проводити короткі перерви під час тривалої роботи з екраном.

– Забезпечення правильної позиції тіла: Рекомендується належно налаштувати робоче місце, включаючи висоту стільця та стола, кут нахилу екрана і клавіатури. Привертайте увагу до правильної позиції спини, шиї і рук під час роботи перед екраном.

– Організація робочого простору: Важливо забезпечити належне освітлення робочого місця, уникати відблисків на екрані та регулювати контрастність та яскравість екрану. Крім того, слід стежити за правильним розміщенням робочого обладнання та аксесуарів.

– Застосування засобів індивідуального захисту: Врахуйте можливість використання антиблискових екранів, захисних окулярів або інших засобів індивідуального захисту для зменшення впливу шкідливих факторів.

– Свідоме використання часу поза робочим місцем: Забезпечте собі достатньо часу для фізичної активності, відпочинку і віддалення від екранів поза робочим часом. Це допоможе знизити загальний вплив тривалої роботи з екраном на організм.

– Проведення фізичних вправ: Після тривалої роботи за комп'ютером важливо робити паузи і проводити фізичні вправи для розслаблення м'язів і покращення кровообігу.

– Регулярні огляди зі спеціалістом: Проходження регулярних медичних оглядів та консультацій з офтальмологом може допомогти виявити можливі проблеми зі здоров'ям, пов'язані з роботою за екранними пристроями.

– Коректура робочого часу: Важливо розподіляти робочий час з екранними пристроями раціонально, з урахуванням перерв і відпочинку, щоб уникнути перенавантаження та втоми.

Норми мікроклімату зазначені у санітарних нормах мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [15].

Для аналізу шкідливих та небезпечних факторів під час роботи з екранними пристроями, слід провести оцінку ризику на робочих місцях, що включає:

– Визначення тривалості роботи з екраном та частоти перерв.

– Забезпечення оптимального освітлення: Важливо мати належне освітлення на робочому місці, щоб уникнути зайвого напруження очей. Краще використовувати природне освітлення або штучне освітлення з регульованою яскравістю.

– Перевірка належності робочого місця до ергономічних стандартів, включаючи правильну позицію тіла, регулювання стільця та стола, використання ергономічних пристроїв.

– Оцінка впливу електромагнітного випромінювання та вжиття заходів для зниження можливих ризиків.

– Забезпечення навчання працівників з правил безпеки та користування екранними пристроями.

– Забезпечення належної вентиляції приміщення: Наявність достатньої свіжого повітря у робочій зоні є важливим для підтримки комфортних умов та запобігання накопиченню шкідливих речовин.

– Встановлення акустичного комфорту: Звуковий комфорт є важливим аспектом безпеки та благополуччя працівників. Необхідно використовувати засоби для зменшення шуму, такі як акустичні екрани або навушники з шумозаглушенням.

– Правильне розташування обладнання: Клавіатура, миша та інші пристрої повинні бути розташовані так, щоб забезпечувати природні та комфортні рухи рук і запобігати напруженням м'язів і суглобів.

– Належне організування кабелів: Кабелі повинні бути правильно організовані, щоб уникнути спотикання, заплутування та інших потенційних небезпек.

Загалом, важливо підтримувати баланс та свідомо ставитись до роботи з екранними пристроями, дотримуючись принципів ергономіки та забезпечуючи безпечні умови праці для збереження здоров'я та працездатності. Врахування цих вимог допоможе зменшити вплив небезпечних та шкідливих факторів на

здоров'я працівників, покращити комфорт роботи та забезпечити безпечні умови праці.

4.2 Методи нейтралізації або зменшення впливу негативних факторів під час роботи з екранними пристроями

Нейтралізація шкідливих та небезпечних факторів під час роботи з екранними пристроями може бути досягнута за допомогою наступних заходів:

– Використання фільтрів на екрани: Фільтри на екрани можуть допомогти зменшити випромінювання і відблискі, що негативно впливають на очі. Вони зменшують поглинання шкідливих променів і допомагають знизити напруження очей.

– Регулювання яскравості та контрастності екрану: Налаштування яскравості та контрастності екрану на комфортний рівень може знизити напруження очей. Потрібно налаштовувати екран таким чином, щоб було зручно читати текст і розрізняти зображення.

– Використання правильного розташування екрану: Екран повинен бути розташований на відстані близько 50-70 см від очей, з огляду на комфортну зону зору. Правильне розташування екрану допомагає знизити напруження очей і шиї.

– Регулярні перерви та вправи для очей: Важливо робити перерви під час тривалої роботи з екраном. Під час перерв можна виконувати спеціальні вправи для очей, які допомагають зменшити напруження та втому очей.

– Користування антибліковими окулярами: Антиблікові окуляри з спеціальними покриттями можуть допомогти зменшити відблисків і поглинання шкідливого світла, що негативно впливає на очі.

– Організація робочого простору: Належна організація робочого місця може знизити вплив негативних факторів. Наприклад, забезпечення належного

освітлення, правильного розташування меблів і пристроїв, використання зручного стільця та клавіатури.

– Здоровий спосіб життя: Загальний стан здоров'я також впливає на толерантність до роботи з екранами. Важливо вести здоровий спосіб життя, забезпечувати достатню фізичну активність, правильне харчування та регулярний сон.

Ці методи можуть допомогти зменшити негативний вплив роботи з екранними пристроями на організм та забезпечити більш комфортні умови праці.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно вирішено мету інженерії програмного забезпечення, а саме розроблено програмне забезпечення для надання розкладу занять студентам НУК. В ході виконання роботи були вирішені наступні завдання:

- Проведено аналіз предметної області.
- Проведено аналіз існуючих аналогів.
- Було сформульовано мету та постановку задачі на розробку.
- Були побудовані моделі варіантів використання та їх специфікації.
- Була розроблена архітектура програмного забезпечення.
- Було створено технічний проект програмного забезпечення.
- Було розроблено програмне забезпечення, а саме Telegram-бот для надання розкладу занять та веб-сервіс для керування розкладом.
- Було проведено тестування програмного забезпечення.

У процесі розробки були використані різні інструменти, включаючи TypeScript, Flutter, Dart, Telegraf, Firebase, Cronjob. Ці інструменти допомогли забезпечити функціональність, зручний доступ до розкладу занять для користувачів через Telegram-бот та його керування через веб-сервіс для адміністраторів та модераторів. Програмне забезпечення «BingNUOS» було успішно впроваджено для надання розкладу занять студентам НУК. У процесі впровадження, Telegram-бот та веб-сервіс були розгорнуті на хмарної платформі Firebase, щоб забезпечити доступ користувачів до сервісу через мережу Інтернет. Після успішного впровадження програмного забезпечення, студенти та викладачі НУК можуть активно користуватися Telegram-ботом та веб-сервісом для перегляду розкладу занять, отримання персоналізованих повідомлень та зручного керування своїм навчальним графіком.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bots: An introduction for developers URL: https://core.telegram.org/bots (дата звернення: 02.05.2023).
2. TypeScript is JavaScript with syntax for types. URL: https://www.typescriptlang.org/ (дата звернення: 02.05.2023).
3. Free cronjobs - from minutely to once a year: https://cron-job.org/en/ (дата звернення: 02.05.2023).
4. Firebase URL: https://firebase.google.com/ (дата звернення: 02.05.2023).
5. Flutter. Build apps for any screen URL: https://flutter.dev/ (дата звернення: 02.05.2023).
6. Cloud Functions for Firebase URL: https://firebase.google.com/docs/functions (дата звернення: 02.05.2023).
7. Node.js URL: https://nodejs.org/ (дата звернення: 02.05.2023).
8. Telegraf – npm URL: https://www.npmjs.com/package/telegraf (дата звернення: 02.03.2023).
9. Django: The web framework for perfectionists with deadlines URL: https://www.djangoproject.com/ (дата звернення: 02.05.2023).
10. Material Design URL: https://m3.material.io/ (дата звернення: 02.05.2023).
11. Dart Programming language | Dart URL: https://dart.dev/ (дата звернення: 02.05.2023).
12. My Study Life - School Planner. URL: https://play.google.com/store/apps/details?id=com.virblue.mystudylife (дата звернення: 02.05.2023).
13. SchedulePro Companion App URL: https://play.google.com/store/apps/details?id=com.shiftboard.schedulepro (дата звернення: 02.05.2023).

14. Про затвердження Вимог щодо безпеки та захисту здоров’я працівників: Закон України від 14.02.2018 № 207 Дата оновлення: 17.06.2015. URL: https://zakon.rada.gov.ua/laws/show/z0508-18 (дата звернення: 02.05.2023).

15. Санітарні норми мікроклімату виробничих приміщень (ДСН 3.3.6.042-99): https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення: 02.05.2023).

Додаток А – Технічне завдання на розробку програмного забезпечення для надання розкладу занять студентам НУК

Вступ

Повна назва програмного забезпечення: веб-сервіс та Telegram бот для надання розкладу занять студентам НУК «BingNUOS»

Сфера застосування: навчальні заклади.

1. Підстави для розробки

Підставою для розробки даного програмного забезпечення є наказ на затвердження тем кваліфікаційних робіт бакалаврів зі спеціальності 121 «Інженерія програмного забезпечення» в Національному університеті кораблебудування ім. адмірала Макарова.

2. Призначення розробки

2.1 Експлуатаційне призначення

Веб-сервіс має експлуатуватися на персональній ЕОМ користувача через браузер. Telegram бот має експлуатуватися у мобільному додатку Telegram

2.2 Функціональне призначення

Для користувача Telegram бота програма надає наступні можливості: перегляд та вибір груп, детальний перегляд розкладу із назвою предмета, кабінету та викладача на кожен будній день неділі та підписка або скасувати підписку на повідомлення о майбутніх парах

Для модератора веб-сервісу програма дозволяє додавати/редагувати/видаляти розклад із дозволених груп модерації.

Для адміністратора веб-сервісу програма дозволяє додавати/редагувати/видаляти нові групи та розклад. Також передбачається можливість додавати модераторів та редагувати дозволені групи для модерації.

3. Вимоги до програми або програмного виробу

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Функції користувача

– Користувач повинен мати можливість взаємодіяти з Telegram-ботом для вибору групи, підписки на сповіщення про майбутні заняття та перегляду розкладу для обраної групи на кожен робочий день.

– Користувач повинен отримувати сповіщення за 10 хвилин до початку кожного заняття.

– Користувач не повинен мати доступу до веб-панелі адміністратора.

Функції модератора

– Модератор повинен мати можливість авторизуватися в веб-сервісі планування за допомогою електронної пошти та пароля.

– Модератор повинен мати можливість переглядати розклад для дозволених груп модерації на кожен робочий день.

– Модератор повинен мати можливість додавати, редагувати та видаляти розклад для дозволених груп модерації на кожен робочий день.

– Модератор не повинен мати можливості додавати, редагувати або видаляти групи, реєструвати нових модераторів.

Функції адміністратора

– Адміністратор повинен мати можливість авторизуватися в веб-сервісі планування за допомогою електронної пошти та пароля.

– Адміністратор повинен мати можливість реєструвати нових модераторів і призначати їх до груп модерації.

– Адміністратор повинен мати можливість переглядати розклад усіх груп на кожен робочий день.

– Адміністратор повинен мати можливість створювати, редагувати та видаляти групи та розклади для кожного робочого дня.

3.1.2 Вимоги до організації вхідних та вихідних даних

Дані про розклад та користувачів зберігаються у базі даних. СУБД забезпечує розмежування прав доступу до даних – надає користувачу права на читання, а модератору та адміністратору – на читання та запис

Вхідні дані:

– Користувач повинен вказати номер своєї групи натиснувши відповідну кнопку із номером групи у Telegram-боті.

– Користувач повинен підписатися або відписатися від сповіщень використовуючи відповідну команду у Telegram-боті (/subscribe або /unsubscribe)

– Модератор та Адміністратор повинен вказати свою електронну пошту та пароль у відповідні поля для авторизації у веб-сервісі планування.

– Модератор та Адміністратор повинні вказати номер групи та деталі розкладу у відповідні поля для додавання або редагування розкладу у веб-сервісі планування.

– Адміністратор повинен вказати ім’я електронну пошту модератора та призначити його/її до груп модерації у відповідні поля для реєстрації модератора у веб-сервісі планування.

Вихідні дані:

– Телеграм-бот повинен показувати розклад для обраної групи на кожен будній день.

– Telegram-бот повинен надсилати сповіщення за 10 хвилин до початку кожного заняття користувачам, на яких він підписаний.

– У веб-сервісі планування для модератора повинен відображатися розклад для дозволенних груп модерації на кожен робочий день.

– У веб-сервісі планування для адміністратора повинен відображатися розклад для усіх груп на кожен робочий день.

– У веб-сервісі планування адміністратора повинен відображатися список модераторів та призначених їм груп модерації.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійного (сталого) функціонування програми

Надійне (стійке) функціонування програми має бути забезпечене виконанням сукупності організаційно-технічних заходів, перелік яких подано нижче:

– захист від некоректних дій користувача(під час розпізнавання);

– встановлення Flutter Framework не нижче версії 2.18.6.

3.2.2 Час відновлення після відмови

Вимоги до часу відновлення після відмови не пред’являються.

3.2.3 Відмови через некоректні дії оператора

Відмова можлива при наступних умовах:

– внаслідок закриття програми під час розпізнавання, в цьому випадку користувачеві показується додаткове вікно, чи впевнений він що хоче закрити програму;

– якщо розмір файлу перевищує розмір оперативної пам’яті, то з’являється вікно про недостачу оперативної пам’яті для розпізнавання файлу.

3.3 Умови експлуатації

3.3.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні відповідати вимогам, що пред'являються до технічних засобів в частині умов їх експлуатації.

3.3.2 Вимоги до чисельності та кваліфікації користувачів

Користувач повинен мати базові навички роботи з комп’ютером.

3.4 Вимоги до складу і параметрів технічних засобів

Для роботи із програмним забезпеченням необхідний наступний склад технічних засобів з мінімальними параметрами:

– Процесор Intel Pentium 1Ghz.

– Оперативна пам'ять 512Мб.

– Дискового простору 1 Гб.

– Роздільна здатність екрану 800x600 пікселів

– Інтернет-з’єднання. Програмне забезпечення вимагає доступу до Інтернету.

3.5 Вимоги до інформаційної та програмної сумісності

3.5.1 Вимоги до вихідних кодів і мов програмування

Вихідні коди програми повинні бути реалізовані на мовах Dart та TypeScript. В якості інтегрованого середовища розробки програми буде використовуватися середовище Microsoft Visual Studio Code. Використовувати Flutter Framework версії не нижче 2.18.6.

Системні програмні засоби, які використовуються програмою, повинні бути представлені ліцензійною версією операційної системи Windows (версії не нижче Windows 7).

3.6 Спеціальні вимоги

Програма повинна забезпечувати взаємодію з користувачем за допомогою графічного інтерфейсу користувача, розробленого відповідно до рекомендацій компанії-виробника операційної системи.

4. Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

– технічне завдання;
– програму та методику випробувань;
– керівництво користувача;
– опис програми;
– код програми.

5. Техніко-економічні показники

У даній кваліфікаційній роботі техніко-економічні показники не розраховуються.

6. Етапи роботи

Етапи роботи представлені у таблиці А.1

Таблиця А.1 – Етапи роботи

Номер	Назва етапів роботи створення програмного забезпечення	Терміни виконання
1.	Аналіз вимог до ПЗ	01.03.2023
2.	Розробка архітектури ПЗ	21.03.2023
3.	Розробка проекту ПЗ	16.04.2023
4.	Реалізація та тестування ПЗ	26.05.2023
5.	Випробування ПЗ	29.05.2023

7. Порядок контролю та приймання

Контроль за аналізом та проектуванням кожної окремої частини програмного забезпечення на кожному етапі з урахуванням вимог, визначених у технічному завдані. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Прийом проводяться відповідно до програми і методики випробувань і документують за допомогою протоколу проведення випробувань. У разі знаходження помилок підчас прийому програмного виробу складається акт про знайдені помилки, який підписуються представниками замовника і розробника і затверджуються керівником організації – замовника та організації – розробника. Розробник повинен на протязі не більше ніж 2 тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

Додаток Б – Вихідні тексти програмних модулів

index.ts – Головний файл функцій Firebase:

```
// Імпорт необхідних модулів
import * as dotenv from "dotenv";
import * as admin from "firebase-admin";
import * as functions from "firebase-functions";
import { TelegramBot } from "./telegram/telegram_bot";
import { Cronjob } from "./cronjob/cronjob";
import { AdminApp } from "./firebase/admin_app";
import { FlutterFunctions } from "./firebase/flutter_functions";

// Завантаження конфігурації з файлу .env
dotenv.config();

// Ініціалізація платформи Firebase
admin.initializeApp(functions.config().firebase);

// Ініціалізація та експорт TelegramBot для обробки запитів
export const telegramBot = new TelegramBot().getHandler();

// Ініціалізація та експорт Cronjob для виконання планових завдань
export const cronjob = new Cronjob().getHandler();

// Створення обробників подій для додавання, видалення та оновлення розкладу груп
export const onCreateSchedulesGroup = new AdminApp().onCreateSchedulesGroup;
export const onDeleteSchedulesGroup = new AdminApp().onDeleteSchedulesGroup;
export const onUpdateSchedulesGroup = new AdminApp().onUpdateSchedulesGroup;

// Створення обробників подій для реєстрації та видалення користувачів
export const registerUser = new FlutterFunctions().registerUser;
export const removeUser = new FlutterFunctions().removeUser;
```

Цей код визначає основні модулі програмного забезпечення та експортує їх для використання в системі. Включається необхідна конфігурація, ініціалізується Firebase та створюються обробники подій для взаємодії з Telegram-ботом.

telegram_bot.ts – реалізація Telegram-бота для взаємодії з користувачами:

```
import * as functions from "firebase-functions";
import {Context, Telegraf} from "telegraf";

import {Commands} from "./commands";
import {Queries} from "./queries";

import {RealtimeDatabase} from "../firebase/realtime_database";
import {FirestoreDatabase} from "../firebase/firestore_database";

export interface CustomContext extends Context {
  db: RealtimeDatabase;
  firestore: FirestoreDatabase;
}

export class TelegramBot {
  private bot: Telegraf<CustomContext>;

  constructor() {
    const botToken = process.env.BOT_TOKEN;
    if (!botToken) {
      console.error("Missing BOT_TOKEN environment variable");
      process.exit(1);
    }
    this.bot = new Telegraf(botToken);
    this.bot.use(async (ctx, next) => {
      const customContext = ctx as CustomContext;
      customContext.db = new RealtimeDatabase();
      customContext.firestore = new FirestoreDatabase();
      await next();
    });
    this.init();
  }

  private catch(err: unknown, ctx: Context) {
    functions.logger.error(`Telegram bot error: ${err}`);
    ctx.reply("Something went wrong");
  }

  private init() {
    functions.logger.info("Initializing bot");
    this.bot.catch(this.catch);
```

```
    new Commands(this.bot);
    new Queries(this.bot);
  }

  public getHandler(): functions.HttpsFunction {
    return functions
      .region("europe-west1")
      .https.onRequest((req, res) => {
        this.bot.handleUpdate(req.body, res);
      });
  }
}
```

– TelegramBot - клас, що представляє Telegram-бота. Він ініціалізується з використанням токену бота, який зберігається в змінній середовища BOT_TOKEN. При відсутності цієї змінної, видається помилка. Клас також створює екземпляри об'єктів RealtimeDatabase та FirestoreDatabase для забезпечення доступу до відповідних баз даних.

– catch - функція для обробки помилок, які виникають під час роботи бота. Вона логує помилку та надсилає користувачу повідомлення про помилку.

– init - функція для ініціалізації бота. Вона логує початок ініціалізації, встановлює обробники команд та запитів (Commands та Queries) для взаємодії з користувачами.

– getHandler - функція, яка повертає обробник бота для використання в Firebase Functions. Вона обробляє HTTP-запити, які надходять на вебхук бота і передає їх обробнику бота для подальшої обробки.

cronjob.ts – Функція Firebase яка виконує регулярні завдання для відправки розкладу занять користувачам у вказаний час:

```
import * as functions from "firebase-functions";
import {Telegraf} from "telegraf";
import {lessonTimes, times} from "../telegram/constants";
import {RealtimeDatabase} from "../firebase/realtime_database";
import {Services} from "../telegram/services";

export class Cronjob {
```

```
 private bot: Telegraf;
 constructor() {
  const botToken = process.env.BOT_TOKEN;
  if (!botToken) {
   console.error("Missing BOT_TOKEN environment variable");
   process.exit(1);
  }
  this.bot = new Telegraf(botToken);
 }
 private async sendMessage(uid: string, message: string): Promise<void> {
  await this.bot.telegram.sendMessage(uid, message);
 }
 private async sendSchedules(): Promise<void> {
  const date = new Date();
  const weekday = date.toLocaleString("en", {
   timeZone: "Europe/Kiev",
   weekday: "long",
  }).toLowerCase();
  if (weekday === "saturday" || weekday === "sunday") return;

  const time = date.toLocaleString("uk", {
   timeZone: "Europe/Kiev",
   hour: "numeric",
   minute: "numeric",
  });
  functions.logger.log(`Time: ${time}`);
  const number = times.findIndex((e) => e === time);
  if (number === -1) return;

  functions.logger.log(
   `${times[number]} - will send schedule $weekday on ${number + 1}
lesson`,
  );

  const subscribedUsers = await new
RealtimeDatabase().getSubscribedUsers();

  for (const [group, userIds] of subscribedUsers) {
   let message =
       await Services.getScheduleByPairNumber(group, weekday, number +
1);
   if (message === undefined) continue;
   message += `\n\nПочаток ${number + 1} пари о
${lessonTimes[number]}`;
```

```
      for (const uid of userIds) {
        functions.logger.log(message);
        await this.sendMessage(uid, message);
      }
    }
  }

  public getHandler(): functions.HttpsFunction {
    return functions
      .region("europe-west1")
      .https.onRequest(async (req, res): Promise<void> => {
        functions.logger.info("Cronjob started");
        await this.sendSchedules();
        res.sendStatus(200);
      });
  }
}
```

– Клас Cronjob відповідає за виконання регулярних завдань, використовуючи реалізацію бібліотеки telegraf для взаємодії з Telegram API.

– У конструкторі створюється екземпляр класу Telegraf з використанням токена бота, який зчитується з оточення.

– Метод sendMessage відправляє повідомлення до користувача з використанням Telegram API.

– Метод sendSchedules перевіряє поточний день тижня та час і надсилає розклад занять підписаним користувачам. Він використовує об'єкт RealtimeDatabase для отримання списку підписаних користувачів та об'єкт Services для отримання розкладу занять.

– Метод getHandler повертає обробник для використання в Firebase Functions. Він визначає HTTP-функцію, яка буде виконуватися при спрацюванні таймера. Усередині цієї функції виконується метод sendSchedules, а потім надсилається статус 200.

main.dart – вихідний головний файл Flutter веб-сервісу планування який організовує структуру та логіку програмного забезпечення, встановлює тему додатку, маршрутизацію та локалізацію:

```
import 'package:bingnuos_admin_panel/constants.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/gestures.dart';
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:flutter_localizations/flutter_localizations.dart';
import 'package:flutter_web_plugins/url_strategy.dart' show usePathUrlStrategy;
import 'package:hive_flutter/hive_flutter.dart';
import 'package:provider/provider.dart';

import 'package:bingnuos_admin_panel/providers/app_theme_provider.dart';
import 'package:bingnuos_admin_panel/providers/search_groups_provider.dart';
import 'package:bingnuos_admin_panel/providers/weekday_provider.dart';
import 'package:bingnuos_admin_panel/services/app_router.dart';
import 'package:bingnuos_admin_panel/services/firebase/auth_service.dart';
import 'package:bingnuos_admin_panel/services/hive_service.dart';
import 'package:bingnuos_admin_panel/utils/app_locale.dart';
import 'package:bingnuos_admin_panel/utils/stream_extensions.dart';

import 'firebase_options.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  usePathUrlStrategy();

  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );

  await HiveService.initialize();

  runApp(
    MultiProvider(
      providers: [
        ChangeNotifierProvider(
          create: (_) => AppThemeProvider(),
```

```
        ),
        ChangeNotifierProvider(
          create: (_) => WeekdayProvider(),
        ),
        Provider(
          create: (_) => AuthService(),
        ),
        Provider(
          create: (_) => HiveService(),
        ),
        ChangeNotifierProvider(
          create: (_) => GroupSearchProvider(),
        ),
        ChangeNotifierProvider(
          create: (_) => ManageUserGroupSearchbarProvider(),
        ),
      ],
      child: const MyApp(),
    ),
  );
}

class MyApp extends StatefulWidget {
  const MyApp({super.key});

  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  late ValueListenable<User?> userChanges;

  @override
  void initState() {
    final authService = context.read<AuthService>();
    authService.initialize();
    userChanges =
        authService.authStateChanges.toValueNotifier(authService.user);
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    final router = AppRouter(userChanges).router;
```

```
      return Consumer<AppThemeProvider>(
        builder: (context, themeProvider, _) {
          return ValueListenableBuilder<Box<String>>(
            valueListenable:
                Provider.of<HiveService>(context).languageBoxListenable,
            builder: (context, box, child) {
              final locale = box.get(LANGUAGE_BOX_KEY) ?? 'en';
              return MaterialApp.router(
                locale: Locale.fromSubtags(languageCode: locale),
                theme: themeProvider.selectedTheme,
                routeInformationProvider: router.routeInformationProvider,
                routeInformationParser: router.routeInformationParser,
                routerDelegate: router.routerDelegate,
                debugShowCheckedModeBanner: false,
                title: AppLocale(context).appName,
                localizationsDelegates: const [
                  AppLocalizations.delegate,
                  GlobalMaterialLocalizations.delegate,
                  GlobalWidgetsLocalizations.delegate,
                  GlobalCupertinoLocalizations.delegate,
                ],
                supportedLocales: const [
                  Locale('en', ''),
                  Locale('uk', ''),
                  Locale('ru', ''),
                ],
                scrollBehavior: AppScrollBehavior(),
              );
            },
          );
        },
      );
    }
```

– Виклик функції main є точкою входу в застосунок.

– У функції main відбувається ініціалізація Flutter, налаштування Firebase та локальної бази даних Hive, створення провайдерів та запуск додатку за допомогою runApp.

– Клас MyApp є кореневим віджетом додатку та є становим віджетом. Він містить логіку ініціалізації та рендерингу додатку.

– У build методі MyApp використовуються провайдери для отримання даних та віджетів. Залежно від стану, відображається відповідний вміст.

login_view.dart – Файл реалізації подання для входу в систему. Він включає різні залежності та компоненти, такі як текстові поля, кнопки та віджети для автентифікації.

```
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';
import 'package:provider/provider.dart';
import 'package:bingnuos_admin_panel/constants.dart';
import 'package:bingnuos_admin_panel/services/firebase/auth_service.dart';
import 'package:bingnuos_admin_panel/services/snackbar_service.dart';
import 'package:bingnuos_admin_panel/ui/components/app_text_field.dart';
import
'package:bingnuos_admin_panel/ui/components/bing_nuos_auth/bing_nuos_au
th_widget.dart';
import
'package:bingnuos_admin_panel/ui/components/bing_nuos_auth/forgot_passw
ord_login_widget.dart';
import
'package:bingnuos_admin_panel/ui/components/buttons/app_elevated_button.d
art';
import 'package:bingnuos_admin_panel/utils/app_locale.dart';
import 'package:bingnuos_admin_panel/utils/utils.dart';

class LoginView extends StatefulWidget {
  const LoginView({super.key});
  @override
  State<LoginView> createState() => _LoginViewState();
}
class _LoginViewState extends State<LoginView> {
  final emailTextFieldController =
      ValueNotifier<TextEditingController>(TextEditingController());
  final passwordTextFieldController =
      ValueNotifier<TextEditingController>(TextEditingController());
  final isEmailError = ValueNotifier<bool>(false);
  final isPasswordError = ValueNotifier<bool>(false);
  final isLoading = ValueNotifier<bool>(false);

  void _resetPassword() {
    context.go(resetPasswordLoc);
  }
```

```
void _login() async {
  isEmailError.value = false;
  isPasswordError.value = false;
  isLoading.value = true;

  String email = emailTextFieldController.value.text.trim().toLowerCase();
  String password = passwordTextFieldController.value.text;

  if (_validate(email, password)) {
    Map<bool, String> result =
        await context.read<AuthService>().signInWithEmailAndPassword(
              email: email,
              password: password,
            );
    bool success = result.keys.first;
    String e = result.values.first;
    if (!success && mounted) {
      SnackBarService(context).show(context
          .read<AuthService>()
          .getFirebaseAuthErrorMessage(e, context));
    }
  }
  isLoading.value = false;
}
bool _validate(String email, String password) {
  if (!Utils.emailValid(email)) {
    isEmailError.value = true;
    return false;
  }
  if (!Utils.passwordValid(password)) {
    isPasswordError.value = true;
    return false;
  }
  return true;
}
@override
Widget build(BuildContext context) {
  return Scaffold(
    resizeToAvoidBottomInset: false,
    body: SafeArea(
      child: BingNuosAuthWidget(
        children: [
          Align(
            alignment: Alignment.center,
```

```
  child: Text(
    AppLocale(context).login,
    style: Theme.of(context).textTheme.displayMedium,
  ),
),
const SizedBox(height: 16),
ValueListenableBuilder<bool>(
  valueListenable: isEmailError,
  builder: (context, showError, child) => AppTextField(
    key: const Key("email"),
    labelText: AppLocale(context).email,
    controller: emailTextFieldController.value,
    errorText: AppLocale(context).emailWrong,
    showError: showError,
  ),
),
ValueListenableBuilder<bool>(
  valueListenable: isPasswordError,
  builder: (context, showError, child) => AppTextField(
    key: const Key("password"),
    labelText: AppLocale(context).password,
    controller: passwordTextFieldController.value,
    inputType: InputType.password,
    errorText: AppLocale(context).passwordWrong,
    showError: showError,
  ),
),
ForgotPasswordLoginWidget(
  text: AppLocale(context).forgotPassword,
  buttonText: AppLocale(context).resetHere,
  onTap: _resetPassword,
),
Container(
  margin: const EdgeInsets.only(top: 20),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.end,
    children: [
      ValueListenableBuilder<bool>(
        valueListenable: isLoading,
        builder: (context, isDisabled, child) {
          return AppElevatedButton(
            title: AppLocale(context).login,
            width: 120,
            height: 40,
```

```
                    isDisabled: isDisabled,
                    onPressed: _login,
                  );
                },
              ),
            ],
          ),
        ),
        ValueListenableBuilder(
          valueListenable: isLoading,
          builder: (_, value, __) => value
              ? const CircularProgressIndicator()
              : const SizedBox.shrink(),
        ),
      ],
    ),
  ),
 );
 }
}
```

Додаток В – Опис програмного забезпечення

1. Загальні відомості

1.1 Позначення і найменування програми

Назва програми: «BingNUOS». Програма розроблена для надання розкладу занять студентам НУК. Вона надає студентам, викладачам і адміністраторам зручний доступ до поточного розкладу та персоналізованих повідомлень. Завдяки Telegram боту, користувачі можуть переглядати та вибирати групу, докладно переглядати розклад з назвою предмету, аудиторією та викладачем для кожного робочого дня, а також підписуватися або відписуватися від отримання повідомлень про наближення занять. Модератори веб-сервісу мають можливість додавати, редагувати та видаляти розклад для дозволених груп, а адміністратори можуть створювати, редагувати та видаляти групи та розклад для всіх груп. Програма забезпечує безпеку даних та зручний інтерфейс для всіх користувачів.

1.2 Програмне забезпечення, необхідне для функціонування програми

Для роботи із програмним забезпеченням необхідний наступний склад технічних засобів з мінімальними параметрами:

- Процесор Intel Pentium 1Ghz.
- Оперативна пам'ять 512Мб.
- Дискового простору 1 Гб.
- Роздільна здатність екрану 800x600 пікселів
- Інтернет-з’єднання. Програмне забезпечення вимагає доступу до Інтернету.

Системні програмні засоби, які використовуються програмою, повинні бути представлені ліцензійною версією операційної системи Windows (версії не нижче Windows 7). Веб-сервіс має експлуатуватися на персональній ЕОМ користувача через браузер. Telegram бот має експлуатуватися у мобільному додатку Telegram.

1.3 Мови програмування

Мови програмування, використані в розробці програми "BingNUOS", включають:

– TypeScript: Використовується для розробки Telegram бота, який працює як Firebase функція. TypeScript є типізованою мовою програмування, що забезпечує переваги, такі як перевірка типів, поліпшена інтеграція з інструментами розробки і полегшена читабельність коду.

– Node.js: Використовується як середовище виконання для Firebase функцій, які працюють у хмарі. Node.js дозволяє виконувати JavaScript на серверному боці, що робить його ідеальним вибором для розробки серверної частини програми.

– Фреймворк Flutter, мова програмування Dart: Використовується для розробки веб-сервісу для керування розкладом. Flutter є потужним фреймворком для створення переносних, високопродуктивних додатків, які працюють як на веб-платформі, так і на мобільних пристроях. Використання Flutter дозволяє досягти швидкої розробки, багаторазовості коду і багатофункціонального інтерфейсу користувача.

2. Функціональне призначення

Для користувача Telegram-бота, програма повинна виконувати наступні функції: перегляд та вибір груп, детальний перегляд розкладу із назвою предмета, кабінету та викладача на кожен будній день неділі та підписка або скасувати підписку на повідомлення о майбутніх парах.

Для модератора веб-сервісу, програма повинна дозволяти додавати, редагувати та видаляти розклад із дозволених груп для модерації.

Для адміністратора веб-сервісу програма повинна дозволяти додавати, редагувати, видаляти нові групи та розклад. Також повинна передбачатися можливість додавати модераторів та редагувати дозволені групи для модерації.

3. Опис логічної структури

Програма "BingNUOS" має наступну логічну структуру:

Telegram-бот:

– Модуль обробки повідомлень: Відповідає за приймання та обробку повідомлень від користувачів Telegram. Цей модуль дозволяє користувачам переглядати розклади, вибирати групи, підписуватися на повідомлення про майбутні пари та керувати підпискою.

– Модуль взаємодії з базою даних: Відповідає за звернення до бази даних для отримання необхідної інформації про розклади та користувачів, збереження оновлень щодо підписок та повідомлень.

Веб-сервіс:

– Модуль авторизації та аутентифікації: Дозволяє модераторам та адміністраторам авторизуватися в системі за допомогою свого електронного листа та пароля.

– Модуль керування розкладом для модераторів: Надає модераторам можливість переглядати, додавати, редагувати та видаляти розклади для груп, на які їм надано право модерації.

– Модуль керування групами та розкладами для адміністраторів: Дозволяє адміністраторам переглядати, додавати, редагувати та видаляти групи та розклади.

– Модуль керування модераторів для адміністраторів: Дозволяє адміністраторам додавати, редагувати та видаляти модераторів та їхні дозволені групи для модерації.

База даних:

– Таблиця "Групи": Зберігає інформацію про групи студентів, включаючи їхні номери та інші характеристики.

– Таблиця "Розклади": Містить дані про розклади занять для різних груп, включаючи день тижня, час, предмет, кабінет та викладача.

– Таблиця "Користувачі": Зберігає дані про користувачів, зокрема їхні Telegram ідентифікатори та підписки на повідомлення про майбутні пари.

Ця логічна структура дозволяє програмі «BingNUOS» ефективно керувати розкладами, підписками та доступом до функцій в залежності від ролі користувача (користувач Telegram, модератор або адміністратор).

3.2 Алгоритм програми

Алгоритм роботи Telegram бота:

– Користувач відкриває діалог з Telegram ботом "BingNUOS".

– Бот привітає користувача і запропонує вибрати курс та групу зі списку доступних.

– Користувач вибирає групу, і бот відображає список основних команд.

– Користувач має можливість отримати розклад для цієї групи на кожен будній день.

– Користувач має можливість підписатися або скасувати підписку на повідомлення про майбутні пари.

– За 10 хвилин до початку кожного заняття, бот автоматично надсилає повідомлення з інформацією про пару користувачу, який підписався на повідомлення.

Алгоритм роботи веб-сервісу:

– Модератор або адміністратор відкриває веб-сервіс «BingNUOS» у своєму браузері та авторизується з використанням електронної пошти та пароля.

– Після успішної авторизації, модератор має доступ до функцій модерації, а адміністратор - до функцій адміністрування.

– Модератор може переглядати розклади для груп, на які він має право модерації, додавати, редагувати та видаляти розклади для цих груп.

– Адміністратор може переглядати розклади для всіх груп, додавати, редагувати та видаляти групи та розклади. Він також може додавати та редагувати модераторів та їхні дозволені групи для модерації.

– Зміни в розкладах, групах та модераторах зберігаються в базі даних для подальшого використання.

3.3 Використовувані методи

Програмне забезпечення «BingNUOS» розроблено мовами програмування TypeScript та Dart, які є об’єктно-орієнтованими.

– Метод завантаження розкладу: Реалізація функціональності для завантаження розкладу з бази даних. Цей метод включає отримання даних про групи, предмети, викладачів, кабінети та розклад у відповідному форматі і збереження їх для подальшого використання.

– Метод обробки даних: Реалізація функцій для обробки отриманих даних розкладу. Цей метод включає сортування розкладу за днями тижня та пошук груп за назвою.

– Метод взаємодії з базою даних: Реалізація функцій для зчитування, запису та оновлення даних в базі даних. Цей метод забезпечує зв'язок програми з базою даних, дозволяє зчитувати розклад, додавати нові групи та розклад, оновлювати існуючі записи та реєструвати модераторів.

– Метод відображення інтерфейсу користувача: Реалізація функціональності для відображення розкладу та інших даних в інтерфейсі користувача. Цей метод включає створення візуальних елементів, таблиць, списків та графічних компонентів для зручного представлення даних користувачу.

– Метод взаємодії з Telegram-ботом: Реалізація функцій для обробки команд користувача, надсилання повідомлень, підписки на розклад та іншої взаємодії з Telegram-ботом. Цей метод використовує бібліотеку telegraf для створення та керування функціональністю Telegram-бота.

4 Технічні засоби

У розробці програмного забезпечення "BingNUOS" були використані наступні технічні засоби:

Telegram-бот:

– Використання бібліотеки telegraf для створення Telegram-бота з функціональністю обробки команд користувача, надсилання повідомлень та взаємодії з базою даних Firebase Realtime Database.

– Використання мови програмування TypeScript для написання функціональності бота.

– Використання Firebase Functions для розгортання Telegram-бота як функції Firebase.

Розробка веб-сервісу:

– Використання фреймворку Flutter для створення веб-сервісу планування.

– Використання мови програмування Dart для розробки компонентів інтерфейсу користувача, обробки взаємодії з користувачем та зв'язку з базою даних Firebase Firestore.

– Використання Firebase Hosting для розгортання веб-сервісу планування.

– Використання Firebase Authentication для автентифікації та авторизації користувачів веб-сервісу.

Використання інших засобів:

– Використання Visual Studio Code як середи розробки для зручного написання та редагування коду.

– Використання Firebase CLI для налаштування та управління проектом Firebase.

– Використання сервісу cron-job.org для виклику певної Firebase функції для надсилання розкладу у заданий час.

5. Виклик та завантаження

Виклик та завантаження програмного забезпечення «BingNUOS» відбувається запуском Telegram-бота у платформі Telegram. Виклик та завантаження веб-сервісу планування «BingNUOS» відбувається шляхом доступу до його URL-адреси на Firebase Hosting. Після розгортання

програмного коду на Firebase Hosting, веб-сервіс стає доступним за зазначеною URL-адресою.

6. Вхідні дані

Дані можна вводити в систему за допомогою різних пристроїв введення, таких як клавіатура та миша. Введення даних може здійснюватися шляхом ручного введення або вибору зі списків, що надаються.

7. Вихідні дані

Дані можна виводити з системи за допомогою пристроїв виведення, таких як монітор. Це дозволяє користувачу переглядати інформацію, яка була оброблена системою.

Додаток Г – Інструкція користувача

За допомогою цієї інструкції ви зможете використовувати основні функції програмного забезпечення «BingNUOS», такі як перегляд розкладу, додавання нових груп та предметів, зміна даних, а також налаштування модераторів для редагування розкладу.

Інструкція користувача для Telegram-бота «BingNUOS»:

1. Реєстрація та авторизація:

– Для доступу до Telegram-боту «BingNUOS» вам необхідно завантажити та встановити мобільний додаток Telegram на свій пристрій.

– Після встановлення Telegram, відкрийте додаток та знайдіть пошукове поле.

– Введіть назву бота «bingNUOS_bot» і знайдіть його в результаті пошуку.

– Виберіть бота «BingNUOS» і натисніть кнопку «Start» для початку використання бота.

2. Вибір групи та перегляд розкладу:

– Після старту бота «BingNUOS» вам буде запропоновано вибрати свій курс та свою групу. Виберіть номер своєї групи за вказівкою у меню.

– Після вибору групи ви зможете переглянути розклад занять для своєї групи на кожний робочий день. Інформація про предмети, аудиторії та викладачів буде відображатися.

3. Підписка на повідомлення:

– Якщо ви бажаєте отримувати повідомлення про наближення занять, ви можете підписатися на них. Виберіть опцію підписки на сповіщення у боті «BingNUOS» або за допомогою команди /subscribe.

– За 10 хвилин до початку кожного заняття ви отримаєте повідомлення з нагадуванням.

4. Відписка від повідомлень: Якщо ви більше не бажаєте отримувати повідомлення про наближення занять, ви можете відписатися від них. Виберіть опцію відписки від сповіщень у боті «BingNUOS» або за допомогою команди /unsubscribe.

Використання веб-сервісу модератора (для модераторів):

– Для доступу до веб-сервісу модератора відкрийте веб-браузер і введіть URL-адресу веб-сервісу https://bingnuos.web.app/.

– Введіть свою електронну пошту та пароль для авторизації.

– Після успішної авторизації ви зможете переглядати розклад для груп, які ви модеруєте, на кожний робочий день. Ви також зможете додавати, редагувати та видаляти розклад для цих груп.

Використання веб-сервісу адміністратора (для адміністраторів):

– Для доступу до веб-сервісу адміністратора відкрийте веб-браузер і введіть URL-адресу веб-сервісу "BingNUOS".

– Введіть свою електронну пошту та пароль для авторизації.

– Після успішної авторизації ви зможете переглядати розклад для всіх груп на кожний робочий день. Ви також зможете створювати, редагувати та видаляти групи і розклади. Також ви зможете переглянути список модераторів та призначені групи.

Користування основними функціями:

– Для переключення між днями тижня використовуйте панель, розташовану у верхній частині додатку. Ви можете обрати потрібний день, щоб переглянути розклад на цей день.

– Щоб додати нову групу або новий предмет до розкладу, натисніть на пусте поле у розкладі. У вікні додавання групи введіть номер групи, а в вікні додавання предмета введіть його назву, номер кабінету та прізвище викладача.

– Якщо ви хочете внести зміни в існуючий предмет або групу, натисніть на відповідне поле у розкладі. Ви зможете змінити дані, пов'язані з предметом або групою, і зберегти зміни.

Налаштування модераторів (для адміністраторів):

– Адміністратор може додавати модераторів, які будуть відповідальні за редагування розкладу для певних груп. Для цього натисніть кнопку меню в правому верхньому куті та виберіть пункт "Керувати користувачами".

– Ви можете переглядати та редагувати кожного користувача у веб-сервісі та за бажанням видалити його.

– Для додавання або редагування користувача натисніть на відповідну кнопку та введіть ім'я та електронну пошту модератора, а також налаштуйте доступні групи для модерації. Модератор зможе увійти до своєї панелі, використовуючи пароль, який буде відправлений на його пошту.

Додаток Д – Програма та методика випробувань програмного забезпечення

1. Об’єкт випробування

Об'єктом випробування є програмне забезпечення «BingNUOS», яке розробляється для студентів НУК з метою надання розкладу занять. Ця розробка проводиться на основі потреб і вимог студентів для полегшення організації їх навчального процесу.

2. Мета випробування

Метою випробування програми «BingNUOS» є перевірка її функціональності, надійності та відповідності вимогам студентів НУК. Випробування спрямоване на виявлення можливих дефектів, помилок та проблем, які можуть вплинути на роботу програми та точність розкладу занять студентів НУК.

Конкретні цілі випробування включають:

– Перевірку коректності вибору групи.

– Перевірку перегляду розкладу для обраної групи.

– Перевірка підписки та відписки від сповіщень.

– Перевірку доставки сповіщень за 10 хвилин до початку кожного заняття.

– Перевірку авторизації модератора та адміністратора в веб-сервісі.

– Перевірка правильності відображення розкладу для дозволених груп для модератора.

– Перевірку додавання, редагування та видалення розкладу модератором та адміністратором.

– Перевірка правильності відображення розкладу усіх груп для адміністратора.

– Перевірку створення, редагування та видалення груп та розкладів адміністратором.

– Перевірку створення, редагування та видалення модераторів та їх груп модерації адміністратором.

3. Вимоги до програмного забезпечення

При проведенні випробувань програми «BingNUOS» функціональні характеристики підлягають перевірці на відповідальність вимогам, викладеним у пункті «Вимоги до функціональних характеристик» технічного завдання.

4. Вимоги до програмної документації

Програмна документація повинна включати в себе наступні документи:

1) Текст програми.

2) Опис програми.

3) Програма та методика випробувань.

4) Інструкція користувача.

5. Засоби і порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

– CPU: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
– RAM: 16 GB 2400MHz
– SSD: Samsung SSD 970 EVO Plus 500GB
– Ethernet: TP-LINK Gigabit Ethernet USB Adapter

5.2 Програмні засоби, що використовуються під час випробувань

– Windows 10 x64 22H2 19045.2913
– Visual Studio Code 1.78.2
– Google Chrome 108.0.5359.215
– Telegram 4.8.3
– Flutter 3.10.2
– Dart SDK 3.0.3
– TypeScript 5.0.4
– Firebase CLI 12.2.1
– Node.js 16.17.1

5.3 Порядок проведення випробувань

Випробування програми «BingNUOS» складається з двох етапів: перевірки вимог до програмної документації та перевірки вимог до функціональних характеристик програмного забезпечення.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

Під час перевірки вимог до програмної документації було перевірено наявність всіх необхідних документів і їх відповідність вимогам до змісту і формату. Далі було оцінено відповідність документів вимогам до змісту, зокрема перевірено наявність необхідних даних і опису програми, а також відповідність програми та методики випробувань вимогам до тестування. Також було перевірено, що інструкція користувача містить зрозумілі кроки щодо використання програмного забезпечення. Для аналізу структури та оформлення документів було перевірено наявність заголовків, нумерації сторінок та змісту, а також оцінено читабельність тексту та його відповідність граматичним та орфографічним правилам. Було перевірено, що документи відповідають стандартам документації та мають зрозумілу структуру.

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Під час тестування програмного забезпечення «BingNUOS» для надання розкладу занять студентам НУК були проведені наступні тести для перевірки вимог до функціональних характеристик:

– Перевірка коректності вибору групи.
– Перевірка перегляду розкладу для обраної групи.
– Перевірка підписки та відписки від сповіщень.
– Перевірка доставки сповіщень за 10 хвилин до початку кожного заняття.
– Перевірка авторизації модератора та адміністратора в веб-сервісі.

– Перевірка правильності відображення розкладу для дозволених груп для модератора.

– Перевірка додавання, редагування та видалення розкладу модератором та адміністратором.

– Перевірка правильності відображення розкладу усіх груп для адміністратора.

– Перевірка створення, редагування та видалення груп та розкладів адміністратором.

– Перевірка створення, редагування та видалення модераторів та їх груп модерації адміністратором.

У таблиці Д.1 представлено перелік випробувань.

Таблиця Д.1 – Перелік випробувань основних функціональних можливостей

№	Дія	Очікуваний результат	Результат
1	2	3	4
1	Введення команди /group та вибір конкретної групи із меню Telegram-бота	Telegram-бот підтверджує вибір групи	Telegram-бот успішно підтвердив вибір групи
2	Введення команди /schedule та вибір пункту меню із робочим днем.	Telegram-бот відображає розклад занять для вибраної групи на вибраний робочий день	Розклад занять для вибраної групи на вибраний робочий день відображається правильно
3	Введення команди /subscribe /unsubscribe для підписки або відписки від сповіщень	Telegram-бот змінює статус підписки на сповіщення	Статус підписки на сповіщення успішно змінено
4	Наявність підписки на сповіщення перед початком заняття	Telegram-бот надсилає розклад занять за 10 хвилин до початку кожного заняття	Розклад занять надсилається за 10 хвилин до початку кожного заняття
5	Введення коректних облікових даних модератора або адміністратора	Веб-сервіс авторизує модератора або адміністратора	Модератора або адміністратора успішно авторизовано

Продовження таблиці Д.1

1	2	3	4
6	Авторизація модератора та відкриття головної сторінки веб-сервісу	Веб-сервіс відображає розклад для дозволених груп модератора	Розклад для дозволених груп модератора відображається правильно
7	Дії модератора або адміністратора з розкладом (додавання, редагування, видалення)	Веб-сервіс виконує відповідні дії з розкладом і зберігає зміни	Додавання, редагування, видалення розкладу виконано успішно
8	Авторизація адміністратора та відкриття головної сторінки веб-сервісу	Веб-сервіс відображає розклад для всіх груп	Розклад для всіх груп відображається правильно
9	Дії адміністратора з групами та розкладами (створення, редагування, видалення)	Веб-сервіс виконує відповідні дії з групами та розкладами і зберігає зміни	Створення, редагування, видалення груп та розкладів виконано успішно
10	Дії адміністратора з модераторами та їх групами (створення, редагування, видалення)	Програма виконує відповідні дії з модераторами та їх групами і зберігає зміни	Створення, редагування, видалення модераторів та їх груп модерації виконано успішно

В результаті проведення випробувань програмного забезпечення «BingNUOS» для надання розкладу занять студентам НУК були перевірені його функціональні характеристики. На основі результатів перевірки можна зробити висновок, що програмне забезпечення «BingNUOS» відповідає вимогам до функціональних характеристик, оскільки усі випробування дали очікувані результати.