Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки_____
(повна назва)

КВАЛІФАКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти ____другий (магістерський)_____
(рівень вищої освіти)

Спеціалізована веб-браузерна комунікаційна система між викладачами та студентами в освітньому процесі ХНУРЕ
(тема)

Виконав: студент 2 курсу, групи СКСм-20-1
_____Рог С.В._____
(прізвище, ініціали)
Спеціальність *123_Комп'ютерна інженерія*
Тип програми ____освітньо-наукова_____
(освітньо-професійна або освітньо-наукова)
Освітня програма_____
___*Спеціалізовані комп'ютерні системи*_____
(повна назва освітньої прграми)

Керівник роботи _доц. Шкіль О.С.___
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _Чумаченко С.В
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерної інженерії та управління _____

Кафедра_____Автоматизації проектування обчислювальної техніки_____

Рівень вищої освіти____другий (магістерський)_____

Спеціальність_____123_Комп'ютерна інженерія_____
(шифр і назва)

Тип програми _____Освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____Спеціалізовані комп'ютерні системи_____
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«__04__»_____11__ 2021 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові_____Рогу Сергію Вікторовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Спеціалізована веб-браузерна комунікаційна система між викладачами та студентами в освітньому процесі ХНУРЕ.
Specialized Web-browser Communication System for Teachers and Students in Educational Process of NURE.
затверджена наказом по університету від «_04_»_____11_____2021 р. №__1635_
2. Термін подання студентом роботи до екзаменаційної комісії _____24.12.2021_____
3. Вихідні дані до роботи (проекту)_____
Angular Framework, Web application
Web application
Cloud сервіси FireBase
Мова програмування TypeScript(JavaScript)
Середовище розробки Visual Studio Code

4. Перелік питань, що потрібно опрацювати у роботі_____
Аналіз існуючих рішень реалізацій комунікаційних систем
Обрання сучасного рішення для побудови спеціалізованої веб системи
Аналіз та реалізація специфічного функціоналу в системі
Розробка програмної частини системи
Імплементація клієнтського інтерфейсу системи

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 16 слайдів

6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

7. Дата видачі завдання 02.09.2021

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів роботи	Примітка
1	Видача теми проекту, узгодження і затвердження	02.09.2021-08.09.2021	
2	Аналіз проблемної галузі, постановка задачі, вибір засобів розробки	09.09. 2021-15.09. 2021	
3	Розробка моделі комунікаційної системи	16.09.2021-29.09.2019	
4	Налаштування серверної частини системи	30.09. 2021-13.10.2021	
5	Розробка програмної частини системи	14.10. 2021-31.10. 2021	
6	Побудова інтерфейсу користувача	01.11. 2021-15.11. 2021	
7	Проведення випробування системи	15.11. 2021-30.11. 2021	
8	Оформлення пояснювальної записки	01.12. 2021-15.12. 2021	
9	Перевірка виконаного проекту керівником	20.12. 2021-23.12. 2021	
10	Захист проекту	23.12. 2021-28.12. 2021	

Студент _____
(підпис)

Керівник роботи (проекту) _____ доц. Шкіль О.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 95 сторінок, 51 рисунок, 16 лістингів коду. 7 джерел за переліком посилань.

SPECIALIZED WEB SYSTEM, ANGULAR FRAMEWORK, FIREBASE CLOUD SERVICES, COMMUNICATION SYSTEM

Метою роботи є розробка спеціалізованої веб-браузерної системи для комунікації студентів та викладачів в освітньому процесу ХНУРЕ, та її реалізація за допомогою залучення сучасних технологій. Покращення іміджу ХНУРЕ та скорочення часу комунікації викладача між студентами під час контролю освітнього процесу.

Проведено аналіз комунікаційних можливостей викладачів між студентами в межах порталу дистанційного навчання ХНУРЕ, та загальних популярних комунікаційних систем в освітньому процесі університету. За результатом аналізу визначено функціональні та інші специфічні можливості системи.

До розробки веб-системи залучені технології такі як: Firebase; Angular Framework; NGRX; Tailwind; RXJS.

Виконане тестування побудованої системи для перевірки спроектованих функціональних можливостей. Змодельовано тестову ситуації комунікації викладача зі студентами в межах кімнати учбової групи студентів.

ABSTRACT

Master`s thesis contains: 95 pages, 5 pictures, 16 code blocks. 7 sources according to the list of link.

SPECIALIZED WEB SYSTEM, ANGULAR FRAMEWORK, FIREBASE CLOUD SERVICES, COMMUNICATION SYSTEM

The aim of the work is to develop a specialized web browser system for communication of students and teachers in the educational process of KNURE, and its implementation through the use of modern technologies. Improving the image of KNURE and reducing the time of teacher communication between students during the control of the educational process.

Performed analysis of the communication capabilities of teachers between students within the distance learning portal KNURE, and general popular communication systems in the educational process of the university. As a result of the analysis, the functional and other specific features of the system are determined.

Technologies such as Firebase, Angular Framework, NGRX, Tailwind, RXJS are involved in the development of the web system.

The system was tested to verify the designed functionality. The test situation of communication of the teacher with students within the room of the study group of students is modeled.

ЗМІСТ

ВСТУП

Людина повинна вміти вірно передавати свої думки іншим людям через голос, текст, малюнки і т.д. Адже без здатность спілкуватись та комунікувати з іншими людьми, цінність людини для соціуму зводиться до нулю.

Від печерних малюнків до телеграфів, від передачі листу голубами до радію, цей список можно продовжувати але сутність в тому що задача людини це - комунікація, і з кожним кроком людства, коммунікаційні примітивні технології прогресують в нині відомі сучасні цифрові.

З часів появлення інтернету, смартфону, та комп'ютера в кожному домі, зв’язок з рідними, колегами по роботі, одногрупниками по навчанню, став майже моментальним.

Прогрес в глобальних веб комунікаційних технологіях приніс можливість простій людині ділитися новинами на дистанції в голосовому, текстовому та відео форматах, в буквальному сенсі слова, на ходу, що дало початок зародженню індустрія інтернет чатів, месенджерів соціальних мереж, форумів, різноманітних сервісів, для певної аудиторії, на специфічну тематику, по цей день існує неймовірна кількість, а їх різноманітність і типи досі зростає .Зароджуються комунікаційні системи для корпоративних заходів з прицілом на функціонал для зв'язку груп людей в робочому процесі. Секретні комунікаційні системи державного рівня з нахилом на безпеку та захищеність потоку комунікації. Соціальні мережі з великою кількістю різноманітних додаткових можливостей і маркетингових заходів для залучення нових користувачів, окремо відведені форуми під задані тематики або інтереси.

Саме тому можна виділити, що сучасні комунікаційні інтернет-системи сильно відрізняються один від одного за ціллю її використання, використаними технологіями, сегменту аудиторію. Для звичайного користувача важливим є конфіденційність та захищеність його даних, чим скоріше як виключення забезпечена обрана ним система. Важливим аспектом

для користувача також є інтуітивність інтерфейсу та функціональних можливостей в обраній системі, що особливим чином впливає на тривалість використання, або на вибір комунікаційної системи в цілому.

В роботі розглянуто питання проектування спеціалізованих веб-комунікаційної системи. Аналіз технологій та рішень на ринку за результатом якого обрані технології та спроектована архітектура системи. Розглянуто специфічні функціональні можливості при комунікації між студентами та викладачами в ХНУРЕ.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд типів комунікаційних систем та їх спеціалізація

Проведення аналізу предметної області в цілях подальшого проектування спеціалізованої системи є завданням, що формує цілісний погляд на цілі, які в предметної області переслідуються.

Комунікація в Інтернеті може приймати мати різноманітні форми і відбуватися в багатьох контекстах. Жанри та традиційні способи спілкування з використанням комунікаційного засобу еволюціонували в Інтернеті. Ці жанри спілкування багато в чому відповідають офлайн контекстам людського спілкування.

1. Міжособистісна - Веб надає користувачам можливість створити домашню сторінку, яка зазвичай передає особисту або професійну інформацію. Практика створення домашньої сторінки виникла з технічної необхідності визначення сторінки за замовчуванням, яку веб-браузер відображає, коли запитує інформацію у веб-сервера, коли вказано лише ім’я хоста або ім’я хоста та каталогу. Домашні сторінки зазвичай є сторінками вищого рівня для сервера, організації чи персони. При створенні приватними особами домашні сторінки часто містять детальну конфіденційну інформацію про їх авторів і вказуються в каталогах домашніх сторінок. Також, користувачі часто дотримуються традиції розміщення посилань на сторінки колег або друзів, створюючи електронні групи. При міжособистісному використанні персональні домашні сторінки пропонують комунікацію один до одного, хоча технічна робота всіх сторінок в інтернеті виконується один до багатьох.

2. Група - як описано раніше, угрупування особистих сторінок в інтернеті можуть визначати конкретну групу Інтернету. Отже таким чином користувачі можуть створювати асоціації в Інтернеті, які не залежать від географії знаходження кожного користувача та орієнтовані на інтерес загальної теми організації. Розподілення інформації в інтернеті за деревом

тем часто розвивається внаслідок відвітвлень загальної тематики, що описує тему. Подібним чином групи людей об’єднуються в інтернеті на основі спільних інтересів у спілкуванні (наприклад, професіональна структура яка має веб-сервер для закликів проведення арт-виставок).

3 Організаційні - Багато веб-серверів що тільки з’явились у Інтернеті, належать організації, а не окремим фізичним особам, тому домашня сторінка сервера часто визначає установу або організацію, якій належить сервер. Таким чином на веб-серверах навчальних закладів розвивався жанр Campus-Wide Information System (CWIS). Аналогічно, комерційні, урядові та неурядові організації значною мірою дотримуються моделі, встановленої CWIS.

4 Маса – Також як інші засоби масової інформації використовувалися для поширення інформації «один до багатьох» (телебачення, газети, радіо,), інтернет, в наш час, також використовується для масової комунікації. Через інтернет поширюється багато комерційних і некомерційних журналів та інших видань. Більше того, як зазначалося раніше, усі загальнодоступні веб-сторінки потенційно доступні для читання будь-кому, хто використовує Інтернет, і, таким чином, потенційно є комунікаціями один до багатьох.

Ключова концепція, яку слід зрозуміти, полягає в тому, що Інтернет як комунікаційну систему можна гнучко використовувати для вираження різноманітних комунікацій. Класифікація комунікації (у категоріях, що перелічені раніше) залежить від того, хто бере участь у комунікації.

З огляду на аналіз специфікацій типів «багатокористувацьких» комунікаційних систем для побудови спеціалізованої системи для комунікації в рамках освітнього процесу ХНУРЕ ,CWIS орієнтована система є тим рішенням яке найкраще підходить подібному роду систем та буде обрано для подальшої її реалізації.

ХНУРЕ має велику екосистему веб сервісів таких як наприклад dl.nure або cist. При їх наявності, реалізуючи комунікаційну веб систему для ХНУРЕ, одним за найважливіших аспектів є входження у цю екосистему залученням її сервісів та використовуючи посилання на ресурси цих сервісів. Такі рішення є побажаннями опитаних викладачів та студентів що обуло

проведено для створення списку функціональних специфічних рішень. А отже спеціалізована система ХНУРЕ це є клієнто орієнтована, тобто система ща орієнтована на потреби студентів та викладачів.

1.2 Огляд сучасного стану проблеми

Помітно швидко впроваджується в освітній сектор технологія хмарних обчислень для покращення навчання процесу, досліджень та інших адміністративних процесів. більшість із існуючих реалізацій в світі відбувається в західних країнах, таких як Великобританія, США, тоді як рівень впровадження хмарних обчислень в Україні, та в подібних країнах що розвиваються, є рідкісним. Більше того, впровадження технології хмарних обчислень в освітній сектор вимагає прийняття рішень менеджерами відділу інформаційних технологій, наприклад, вибір відповідної моделі розгортання, систему хмарних послуг тощо, у відповідному університеті чи інституті. Вибір структур реалізації системи відіграє ключову роль, оскільки він може вплинути на вимоги різних зацікавлених сторін, таких як студенти, викладачі, адміністративний персонал співробітників IT-відділу тощо. Природньо, що при реалізації таких стратегічних ініціатив будь-яким навчальним закладом необхідно прийняти відповідне рішення, аналізуючи багато точок зору.

Саме те що впровадження подібного роду систем не відбувається, в вітчізняних розробках немає аналогів комунікаційних систем для університету щодо, тобто фактично досвід щодо організації таких спеціалізованих систем, та процесу їх розробки, відсутня.

Наявність «масових» та «групових», по класифікації що описана вище, систем гігантів на ринку затьмарює ринок своєю універсальністю відсутності націлювання на конкретну групу користувачів.

Обширний функціонал таких систем з одного боку є плюсом, з іншого – роздуває систему можливостями якими рядовий користувач може й не користуватися взагалі.

Використання таких систем для використання в освітній діяльності ХНУРЕ не завжди є комфортним, адже такі структури , як прийнято вважати, не мають потрібної архітектури для вільного та швидкого використання, адже як і було раніше сказано, загальні комунікаційні системи не є CWIS орієнтованими системами. Тому користувачі повинні об’єднуватись та реалізовувати цей патерн в рамках загальних веб комунікаційних систем (наприклад Facebook, Telegram). Це є некомфортним, не завжди результативним в процесі об’єднання користувачів що є членами університетської діяльності (студенти, викладачі та ін.) в групу за специфікацією адже не завжди користувач існує або його можна знайти в системі. Отже такі системи приносять дискомфорт в організації CWIS підходу.

За рахунок того що загальні системи є роздутими за функціоналом, та обмеженими наявністю специфічних можливостей, а наявність відомих рішень спеціалізованих систем в Україні відсутня, виникає потреба в орієнтованій системі комунікації для ХНУРЕ, в даному випадку в системі яка реалізує структурний підхід CWIS.

1.3 Електронні сервіси ХНУРЕ в освітнього процесі

Освітні процеси ХНУРЕ мають свою специфіку та злагодженість. Для контролю учбового процесу на протязі всього життя ХНУРЕ існують відповідальні сервіси що цим займаються. Наприклад можна виділити хід підготовки ресурсів навчального процесу, до яких умовно входить формування освітніх програм, забезпечення навчально-методичних комплектів та інші підготовчі процеси.

Також ХНУРЕ має свої сервіси по управлінню процесом навчання студента в університеті. До таких процесів відноситься: формування розкладу занять студентів; проведення атестаційних та контрольних робіт; контроль відвідування занять, формування звітності про успішність студенту та інші.

Більшість з таких процесів по контролю та підготовки ходу навчання студента в ХНУРЕ перейшли в електронну форму за допомогою спеціалізованих систем розроблених спеціально для ХНУРЕ та інших загальних рішень.

Рисунок 1.1 – Електронні сервіси управління навчальним процесом ХНУРЕ

Важливу роль в формування сучасного процесу навчання ХНУРЕ відіграють власні системи cist.nure.ua, ХНУРЕ ДН (Дистанційне навчання), та система комунікації між викладачем та студентами.

Cist – сервіс що має веб інтерфейс. Основна цінність як в порталі що є інформативною дошкою університету, репрезентує дані щодо сформованого процесу навчання, статистики успішності в навчанні різного роду, розклад, дані структури університету та інше. Також надає API взаємодія з котрим надає ті ж самі дані що транслюються до веб-інтерфейсу але в вигляді json файлів, що використовується для побудови комунікації, та роботи даними інших сервісів ХНУРЕ.

DL NURE (ХНУРЕ ДН) – учбовий портал дистанційного навчання ХНУРЕ, об’єднує в собі багато систем таких як: контроль відвідування заняття користувачем, доступ до ресурсів освітніх програм студента,

методичних вказівок та ін., вбудована система комунікації в межах освітніх програм студента.

Кожен з існуючих сервісів ХНУРЕ формує основу у сучасній взаємодії викладача та студента один між одним та між освітніми ресурсами.

Проведимо розбір та аналіз ХНУРЕ систем, а також зовнішніх систем, що залучені в процесі комунікації студентів з викладачами для виділення об'єктивних переваг та недоліків на основі котрих буде сформовано постановку задачі.

Розглядаючи систему dl nure, перш за все звертається увага на те що dl nure є агрегуючою системою, тобто поєднує в собі велику кількість функціоналу. Система має чимало можливостей з приводу контролю учбового процесу студентів, подачі освітнього матеріалу, контролю відвідуваності користувача та інше.

Із основних функціональних модулів можна виділити наступні.

1. Календар користувача - в якому відображені всі події що стосуються користувача, а це: заліки, відвідування пар та інше.

2. Секція з курсами студента – найголовніший функціональний модуль порталу dl.nure адже виконує найбільшу частину керування освітнього процесу студентів з перспективи дистанційного навчання. Цей модуль надає можливість користувачу переглянути методичні матеріали, відмітити свою присутність, прочитати останні замітки викладачем дисципліни/курсу.

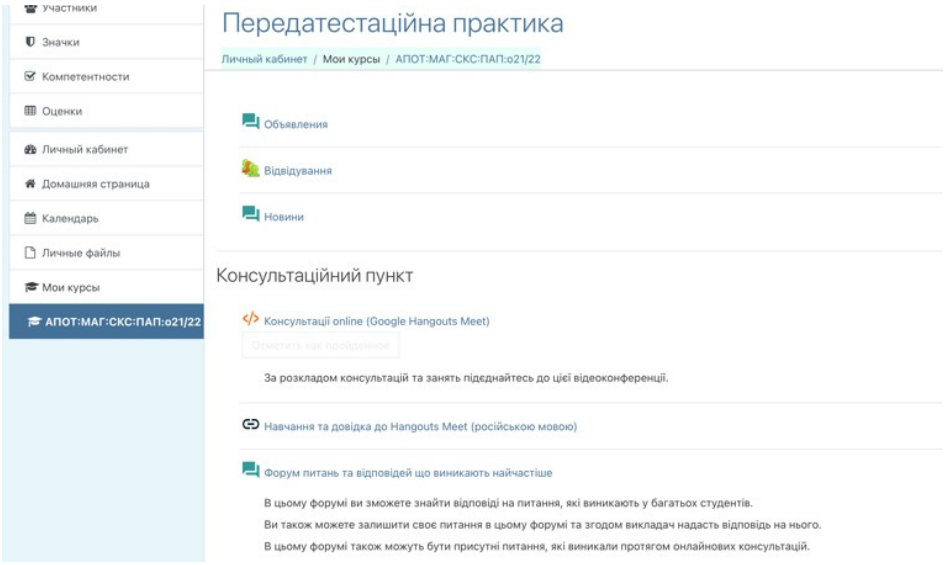


Рисунок 1.2 – Інформаційна сторінка обраного курсу

Окремої уваги треба приділити можливості комунікації в рамках dl.nure.

В функціональному модулі з курсами/освітніми програмами студента комунікація відбувається в односторонньо, тобто, коли викладач додає якісь новини до сторінки курсу, студент може їх продивитись.

Повноцінну систему комунікації реалізовано як другорядну, згідно з розташованого «входу» до функціоналу комунікації в системі та розташуванням веб елементів цієї системи. Доступ до якої знаходиться по кліку на іконку повідомлення в профілі користувача, також знаходиться в панелі в шапці.

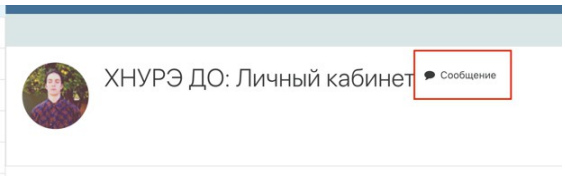


Рисунок 1.3 - Іконка доступу до повідомлень користувача

Після натискання на кнопку «Сообщение» відкривається панель повідомлень.

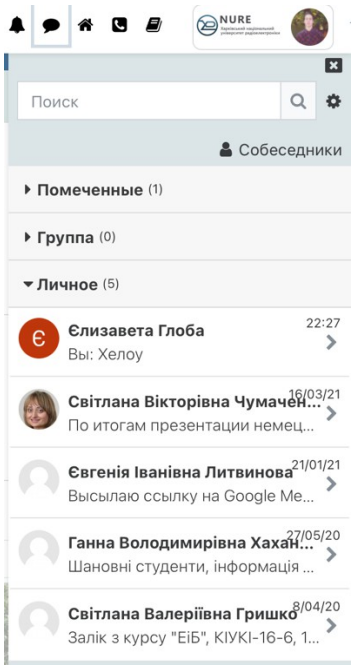


Рисунок 1.4 – Секція мессенджеру dl.nure.

Перейшовши до цієї секції (Рисунок 1.4) бачимо 3 групи чаті: помічене, група, персональне.

У вкладці помічене знаходяться помічені важливі, суб'єктивно, чати користувача.

Персональне – відображаються повідомлення зі зв'язком 1 користувач – 1 користувач.

Група – повідомлення що відносяться до зв'язку один – багато.

Для початку чатування потрібно знайти користувача з яким буде налогоджено зв'язок, це робиться за допомогою строки вводу що зображена в горі рисунку 1.4. Таким чином можна знайти будь якого користувача в dl.nure та почати обмін повідомленнями.

Створити групові повідомлення студенту не надано можливості, цей функціонал доступний викладачам.

Підводячи підсумки згідно комунікаційних можливостей в освітньому процесі можна виділити наступне із плюсів:

– Можливість публікації новин, файлів, та будь якої попереджувальної інформації викладачами для студентів в рамках дисципліни студента;

– Можливість примітивного чатування між викладачами та студентами на довільні теми.

З мінусів такої комунікації можна виділити наступне:

– Відсутність можливості надсилати файли функціонал чату;

– Відсутність доступу до створення групових чатів студентами;

– Пошук потрібних людей для формування групового чату.

Аналізуючи виділені плюси та мінуси можна отримати наступні висновки, що dl.nure надає вичерпну та комфортну систему по контролю та оформленню освітнього процесу студентів, та поєднує в собі комунікаційні елементи в, цьому сегменті системи, які надають більше доступ викладачам до повідомлення студентам інформації, в свою чергу студенти є отримувачами інформації в цьому сегменті системи системі.

З іншої сторони, хоча dl.nure і надає окремий функціонал чатування, але його технології є застарілими та не сучасними, адже використовують простий обмін текстовою інформацією, має старий та непомітний веб-інтерфейс. З огляду на це система комунікації не отримала популярності в повсякденному житті студента та викладача. Більшість викладачів знехтують використанням комунікаційної системи в dl.nure через її не інтуїтивність і складність. Суб’єктивно кажучи, за весь час навчання в університеті мені не знадобилось спілкуватись через цю систему, адже викладачі через описані вище проблеми надають побажання загальним, іншим системам, електронна пошта або google meet та подібні до цієї комунікаційної системи.

1.4 Комунікація в Google Meet в освітньому процесі ХНУРЕ

Причиною до того є недостача комунікаційних можливостей в електронних сервісах ХНУРЕ, що ми розібрали на прикладі dl.nure, для проведення більш продуктивної бесіди за студентами в рамках контролю оцінювання або консультацій.

Обрання стороннього до проведення освітніх заходів зі студентами не є поганим підходом, навпаки, це комфорт для викладача в проведенні якісного освітнього контролю над студентами. В свою чергу, студенти відчувають

різноманітність в таких сервісах адже один викладач може використовувати dl.nure, інший google meet, а третій викладач google classroom. Це приводить до заплутання студентів в запам’ятовування де лежить окремий матеріал по дисципліні окремого викладача, в якому сервісі.

Розглянемо приклад використання викладачем Google Meet.

Google Meet – безкоштовний сервіс, що надає можливість створювати відео конференції, залучати до конференції будь кого за посиланням, демонструвати екран, та інші сучасні можливості будь яких інших відео конференційних сервісів (Skype наприклад, або Microsoft teams). З навчального процесу в ХНУРЕ, деякі викладачі що використовують dl.nure та google meet, додають посилання до інформаційної дошки дисципліни.

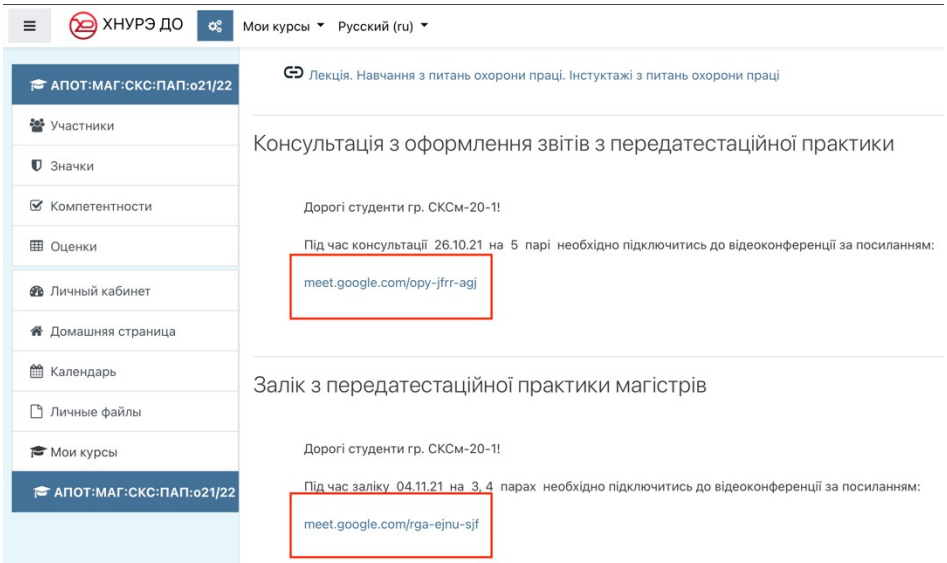


Рисунок 1.5 - Прикріплення посилання відео-конференції до інформаційної дошки дисципліни в dl.nure

По при таку чудову можливість прикріплення посилання до інформаційної дошки освітньої програми. Даний функціонал не є прямим сповіщенням студентів про майбутню відео-конференцію. Або існує велика можливість що викладач забуває додати посилання до інформаційної дошки. Тоді в випадку коли настає час проведення відео-конференції, викладач змушений встановлювати зв’язок зі старостою для сповіщення екстреної інформації (Рисунок 1.6).

Підбиваючи підсумки щодо переваг та недоліків залучення google meet в освітній процес ХНУРЕ можна виділити такі плюси:

– Сучасна безпечна відеоконференція, безпосередньо, має великі можливості по проведенню обліку студентів, контролю відвідуваності, бесіді зі студентами та викладачами в усному форматі;

– Залучення технології веб-конференції в освітнє життя є необхідним в умовах карантину;

Із мінусів можна виділити наступне:

– Неможливість приєднання файлу в чат конференції;

– Чат конференції, а тобто всі данні чату будуть видалені по завершенню конференції, що унеможливлює повторне до них доступ без дуплікації в інші сервіси;

– Повідомлення викладачем про екстрені конференції потребують залучення додаткового часу та комунікаційних можливостей (Рисунок 1.6).

1.5 Постановка задачі дослідження

Серед усіх зазначених електронних сервісів ХНУРЕ мало розвинутим є продуктивний та злагоджений процес комунікації. Хоча ХНУРЕ ДН і надає інтерфейс комунікаційної системи та можливість комунікації між студентами та викладачами, але в більшості, студенти та викладачі не використовують портал дистанційного навчання за цією потребою, навпаки, надають перевагу єдиним рішенням в якості обраної системи комунікації які не співпадають.

Також викладачі розпиляють студентів по зовнішнім сервісам які для викладачів є інструментарієм що надає послуги за специфічних потреб, такі як google meet, Microsoft teams, або Skype для проведення відео-конференцій на регулярній основі, та інші сервісів для забезпечення додаткового контролю освітнього процесу студентів, наприклад goggle class room.

Таким чином інформування студентів про будь які зміни відбуваються в цих системах не завжди забезпечують студента оповіщеннями про зміни. Що приводить викладачів до потреби зв’язку зі старостою групи студентів для злагодження процесу навчання (Рисунок 1.6).

Викладачі обирають індивідуальні переписки через пошту, а студенти формуються в групові бесіди по групам університету в популярних соціальних мережах. Така ситуація лишає ефективності в спілкуванні з студентами, в свою чергу віддаляючи спроможність контактувати та залучатись студентам та викладачам в спільні бесіди, що може сприяти покращення відношення між ними що в свою чергу покращує підсумковий загальний рівень успішності студента в навчанні та викладача в наданні якісних знань.

Рисунок 1.6 – Зв’язок викладача з групою через старосту в ХНУРЕ.

Тому виникає потреба в системі, що забезпечить швидший процес контактування викладача з групою студентів та навпаки в екстрених випадках, проведення збору контрольних робіт або робіт іншого роду, в системі, що допоможе зберігати надіслані файли в контексті групи студентів а не в контексті освітньої програми та інших зовнішніх сервісах, що допоможе швидко обмінятися або надіслати файл між студентами та викладачами. Така система допоможе позбавитись участі старости як елементу зв’язку між студентами та викладачами в більшості випадків.

Комунікаційна система для ХНУРЕ між викладачами та студентами також грає велику роль в формуванні іміджу університету, що в свою чергу приверне більше викладачів та студентів.

Нова система змінить парадигму в комунікації студентів та викладачів за допомогою їх зближення в одній системі, що має позитивні наслідки в покращенні успішності в роботі на освітніми програмами обох типів користувачів системи.

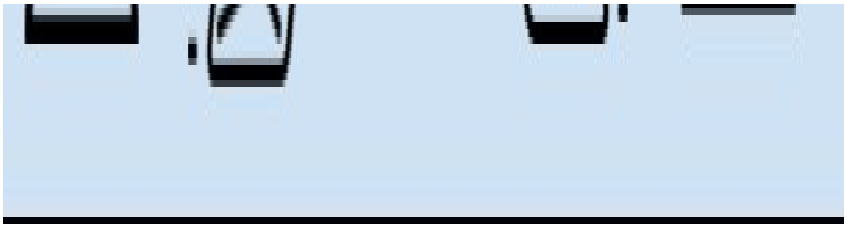


Рисунок 1.7 – Процес зв’язку в комунікаційній системі ХНУРЕ, що планується побудувати.

Об’єкт дослідження в роботі – процес комунікації між студентами та викладачами за допомогою веб системи розробленої для ХНУРЕ.

Предмет дослідження – способи та можливості уніфікації процесу комунікації в межах веб систем ХНУРЕ. Функціональні можливості комунікаційного процесу в межах структури ХНУРЕ.

Мета роботи – побудувати додатковий веб сервіс ХНУРЕ що забезпечить комунікаційними можливостями студентів та викладачів один між одним в рамках структури ХНУРЕ без додаткових витрат часу. Побудування інтуїтивного інтерфейсу системи. Покращення іміджу ХНУРЕ.

Для досягнення мети, потрібно вирішити наступні задачі:

– Провести аналіз комунікаційних можливостей та способів комунікації в ХНУРЕ за допомогою власних сервісів та зовнішніх:

– Теоретично розробити функціонал системи з огляду на аналіз існуючих та покращення іміджу ХНУРЕ;

– Проаналізувати та обрати сучасні технологічні рішення для реалізації подібного роду систем;

– Отримати дані згідно структурної побудови ХНУРЕ для використання в побудові системи;

– Розробка веб системи для комунікації між викладачами та студентами ХНУРЕ;

– Тестування технічних можливостей побудованої та комунікації між викладачами та студентами в побудованій системі.

1.6 Теоретичні розробки

За результатами проведення аналізу предметної області та висвітлення сучасної проблематики існуючих рішень був зібраний концепт та напрям подальшого просування в теоритичній реалізації системи для ХНУРЕ.

По перше, система CWIS повинна бути реалізована згідно архітектури структурних одиниць університету ХНУРЕ з їх ієрархічної принадлежністю, тобто: факультети; напрями; спеціалізації; та групи. Слідуючи побудуванню системи з організацією структури університету, з’являєтсья можливість реалізувати CWIS системи в кінцевому вигляді якої, структурна одиниця «група» буде кінцева для користувача системи. Комунікація буде проводитись в рамках групи університету. А утримання ієрархічного зв’язку цих структурних елементів допоможе в подальшому організовувати групи користувачів в рамках, наприклад, однієї спеціальності і т.д. Наприклад студенти що мають приналежність до групи СКСм-20-1 будуть об’єднані та зможуть обмінюватися інформаціє один між одним в системі одразу, без будь яких додаткових дій в системі як це могло бути при використанні загальних систем.

По друге, в систему буде додано функціонал що спрямований на інтереси різних ролей користувачів , реалізація дозволів до системних функціоналів різним ролям користувачів. А саме, для студентів, викладачів, та в подальшому для адміністрації системи та старост груп або модераторів інших структурних одиниць групових об’єднань (факультетів, спеціальностей і т.д).

Після проведення бесід безпосередньо з представниками різних по ролям людей що задіяні в освітньому процесі ХНУРЕ, а саме викладача та студента, були проаналізовані та затверджені функціональні ідеї.

1. Розробити систему автентифікації користувачів за email з доменом @nure.ua. Мається на увазі що, будь яка людина що має пошту з доменом nure@ua є аутентифікованими та мають змогу авторизуватися в системі без введення паролей або іншої конфіденційної інформації, процес авторизації проходить лише через підтвердження що приходять на пошу nure@ua користувача.

2. Розподілення користувачів на ролі, що буде дозволяти викладачам увійти в будь яку кімнату системи а студентам лише знаходится в кімнаті своєї групи. Також підготування інших ролей в системі та їх функціональних можливостей винесене в чергу до реалізації, а саме наступні ролі: адміністратор; голова групи. Адміністратор системи буде мати доступ до сторінки адміністрування, в свою чергу ця сторінка буде включати в себе інтерфейс до модерації та адміністрування системи та їх користувачів. Голова групи буде мати більший за роль студента функціональний доступ в системі, а саме: модерування контенту в кімнаті групи; модерування принадлежності членів кімнати до цієї кімнати, та інші можливості по модеруванню контенту в рамках групової кімнати бесіди.

3. Передача медіа файлів в рамках кімнат бесід.

4. Приєднання доступу до розкладу групи в «шапці» кімнати групи. Винайти функціональну можливість отримання доступу до порталу cist.nure.ua до специфічної групи для отримання її розкладу.

Списком вище зазначено основні пріоритетні функціональні можливості що будуть взяті до розробки. Інші пріоритетні можливості, наприклад такі як; функціонал приєднання-від’єднання файлу до групи; індивідуальні кімнати бесіди та під’єднання посилань на ресурси інших систем ХНУРЕ та ін., залишаються на другому пріоритеті.

2 ВИБІР ТЕХНОЛОГІЧНИХ РІШЕНЬ ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Огляд та вибір сучасних технологій проектування програмного продукту

Для реалізації веб комунікаційної системи, концептуально, потрібна технологія реалізації клієнтського застосунку, база даних, сховище великих даних, автентифікаційна система, та застосунок API який надасть доступ в комунікації клієнтського застосунку до баз даних, сховищ, автентифікаційних систем та інших сервісів.

В сучасному підході до розробки веб застосунків використовуються наступні поняття:

– Frontend – клієнтська сторона користувальницького інтерфейсу до програмно-апаратної частини сервісу;

– Backend - програмно-апаратна частина сервісу, що відповідає за функціонування його внутрішньої частини.

Першою ідеєю було реалізувати дві частини системи, тобто фронтенд та бекенд, але під час оцінки ризиків та часу який знадобиться на реалізацію головної бекенд частини, було вирішено позбавитись ідеї її побудови власними силами, адже тільки реалізація цієї частини могла б зайняти декілька місяців, пішовши б на це я ризикував би не вкластися у відведений час реалізації системи. Плюс до того, через специфіку системи потребується база даних реального часу на заміну класичним базам, це означає більше часу на вивчення програмних рішень щодо налаштування та включення подібної технології в бекенд частину та багато інших моментів що відсторонили б мене від роботи безпосередньо над функціоналом системи та зосередило на розробці технічної інфраструктури на нижчому рівні. Саме цьому бекенд частину було вирішено замінити існуючими рішеннями, а тобто, хмарними сервісами.

Основною частиною роботи буде реалізація клієнтського коду, налаштування усіх серверних хмарних сервісів, реалізація структури даних

системи, реалізація веб дизайну системи та комфортного інтерфейсу користувача.

На ринку технологічних рішень з реалізації фронтенд частини застосунку існує величезна кількість фреймворків, є ті що укорінились фаворитами на ринку рішень, це – React, Vue JS, та Angular. Є ті рішення що зараз штурмують ринок та є популярними, ціль яких прискорити процес написання бізнес застосунків в веб. Але такі рішення не завжди є універсальними та в більшості разів в той час як припинялась хвиля популярності або на прикладі реалізацій новими фреймворками реального проекту – становилось видно що більшість з нових рішень є далеко не кращими.

Тому було вирішено проаналізувавши ведучі 3 рішення, а саме: React; Vue Js; Angular.

Далі буде наведено перелік переваг та недоліків кожного з 3-х рішень:

а) Angular.

Переваги наступні.

1 Angular використовується разом з Typescript. Він має виняткову підтримку для цього.

2 Детальна документація, що дозволяє розробнику отримати всю необхідну інформацію, не вдаючись до його колег. Однак це потребує більше часу для навчання.

3 MVVM (Model-View-ViewModel), яка дозволяє розробникам працювати окремо над тим самим розділом програми, використовуючи той самий набір даних.

4 Структура та архітектура, спеціально створені для великої масштабованості проекту.

Недоліками є наступний список.

5 Різноманітність різних структур (Injectables, Components, Pipes, Modules тощо) ускладнює вивчення порівняно з React і Vue.js, які мають лише «Component».

6 Відносно повільна продуктивність з огляду на різні показники. З іншого боку, це можна легко вирішити, використовуючи так званий ChangeDetectionStrategy, який допомагає вручну контролювати процес рендерингу компонентів.

б) React.

Перевагами React є наступні моменти зазначені у списку.

1. Легко вивчити завдяки простому дизайну використання JSX (HTML-подібний синтаксис) для шаблонів і дуже докладної документації. Розробники витрачають більше часу на написання сучасного JavaScript і менше турбуються про код, специфічний для фреймворку.

2. Дуже швидка завдяки реалізації React Virtual DOM і різним оптимізаціям рендерингу.

3. Відмінна підтримка рендерингу на стороні сервера, що робить його потужною платформою для контент-орієнтованих програм.

4. Першокласна підтримка Progressive Web App (PWA) завдяки генератору додатків `create-react-app`.

5. Прив'язка даних є односторонньою, що означає менше небажаних побічних ефектів.

6. Redux, найпопулярніша платформа для керування станом додатків у React, її легко вчити та використовувати.

7. React реалізує концепції функціонального програмування (FP), створюючи простий у тестуванні і код, що багаторазово використовується.

8. Програми можуть бути створені за допомогою TypeScript або Facebook Flow, що мають вбудовану підтримку JSX.

9. Перехід між версіями, як правило, дуже простий: Facebook надає кодові модулі для автоматизації більшої частини процесу.

10. Навички, отримані React, можуть бути застосовані до розробки на React Native.

Недоліками є наступний перелік нижче.

1 React не однозначний і залишає розробникам можливість вибирати найкращий спосіб розвитку. Це може бути вирішено сильним лідерством проекту та добрими процесами.

2 Спільнота ділиться за способами написання CSS у React, які поділяються на традиційні таблиці стилів (CSS Modules) та CSS-in-JS (тобто Emotion та Styled Components).

3 React відходить від компонентів на основі класів, що може стати на заваді розробникам, яким комфортніше працювати з об'єктно-орієнтованим програмуванням (ООП).

4 Змішування шаблонів з логікою (JSX) може спантеличити деяких розробників при перших знайомствах з React.

в) VueJS

Перевагами VueJs. Є наступний список зазначений нижче.

1. Посилений HTML. Це означає, що Vue.js має багато характеристик схожих з Angular, а це завдяки використанню різних компонентів допомагає оптимізації HTML-блоків.

2. Детальна документація. Vue.js має дуже докладну документацію, яка може прискорити процес навчання для розробників та заощадити багато часу на розробку програми, використовуючи лише базові знання HTML та JavaScript.

3. Адаптивність. Може бути здійснений швидкий перехід від інших фреймворків до Vue.js через схожість з Angular та React з погляду дизайну та архітектури.

4. Чудова інтеграція. Vue.js можна використовувати як для створення односторінкових програм, так і для більш складних веб-інтерфейсів. Важливо, що невеликі інтерактивні елементи можна легко інтегрувати до існуючої інфраструктури без негативних наслідків.

5. Масштабування. Vue.js може допомогти в розробці досить великих шаблонів багаторазового використання, які можуть бути зроблені майже за той самий час, що і простіші.

6. Крихітний розмір. Vue.js важить близько 20 КБ, зберігаючи при цьому свою швидкість і гнучкість, що дозволяє досягти кращої продуктивності в порівнянні з іншими платформами.

Недоліками Vue JS зазначено наступні моменти в списку нижче.

1. Нестача ресурсів. Vue.js, як і раніше, займає досить невелику частку ринку в порівнянні з React або Angular, що означає, що обмін знаннями в цьому середовищі все ще знаходиться на початковій стадії.

2. Ризик надмірної гнучкості. Іноді у Vue.js можуть виникнути проблеми при інтеграції у величезні проекти, і поки що немає досвіду можливих рішень, але вони обов'язково з'являться найближчим часом.

Провівши аналіз трьох технологій, я дійшов рішення на користь Angular Framework, причини цьому вибору наступні: перша і головна причина – об’єктна орієнтована програмна розробка; комфортна в побудові великих систем; відома мені технологія.

2.2 Аналіз програмних середовищ та вибір підходящого

Розробники використовують різні інструменти та фреймворки під час розробки програмного забезпечення, будь то для створення коду, проектування UI-UX або тестування. Ці інструменти ретельно відбираються на ринку для певного процесу розробки, який забезпечує практичну продуктивність, особливо для Angular розробки.

Інструменти та фреймворки розробки включають різні типи текстових редакторів, компіляторів, бібліотек коду тощо. Оскільки Angular продовжує залишатися найкращою платформою JavaScript для розробки мобільних і веб-додатків, разом з цим збільшується кількість IDE (Інтегроване середовище розробки).

Інтегровані IDE не обмежуються лише внутрішньою розробкою; насправді, їхня ефективність і продуктивність можуть бути корисними при розробці інтерфейсу.

Вибір програмних середовищ це більше вибір вподобання інтерфейсу середовища. Існує декілька популярних рішень які ідеально підходять для розробки застосунків та великих систем за допомогою Angular, такі як: Visual Studio Code; Web Storm; Angular IDE та ін.

Angular IDE розроблено спеціально для Angular Framework. Він доступний як окремий плагін, так і в рамках плагіна Eclipse. Це найкраща безкоштовна (інтегроване середовище розробки) IDE, оскільки вона проста для навчання новачкам і надає більше можливостей експертам Angular.

Перевага Angular IDE полягає в тому, що вона швидша та ефективна і повністю оптимізована для використання всіх переваг фреймворка.

Webstorm — це редактор коду, створений на IntelliJ. Він також надає вбудовані елементи для TypeScript і, таким чином, є чудовим редактором для програм Angular. Він також має середовище плагінів із включеною конфігурацією та функціями локальної історії тощо. Цей редактор коду Angular також зменшує вимоги до сторонніх плагінів, автоматично генеруючи код TypeScript.

Visual Studio Code — це потужний редактор коду, створений для постійної допомоги веб-розробникам. Його підсвічування синтаксису, фрагменти та функції рефакторингу коду допомагають полегшити розробку.

Цей редактор коду для Windows, OS і Linux розроблений Microsoft. Він пропонує підтримку коду TypeScript, інтелектуальне автоматичне завершення коду, функції, налагодження та інтегровані модулі.

Підбиваючи висновки по вибору середовища розробки, найкращим вибором буде слідувати досвіду використання цих середовищ. Я мав змогу працювати в Visual Studio Code та у Web storm, Обидва рішення маються право буди використаними, але найбільш комфортно працювати в Visual Studio Code через юзер інтерфейс цього середовища, він є інтуїтивним, та наповненим функціоналом. Також дуже комфортний вбудований інтерфейс управління Git безпосередньо в середовищі ідеальний для використання. Ще менеджер плагінів та Itelisence підсвічення коду прийшло до вподоби за вичерпну інформативність.

Для підвищення швидкості розробки програмної частини системи, або для рішення типових проблем при використанні того чи іншого программного рішення, наприклад для форматування коду в файлах, або включення нових itelisence рішень для css та html файлів, чи для доповнення git фунціоналу в межах середовища використовуються зовнішні плагіни. Наступні плагіни є популярними і рекомендованими рішеннями для покращення робочого процесу в середовищі Visual Studio Code при роботі з Angular Framework.

1. Angular language service – цей плагін вносить покращення в процесі редагування html документів наряду зі зв’язкою з компонентами ангуляр. Ці покращення виражаються в підсвіченні різним кольором елементів html. Надає підсказки при написанні коду в html документі, добавляє можливість переходу до Angular файлу з кодом елементу html що репрезентується в відповідному елементі.

2. IntelliSense для імен класів CSS у HTML - розширення Visual Studio Code, яке забезпечує авто-завершення імені класу CSS для атрибута класу

HTML на основі визначень, знайдених у вашій робочій області або зовнішніх файлах.

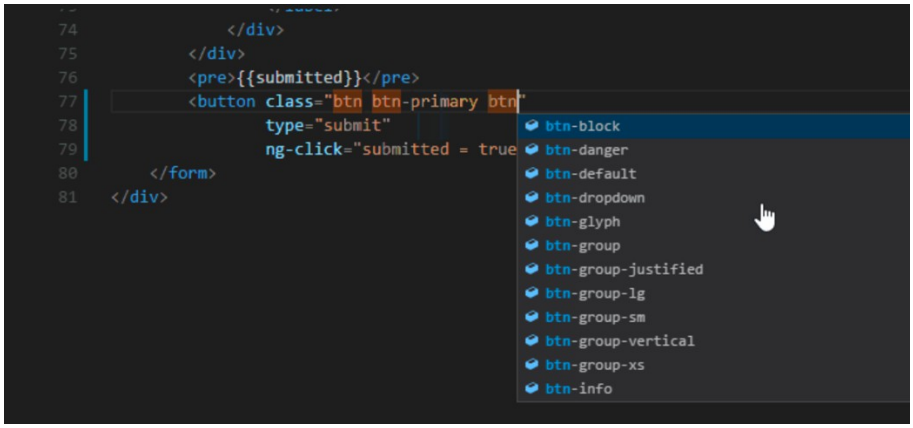


Рисунок 2.1 – Приклад авто-доповнення імені css класу за допомогою плагіну «IntelliSense для імен класів CSS у HTML»

3. Prettier Formatter for Visual Studio Code - це форматувальник коду. Він забезпечує чистий стиль коду, аналізуючи його і повторно його за своїми власними правилами, які враховують максимальну довжину рядка, обертаючи код в інші конструкції, коли це необхідно.

4. TSLint - це інструмент статичного аналізу, який перевіряє код TypeScript на читабельність, ремонтопридатність та помилки функціональності. Він широко підтримується в сучасних редакторах і системах збірки, і його можна налаштувати за допомогою ваших власних правил, конфігурацій і форматувальників.

5. Tailwind CSS Itelisence – це плагін який варто застосовувати разом з пакетом Tailwind якщо другий присутній в проекті. Tailwind CSS IntelliSense покращує досвід розробки Tailwind, надаючи користувачам Visual Studio Code розширені функції, такі як автозаповнення, підсвічування синтаксису та лінтування.

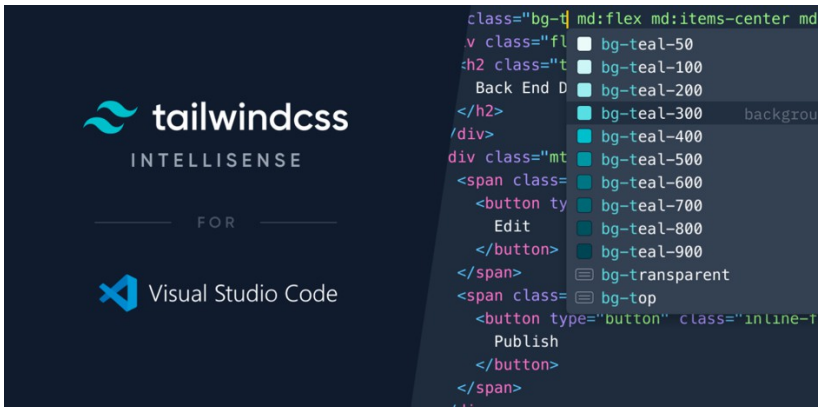


Рисунок 2.2 – Приклад автозаповнення імен Tailwind класів в html файлі

Перелік плагінів не є остаточно кінцевим а може бути модифікований згідно до вподобань розробника. Такі плагіни як GitLens та Autoimport, наприклад, також можуть місце в списку плагінів до використання, адже вони є універсальними та покращують досвід розробки Angular застосунків та систем.

2.3 Система хмарних сервісів для комунікаційної веб системи

Згідно до пункту 1.3 де були описані теоретичні розробки та ідеї функціоналу та архітектури системи що планується побудувати, ключовим є обрання найбільш підходящої системи хмарних сервісів для побудування прогресивної комунікаційної системи що б повністю покривало вимоги реалізації, а саме потрібні: автентифікаційна система; база даних з технологієї підтримки подвійного зв’язку комунікації з базою – realtime database технологією; сховище великих об’єктів, легкий прикладний інтерфейс для обраної технології програмної розробки Angular Framework; функціонал масштабування системи для потреб у майбутньому.

Через різноманітність вибору хмарних сервісів що надаються як BaaS (Backend as a Service) маємо проаналізувати існуючі рішення перед вибором хмарної, такі як:

– Back4App;

- Parse;
- AWS Amplify;
- Firebase.

Через схожість та взагалі не великі відмінності структур та наданих сервісів приведених вище хмарних сервісів, варто загострити увагу на AWS Amplify та Firebase, через те, що Back4App та в Parse Налаштувати локальний сервер аналізу є складним процесом, розгортання застосунків потребують професіональних навичок та не є легким для не спеціалістів в цих клауд платформах.

В свою чергу у Back4App для розміщення застосунки знадобиться додатковий постачальник платформи, необхідно створювати та використовувати власну базу даних, та потрібно надати багато зусиль для створення бекенд середовища. Ці моменти викликають складнощі в процесі розробки комунікаційної вуб системи, на цих клауд системах, через відволікання на другостепеневі складні моменти розробки системи вцілому.

Проведемо аналіз та порівняння AWS Amplify та Firebase для остаточного вибору.

1. AWS Amplify – один з кращих хмарних сервісів постачальників BaaS з відкритим вихідним кодом, який може надати доступ до повного набору інструментів, який створений, щоб допомогти розробникам і організаціям, полегшуючи їм керування бекенд-сервісами.

Може допомогти розробникам створювати безсерверні програми, які легко інтегрувати з інтерфейсом програми на основі JavaScript. Більше того, AWS amplify містить надзвичайно простий у використанні та навігаційний інтерфейс користувача, щоб ефективніше виконувати процеси, що стосуються серверних служб.

Більше того, AWS пропонує бібліотеку, яка дозволяє розробникам легко підключати свої серверні рішення до інтерфейсу мобільних додатків. Це, безсумнівно, чудова можливість для людей, які шукають високоефективну альтернативу Firebase.

Служби AWS Amplify легше інтегрувати в React, iOS, Android та інші популярні веб-фреймворки, включаючи React Native. Найважливіше те, що

він містить повні рекомендації щодо стилю, який може допомогти розробникам створити ефективні компоненти інтерфейсу користувача, які зможуть легко працювати з додатком, розробленим за допомогою AWS.

До плюсів AWS можна віднести наступне: легко налаштувати сервіс; легкий при роботі для початківців; стабільна та висока продуктивність сервісу.

Що-до мінусів AWS Amplify, то сюди можна віднести високу цінову політику до використання сервісу. Для використання – обов'язкова оплата. Ціна залежить від типу трафіку що користувач отримує. При поганому налаштування AWS може коштувати більше аніж це могло бути.

2. Firebase – система хмарних сервісів що пропонує такі функції, як аналітика, бази даних, комунікації, сповіщення про аварійне завершення роботи тощо. Це дозволяє користувачу відкинути в бік всі серверні моменти щодо реалізації та зосередитися на своїх клієнтах та розробці бізнес рішення. Firebase розроблено та масштабується ресурсами Google, навіть у найбільших додатках. Продукти Firebase добре працюють з Android та iOS. Firebase обмінюється інформацією та статистикою.

До плюсів Firebase можна віднести наступне: аутентифікацію можна здійснити за допомогою електронної пошти та пароля, Facebook тощо; Firebase надає дані в режимі реального часу; готовий API надається безпосередньо Firebase; резервне копіювання файлів здійснюється Google Cloud; Файловий хостинг Firebase є статичним; дані репрезентовані як потоки; підтримка масштабування; позбавлення користувача від участі до розробки інфраструктури.

До мінусів входять наступні пункти: традиційні реляційні моделі даних не застосовуються до NoSQL, так як Firebase використовує NoSQL бази даних;

Firebase унеможливлює запити до великих баз даних.

Проаналізувавши всі важливі плюси та мінуси обох хмарних систем можна дійти до наступних важливих висновків що допоможуть обрати найбільш підходящу хмарну систему для побудови комунікаційної системи.

Firebase є швидшим з огляду на працездатність її сервісів та системи взагалі. Також Firebase є дешевшим варіантом аніж AWS Amplify. В свою чергу AWS Amplify є open-source системою, а отже більший доступ користувач має до аналізу елементів системи що у зв'язці з комьюніті AWS Amplify допоможе швидко знайти проблему у разі її виникнення.

Отже, згідно висновку з порівнянням технологічних аспектів, цінової політики, та можливостей розвитку та підтримки продукту, кращим вибором буде використання Firebase через її малу вартість, простоту інтерфейсу до сервісів, прямою інтеграцією з Google інфраструктурою сервісів.

2.4 Залучення технологічних рішень до програмної частини системи

По при важливі рішення що-до front-end фреймворку та хмарної системи BaaS які були зроблені в користь Angular Framework та Firebase відповідно, не менш важливу роль грає обрання шляху реалізації програмної частини системи. Правильно обрані технології та рішення в програмній частині – це успіх довгострокового життя системи, її легкості в питанні розширення або підтримки, залучення сумісних технологій у майбутньому, та реалізації теорітичних розробок що-до функціональних можливостей комунікаційної системи для ХНУРЕ. Важливо взяти до уваги технологічні рішення, паттерни та доступні популярні реалізації цих рішень для використання в Angular Framework.

2.4.1 Збереження стану, паттерн Redux та його реалізація NGRX

Angular Framework сам по собі володіє різними інструментами для зберігання та маніпулювання. Є компоненти всередині яких і в межах яких можна зберігати стан, і вибудовувати залежність від батьківських і дочірніх компонентів. Наприклад, є компонент, що містить масу користувачів у системі (компонент списку користувачів) і компонент дочірнього, який приймає дані від окремого користувача та працює вже з ним (компонент користувача). Таким чином будується залежність і виникає стан. До того ж

можна побудувати залежність і навпаки від дочірнього до батьківського. Але час від часу цього недостатньо. Компоненти зберігають стан всередині себе, і для передачі цього стану обмежувати компоненти зв'язку, а це погано і не завжди можливо. На допомогу в таких ситуаціях з’являються «сервіси».

В сервісах можна зберігати певний стан. За допомогою ін’єкції залежності ми передаємо їх у компоненти, створюємо зв’язок між ними, роблячи компонент підписником на певні дані або навпаки ініціатором змін. Можна використовувати, але для ростючих та великих додатків використання одних сервісів не достатньо. Їх кількість буде розвиватися, рівно як і система стану на їх основі. Так же може з'явитися проблема відстеження ініціатора подій, Намагатися це виправити – рівно написанню своєю власної стейт менеджменту системи.

Готова реалізація стейт менеджмент системи для Angular є NGRX - це група бібліотек, що надає реалізацію Redux паттерну для роботи з Angular Framework. з використанням бібліотеки RxJs.

У свою чергу Redux - це реалізація підходу зберігання подій у додатках з яких збирається стан.

Основні принципи концепції Redux наступні:

– Store – сховище для станів застосунку,
– Actions – об'єкти дій, що описують, що відбулися в системі,
– Reducers – функції перетворення події в стан.

Бібліотека використовує шаблони функціонального програмування – стан декларується як незмінний, а редуктори повинні бути описані як чисті функції. При такому підході спрощується тестування і відлагодження програми.

Такими словами NGRX - це набір модулів, які надають реалізацію системи управління станом застосунку для Angular Framework застосунків.

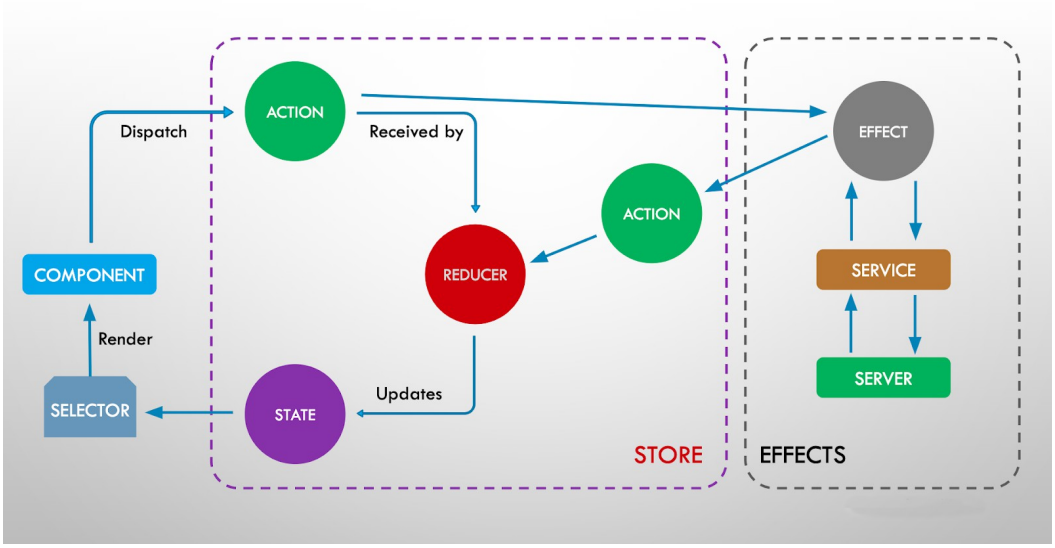


Рисунок 2.3 – Структурна схема взаємодії елементів інфраструктури NGRX

2.4.2 Підтримка інтернаціоналізації в системі. Бібліотека i18n

В комунікаційній системі що розробляється варто приділити увагу інтернаціоналізації системи, а тобто реалізувати підтримку декількох мов для комфортного використання як для громадян України так і її гостей що навчаються в ХНУРЕ. Для початку бажано зробити підтримку Англійської мови, другорядною задачою буде підтримка Української мови в системі. Зробити це допоможе механізм переводу та інтернаціоналізації тексту по всьому програмному проекту – i18n бібліотека для Angular.

Інтернаціоналізація (i18n) – це процес підготовки програмного застосунку, до підтримувати різноманітних мов. Інтернаціоналізований підтримує вимоги локальних ринків по всьому світу, функціонуючи більш належним чином на основі місцевих норм і краще відповідаючи очікуванням користувачів у країні.

I18n зосереджено на розробці продукту так, щоб програмна база коду була спроможна підтримувати світові мови, їх форматування та поведінку, специфічну для певної мови. Це важливий і іноді недооцінений крок, який необхідний для успішного та масштабованого L10n. І навпаки, локалізація

робить продукт специфічним для цільового ринку чи регіону(ів), включаючи переклад інтерфейсу та можливу адаптацію термінології тощо.

Реліз програмного застосунку, яке не було належним чином інтернаціоналізовано, може призвести до багатьох проблем, включаючи відсутність визнання на ринку. Усунення та виправлення помилок i18n після випуску продукту призводить до додаткових витрат, часу, ресурсів і потенційно зіпсованої репутації на місцевих ринках. Переробка продукту для глобального ринку після випуску, на відміну від інтернаціоналізації під час розробки, може зайняти багато часу, коштувати дорого і забезпечити перевагу альтернативним продуктам.

Переваги програмного забезпечення i18n включають:

– Більш якісне програмне забезпечення, яке відповідає технічним та культурним потребам різних регіонів;
– Зменшення часу, витрат і зусиль для локалізації (L10n);
– Єдиний вихідний код для всіх мов продукту;
– Простіше й легше обслуговування для майбутніх ітерацій продукту;
– Більше сприйняття та задоволення клієнтів всередині країни.

2.4.3 Технологія реактивного програмування RXJS

Повною формою RxJS є реактивним розширенням для Javascript. Це бібліотека javascript, яка використовує observable для роботи з реактивним програмуванням, яке має справу з асинхронними викликами даних, зворотними викликами та програмами на основі подій. RxJS можна використовувати з іншими бібліотеками та фреймворками Javascript. Він підтримується javascript, а також typescript.

Згідно з офіційним веб-сайтом RxJS, він визначається як бібліотека для складання асинхронних програм і програм на основі подій за допомогою спостережуваних послідовностей. Він забезпечує один основний тип, Observable, типи супутників (Observer, Schedulers, Subjects) та оператори, створені за допомогою Array#extras (map, filter, reduce, every тощо), щоб дозволити обробляти асинхронні події як колекції.

Нижче наведено переваги використання RxJS.

1. RxJS можна використовувати з іншими бібліотеками та фреймворками Javascript. Він підтримується javascript, а також машинописом. Кілька прикладів: Angular, ReactJS, Vuejs, nodejs тощо.

2. RxJS — це бібліотека, коли справа доходить до обробки асинхронних завдань. RxJS використовує спостережувані елементи для роботи з реактивним програмуванням, яке має справу з асинхронними викликами даних, зворотними викликами та програмами на основі подій.

3. RxJS пропонує величезну колекцію операторів у категоріях математики, перетворення, фільтрації, корисності, умов, обробки помилок, приєднання, що полегшує життя при використанні з реактивним програмуванням.

Нижче наведено недоліки використання RxJS.

1. Налагодження коду за допомогою Observable є невеликою складністю.

2. Коли ви починаєте використовувати Observables, ви можете закінчити свій повний код загорнутим під спостережувані.

По при недоліки, які на справді не є критичними або взагалі вагомими для розгляду можливості відмовитися від даної технології, RXJS є ключем до розв’язку завдання щодо реалізації «онлайн» систем. В прикладі реалізації комунікаційної системи ця технологія є єдиним шляхом реалізації чатування в реальному часі.

2.4.4 Angular Firebase SDK

Для того щоб мати змогу взаємодіяти з Firebase сервісами, Firebase запровадив Java script SDK що надає розробникам вискокорівневий API для взаємодії безпосереднью з будь яким сервісом Firebase (автентифікація, сховище, база даних реально часу, аналітика), також надає можливість налаштування з’єднання з Firebase проектом.

Для комфортної роботи з SDK що не є типізованим – використаємо AngularFire що є пакетом який надає типізацію Angular SDK та робить його придатним до роботи в Angular.

AngularFire згладжує грубі краї, з якими може зіткнутися розробник Angular під час впровадження Firebase JS SDK, , і прагне забезпечити більш комфортний та вірний досвід розробника в рамках Angular.

AngularFire має наступні плюси:

– Впровадження залежностей (DI) – надає можливість впроваджувати angular сервіси, для рооботи з firebase інфраструктурою, в компоненти;

– Observable орієнтованість– підтримка RxJS технології;

– Дружній до NgRx API – інтеграція з технологією NGRX через впровадження власний дій (actions) в AngularFire;

– Lazy loading – AngularFire динамічно імпортує велику частину Firebase відносного коду, скорочуючи час на завантаження програми;

– Google Analytics – надання інтерфейсу до впровадження аналітики в проект у Google Analytics;

– Router Guards – єффективний захист шляхів веб системи Angular за допомогою вбудованих перевірок автентифікації від Firebase;

2.4.5 Tailwind CSS Framework

TailwindCSS – це CSS-бібліотека, яка спрощує стилізацію HTML, тим самим шляхом, як це робить Bootstrap – додаючи величезну кількість різноманітних класів. Але, на відміну від Bootstrap, який додає вже готові до вживання компоненти, такі як кнопки, алерти та навбари, класи TailwindCSS націлені на конкретну властивість. У TailwindCSS немає заздалегіть написаних компонентів.

Плюси що надає даний фреймворк:

– Позбавляє розробника зобовязання декларувати імена класів або ідентифікаторв CSS;

– Мінімальна кількість рядків коду у файлі CSS;

– Можливість налаштувати дизайн для виготовлення компонентів;

- Робить веб-сайт адаптивним;
- Вносить зміни у потрібний спосіб.

Таким чином Tailwind допомагає навіть відмовитись від css фійлів проекту що дасть дуже гарний ефект для ростучих або великих проектів з огляду на швідкість роботи побудованої системи та її розмір (вагу).

3 СТРУКТУРА ДАНИХ СУТНОСТЕЙ У СИСТЕМІ

Одні з фундаментальних даних що потребується визначити та отримати – це дані по сутностям з яких складається структура університету, та їх специфічні значення. Мається на увазі те, що якщо детермінувати університет на ієрархічну структуру, то можна виділити ключові сутності з яких університет складається. Наприклад, припустимо, що існує сутність «університет» що є єдиною і головною сутністю в системі, в межах якої і здійснюються всі операції. Далі від університету можна виділити таке поняття як «факультет», в свою чергу факультет має ще вкладені сутності, наприклад «напрям», і т.д нижче мі зможемо дійти до сутності «група» яка є остаточною в структурній архітектурі приналежності сутностей університету. В межах цієї сутності, групи, розроблена система і буде виконувати більшість операцій.

Сформувавши приблизну структуру даних яка прийме роль фундаменту системи, побудуємо схему остаточного її вигляду що зображено на ієрархічній схемі.

Рисунок 3.1 – Схема ієрархічної структури сутностей університету

За допомогою парсингу cist.nure даних та його публічних API які методів є можливість отримати всі дані згідно намальованої схеми на рисунку 4. Методи cist API видають лише публічну інформацію згідно

університету. Зробивши запит до API методу /P_API_GROUP_JSON ми отримаємо потрібні сутності які які зможемо перетворити згідно до потреб.

Використовуючи отримані дані проведемо їх структурне перетворення яке задовільнить потреби системи.

В результаті приведення даних до комфортного виду з взяттям до уваги того моменту що дані будуть зберігатися та будуть використані за допомоги NoSQL базі даних Firebase.

Сутність Університет – є об’єктом що включає в себе наступні дані, такі як: short_name та full_name які є репрезентацією ім’я університету; faculties – є масивом об’єктів «факультет», включає всі факультети університету в масиві.

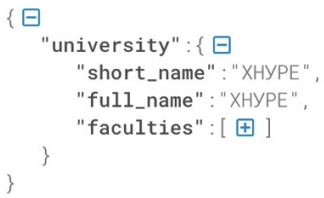


Рисунок 3.2 – Структура об’єкту university в системі

Сутність «Факультет» - є об’єктом що репрезентує факультет університету ХНУРЕ. Включає в себе наступні властивості: id – ідентифікаційний номер факультету; short_name – скорочене ім’я факультету (абревіатура); full_name – повне ім’я факультету; directions – масив об’єктів напрямів факультету.

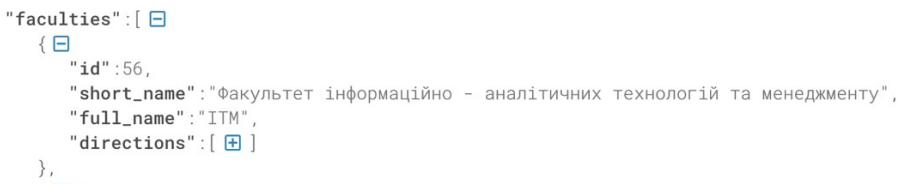


Рисунок 3.3 – Структура об’єкту faculty в системі

Сутність «Напрям» - є об’єктом що репрезентує напрям факультету університету ХНУРЕ. Включає в себе наступні властивості: id –

ідентифікаційний номер напряму; short_name – скорочене ім’я напряму (абривіатура); full_name – повне ім’я напряму; specialities – масив об’єктів напрямів факультету; groups – масив груп що безпосередньо відносяться до напряму і не відносяться до жодного з specialities цього напряму.

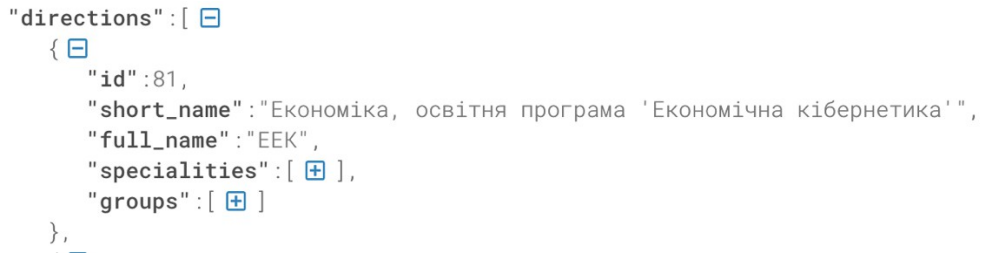


Рисунок 3.4 – Структура об’єкту direction в системі

Сутність «Спеціальність» - є об’єктом що репрезентує спеціальність специфічного напряму окремого факультету університету ХНУРЕ. Включає в себе наступні властивості: id – ідентифікаційний номер спеціальності; short_name – скорочене ім’я спеціальності (абревіатура); full_name – повне ім’я спеціальності; groups – масив груп що безпосередньо відносяться до спеціальності.

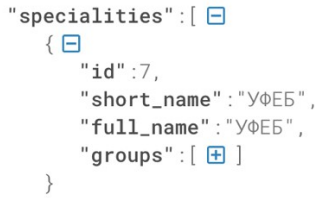


Рисунок 3.5 – Структура об’єкту speciality в системі

Сутність «Група» - є об’єктом що репрезентує групу окремої напряму спеціальності окремого напряму окремого факультету університету ХНУРЕ, або напряму окремого напряму окремого факультету ХУНРЕ. Адже за структурою ХНУРЕ група може не відноситись до будь якої спеціальності окремо взятого напряму. Включає в себе наступні властивості: id – ідентифікаційний номер групи; name – скорочене ім’я групи (абревіатура).

```
"groups":[
  {
    "id":8511182,
    "name":"УФЕБм-20-1"
  },
```


Рисунок 3.6 – Структура об’єкту group в системі.

3.2 Інші сутності системи

3.2.1 Сутність користувача в системі

Сутність користувача в системі я даними авторизаційними, якими наділяється поточний користувач. Сутність користувача по при те що зберігається в системі автентифікації Firebase також дуплікується в базу даних як основний об’єкт користувача, такий запис є розширенням сутності користувача адже Firebase автентифікаційна система не на дає великого спектру можливостей щодо збереження додаткових даних.

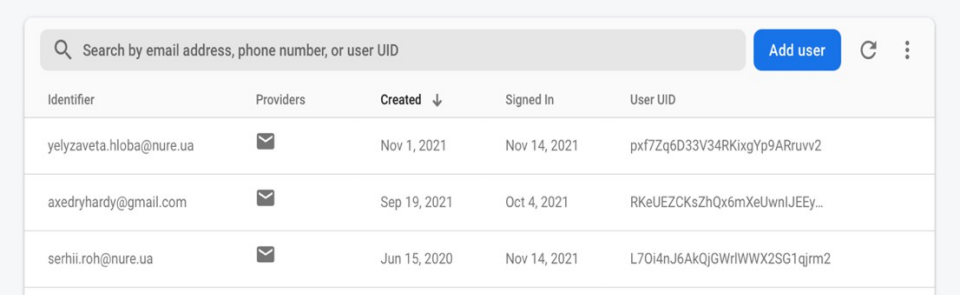


Рисунок 3.7 – Сутність користувача в автентифікаційній системі Firebase

На рисунку 8 продемонстровано сутності користувачів які записані в автентифікаційну систему Firebase. Ці сутності мають лише наступні дані: електронна пошта, унікальний ідентифікатор користувача, що був створений в момент автентифікації користувача в системі вперше.

В свою чергу розширена сутність користувача буде мати всі дані згідно його зв’язків з сутностями в системі, роль, та інші властивості.

Сутність розширеного користувача має наступні властивості:

– email – електронна пошта користувача, пошта за допомогою котрої користувач автентифікований;

– uid – це унікальний айді користувача який є продубльованим унікальним ідентифікатором з сутності користувача в автентифікаційній системі;

– role – роль користувача в системі. Роль надає права доступу до певних секій або функціоналу системи;

– first_name та last_name – ім’я та фамілія користувача відповідно;

– is_approved_account – ця властивість дає змогу визначити чи пройшов користувач етап підтвердження себе адміністрацією сайту, властивість надає змогу визначити чи має користувач доступ до системи чи ні;

– rooms – ця властивість користувача показує масив ідентифікаторів чат кімнат до яких відноситься користувач;

– photo_url – в цій властивості зберігається посилання на завантажену фотографію користувача;

– faculty_id, direction_id, speciality_id, group_id – ці властивості відображають структурну принадлежність користувача до університету.

Рисунок 3.8 – Зображення розширеної сутності користувача та його властивостей

3.2.2 Сутності кімнати для обміну даними користувачів

Для опису кімнати спілкування користувачів, також визначено структуру сутності яка виглядає наступним чином:

id – унікальний ідентифікатор кімнати що дозволяє отримати доступ або побудувати зв’язки с окремою сутністю кімнати;

room_details – сутність що включає в себе метадані кімнати до яких відносяться наступні властивості: amin_ids – масив ідентифікаторів користувачів що мають яким надаються права адміністрування кімнати та її контентом; direction_id, speciality_id, group_id – ідентифікатори сутностей університету, що слідує ієрархії вкладеності, до яких відноситься створена кімната; name – ім’я кімнати; room_image – посилання на зображення заставки кімнати.

users – масив ідентифікаторів юзерів що є членами кімнати.

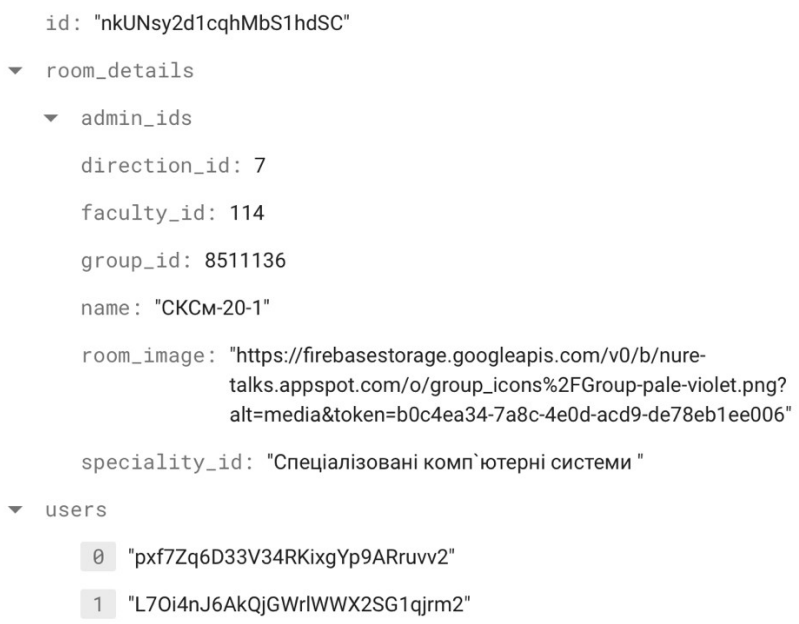


Рисунок 3.9 – Зображення структури та властивостей сутності кімнати в системі

Кожна кімната має масив id користувачів, в свою чергу користувачі мають масив id кімнат до яких вони приналежні, таким чином будується зв’язок «багато к багатьом». Також варто зазначити що на рівні сутності кімнати також зберігається колекція повідомлень цієї кімнати.

Кожна кімната в системі має відповідну колекцію повідомлень. В свою чергу сутність повідомлення спроектована наступним чином, там має такі властивості:

id – унікальний ідентифікатор повідомлення що генерується під час створення та відправлення повідомлення;

lastOperationTime – дата та час останньої операції над повідомленням, що включає в себе зміну тексту повідомлення або його відправлення;

room_id – ідентифікатор кімнати в рамках якої було зроблено повідомлення;

sender_id – ідентифікатор користувача який створив повідомлення;

text – текст повідомлення;

type – тип повідомлення. В системі існує такі типи повідомлень: regular та attachments. Ця властивість дає змогу системі зрозуміти що з себе представляє об’єкт повідомлення, чи це звичайне повідомлення, чи це

повідомлення з додатками файлів. Такий тип дає змогу розширення поняття повідомлення в системи, додаванням нового типу;

time - час створення повідомлення в системі.

attachments – масив об’єктів що репрезентують метадані по файлів що були додані до повідомлення. Об’єкт типу attachment має властивості: file_path: шлях до файлу для скачування; id – ідентифікатор файлу додатку; name: ім’я файлу;

Рисунок 3.10 – Сутність повідомлення в системи типу regular та її властивості.

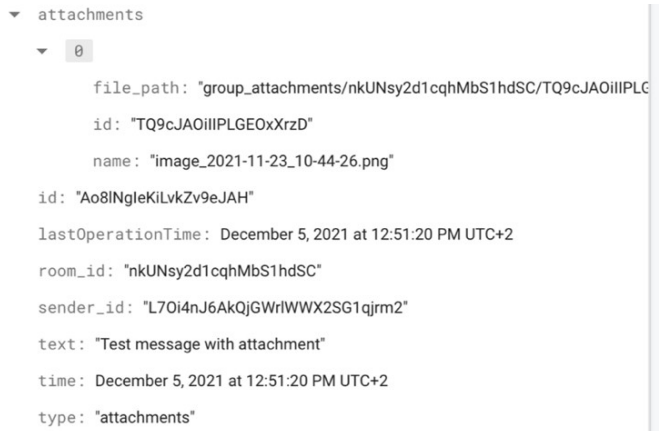


Рисунок 3.11 - Сутність повідомлення в системи типу attachments та її властивості.

3.3 Схема зв’язків сутностей системи

Згідно з розробленою структурою для системи, що включає залежності сутностей, їх опис властивостей, можна побудувати схему залежностей всіх домен сутностей у системі один від одного.

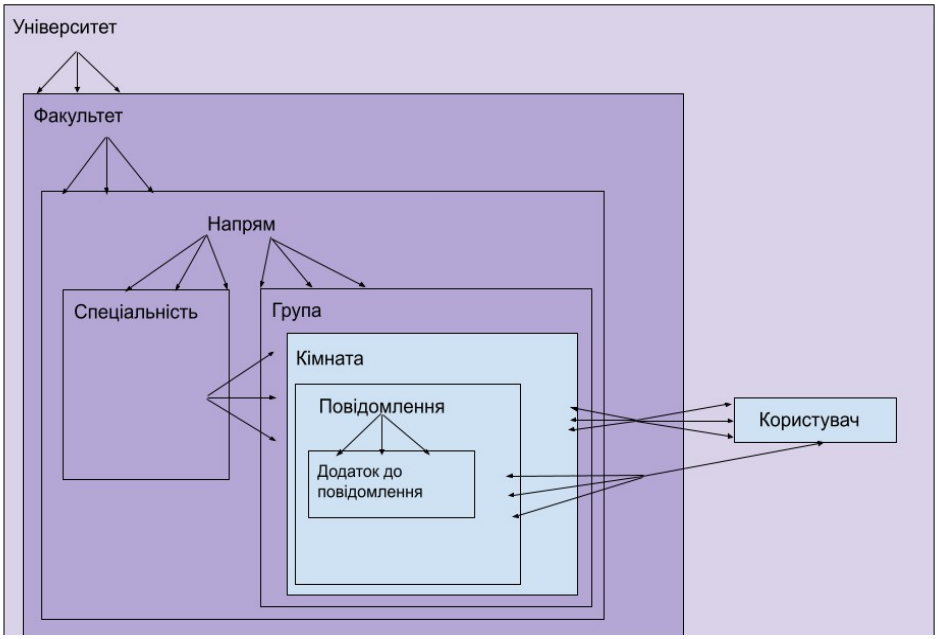


Рисунок 3.12 – Схема взаємозв’язків домен сутностей системи.

На схемі, що зображена на рисунку 3.12, можна виділити наступні елементи:

Рожевий та фіолетовий кольори на схемі зображують блоки сутностей що відносяться до структури сутностей університету. Рожевим зображена корінна сутність «університет» в рамках якої всі інші сутності існують.

– Рожевий та фіолетовий кольори на схемі - зображують блоки сутностей що відносяться до структури сутностей університету. Рожевим зображена корінна сутність «університет» в рамках якої всі інші сутності існують.

– Блакитний колір – зображую сутності що не мають відношення до структури університету але побудовані додатково, та існують в рамках університету та його сутностей.

Зображення залежностей відбувається за допомогою стрілок, де одинарна стрілка зображую залежність одинарну, а потрійна стрілка зображає множинну залежність.

4. РОЗРОБКА ТА ОРГАНІЗАЦІЯ ЧАСТИН СИСТЕМИ

Згідно з попередніми теоретичними розробками та визначенням сутностей системи та їх шлях взаємодій один між одним, можна приступити до розробки функціоналу системи.

4.1 Схематизація роботи функціональних частин системи

Почати слід з візуалізації процесів роботи логічних частин системи один між іншим, тобто почати з побудови схем роботи системи на основі котрих буде реалізовано програмний код системи.

Щоб виконати поставлене завдання, потрібно продумати структуру системи, для того, щоб розбити загальне завдання на більш менші частини, поєднавши які, отримаємо повну та робочу систему. Структура системи – спосіб організації зв’язків між елементами (підсистемами). Під час реалізації структури, надається опис множини елементів і зв’язків між цими елементами, розподілення завдань по рівням і частинами системи, які забезпечують їх рішення.

4.1.1 Високорівнева схема роботи між рівнями системи

Високорівнева схема потрібна для того щоб зрозуміти між якими рівнями та як відбувається взаємодія певних рівней системи, а саме, клієнтської частини між серверною частиною. Також це дає змогу зрозуміти як спілкуються ці елементи між собою. Схема зображена на рисунку 4.1

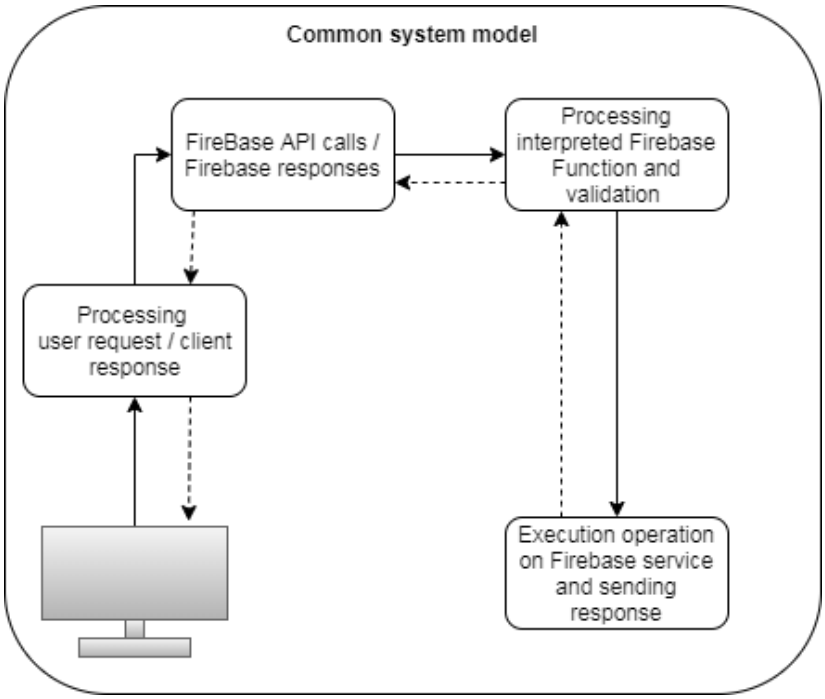


Рисунок 4.1 – Зображення схеми взаємодії елементів системи на високому рівні.

Побудована схема дає змогу побачити що система складається з клієнтської частини, прикладної частини та серверної частини.

Частина клієнтська - відповідає за обробку запитів користувача до серверної частини, відображає візуальний інтерфейс користувача, який дає змогу відобразити стан в якому перебуває система в контексті користувача. Представляє первинну валідацію запитів зі сторони. Користувача до сервера. Ця частина зображено на блок-схемі як «Processing user request / client response».

Прикладна частина є відповідальною за процес надсилання запитів до Firebase сервісів. Дана частина є пов’язуючою частиною між клієнтською і серверною частинами системи. Ця частина є прикладним інтерфейсом що надається SDK Firebase та виступає в клієнтській частині програмним пакетом angular/fire та firebase. SDK надає можливість налагодити зв’язок між клієнтською частиною та серверною, що в результаті забезпечує можливість обміну даними між цими частинами, виконання запитів та отримання результатів на запит для обох частин. На рисунку ця частина зображена блоками “Firebase api calls / Firebase response” – api requests. Та

“Processing interpreted Firebase functions an validations”, що зображує запит до серверного API на клієнтській частині, в Firebase функції, що є компільованими функціями запиту який Firebase зможу зрозуміти.

Серверна частина системи, що зображена на схемі функціональним блоком з підписом “Execution operation on Firebase Service and sending response” означає, що вже безпосередньо код запиту виповнює процедуру виконання та надсилає (не завжди) відповідь на клієнтську частину.

4.1.2 Схема функціонування клієнтської частини системи.

На рисунку 4.2 зображено взаємодію між ключовими функціональними можливостями на вищому рівні, тобто кожен функціональний блок приховує за собою більш детальну функціональність.

Першим стартовим елементом описано запит користувача до системи.

Кожен запит користувача проходить через спроектований механізм «Автентифікаційний перехоплювач запитів» або «Auth interceptor». Завданням перехоплювача запитів є: перевірити чи клієнт-браузер вже має дані користувача; чи підтверджений його запис в системі який відображається властивістю is_approved.

Якщо перевірка перехоплювачем показує дійсний результат – вважається що користувача автентифіковано до системи, та надаємо можливість на виконання запиту що він виконав.

Будь які маніпуляції з даними та всі запити відбуваються у контексті обраної кімнати. В тому випадку якщо під час наступних запитів перевірка повертає незадовільний результат то користувача перенаправляє на перший крок – «Authentication and authorization» зображено на схемі.

Елемент “Conversation room selection” – описує функціональну частину з обранням кімнати бесіди користувачем після успішної автентифікації.

Елемент “Messaging within selected room context” - відображає алгоритм листування та обміну даними в ораній користувачем кімнаті (в поточній кімнаті), обробку та відображення даних повідомлень тощо.

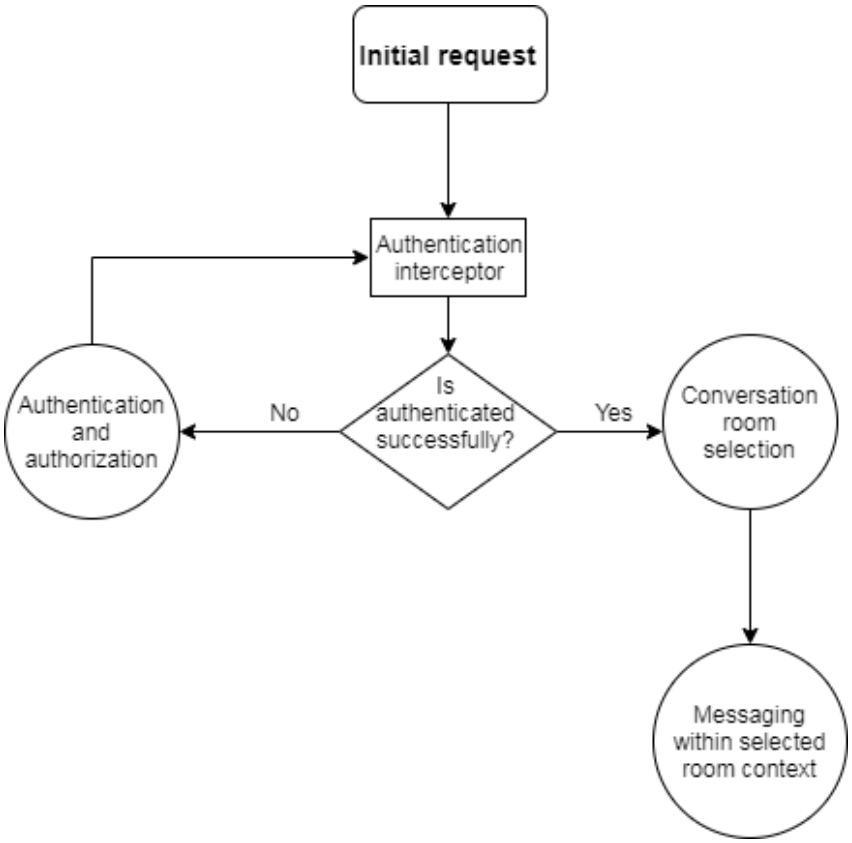


Рисунок 4.2 – Схема ключових функціональних елементів системи

4.1.3 Схема автентифікації та авторизації

Згідно з теоретичних розробок що були проведені у розділі 1, було приняте рішення в реалізації спрощеного процесу входу до системи користувача, де будь який член життя університету, будь то студент чи викладач який має @nure.ua домен пошту, вже є автентифікованим, залишиться лише одна, пройти авторизацію та заповнити метадані для розуміння системою яких прав треба наділити користувача. Тому вхуд до системи не потребує ніяких даних таких як паролі, чи ім’я. Потрібна лише пошта @nure.ua за допомогою якої користувач і зможе підтвердити свою автентифікацію перейшовши по посиланню з отриманого повідомлення.

Після проходження по посиланню, користувача повернуто назад до системи, та зберігає дані про користувача з Firebase браузері клієнта, таким чином користувача автентифіковано, одразу після цього, механізми системи на клієнтській стороні перевіряють чи авторизовано користувача, якщо

користувача не було авторизовано, а тобто в записі про нього в Firebase ще нема даних згідно його ролі в системі та приналежності в структурі університету, то користувачеві відображається наступна форма де він повинен орати свою роль та приналежність. Після відправлення такої авторизаційної форми, користувача чекає перевірка адміністрації, така перевірка допоможе запобігти небажаних дій зі сторони користувача, що може нанести шкоду користувачам. Одна з таких небажаних дій – це обрання неспівпадаючої ролі користувача, наприклад студент обирає роль працівника університету, що надає йому несанкціоновані привілеї в доступі. Таким чином перевірка адміністрації є бажаною в такій системі.

В даній реалізації системі, функціонал для адміністраторів не реалізовується, адже це інше завдання що не входить до основного завдання роботи. Таким чином процес перевірки адміністрації, на даний момент, пропускається, тож користувач вважається підтвердженим одразу після авторизації.

Коли користувач отримує підтвердженні – він отримує доступ до системи, до веб інтерфейсу та функціональних можливостей згідно своєї ролі та інших властивостей користувача.

Весь процес та нюанси такої автентифікації та авторизації зображено нижче, на рисунку 4.3.

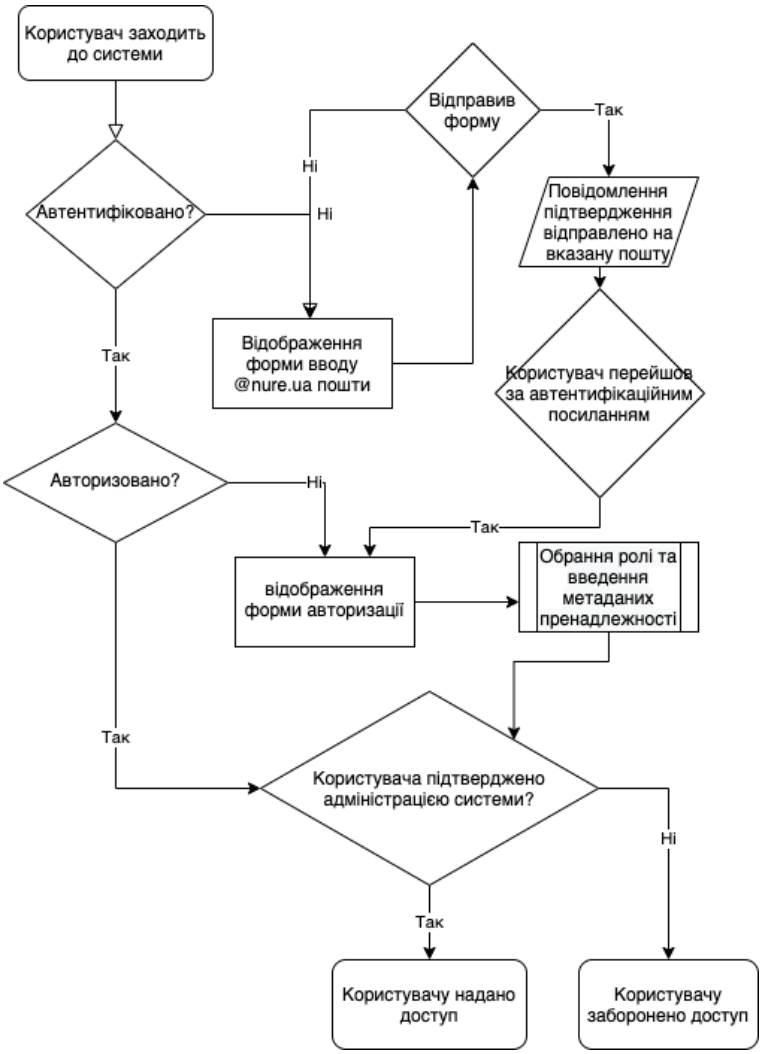


Рисунок 4.3 – Схема процесу автентифікації та отримання доступу в користуванні системою.

4.1.4 Схема доступу користувача до кімнат груп

Після успішної авторизації та проходження підтвердження користувача в системі адміністрацією, користувачу надається доступ до системи, де на веб інтерфейсі йому буде зображено перелік кімнат що існують в системі та ті до яких він має доступ, при виборі кімнати користувачу буде відображено секцію з повідомленнями та формою їх відправлення.

Кожна роль має свої привілеї згідно доступу до кімнат. Наприклад, користувач з роллю «студент» одразу після проходження перевірки акаунту адміністрацією системи, отримує доступ до кімнати групи в якій студент навчається. Що-до доступу користувачам з роллю «працівник в

університеті», користувач буде мати можливість увійти в кожну кімнату групи в системі.

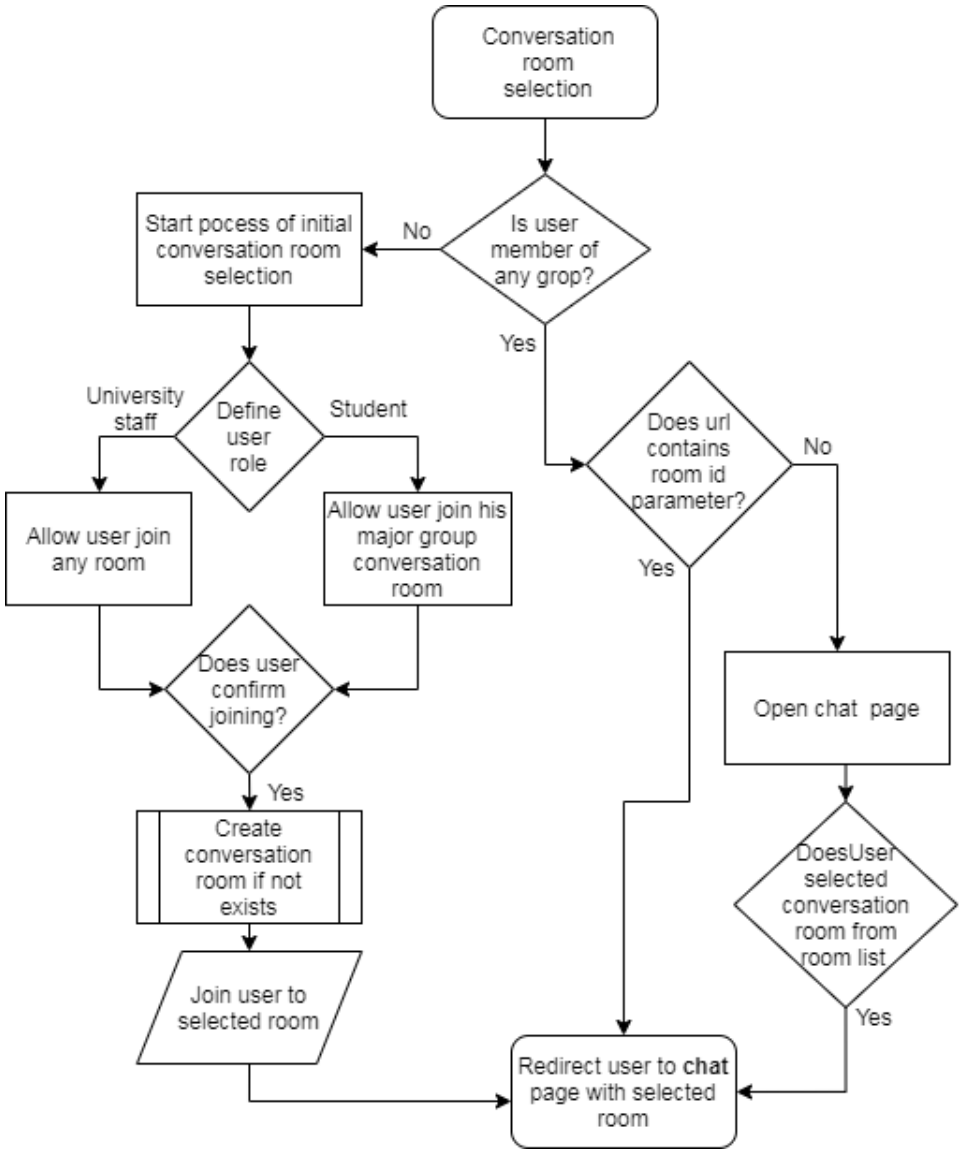


Рисунок 4.4 – Схема алгоритму обрання кімнати користувачем

4.2 Організація інфраструктури та зв'язків сервісів Firebase

Згідно до теоретичних розробок, проведемо інтеграцію сервісів Firebase за допомогою SDK. У програмну частину системи. Організуємо структури даних в базі даних Firebase згідно завдання до організації структури даних в системі, вирішимо задачі нюанси при роботі над організацією структур з базою даних Firebase. Налаштуємо автентифікаційну систему, поєднаємо

користувача з його розширеним записом в базі даних. Налаштуємо сховище великих об’єктів.

4.2.1 Організація зв’язку серверної частини з клієнтським кодом.

Перш ніж додати Firebase до Angular Framework проекту, потрібно створити проект Firebase і зареєструвати тип проекту в цьому проекті, в моєму випадку це Web. Коли додаток реєструється у Firebase, можна отримати об’єкт конфігурації Firebase, який використовується далі для з’єднання програми з ресурсами проекту Firebase.

Далі зображено формат конфігураційних даних що треба вказати в програмному проекті Angular.

```
// For Firebase JavaScript SDK v7.20.0 and later, `measurementId` is an optional field
var firebaseConfig = {
  apiKey: "API_KEY",
  authDomain: "PROJECT_ID.firebaseapp.com",
  databaseURL: "https://PROJECT_ID.firebaseio.com",
  projectId: "PROJECT_ID",
  storageBucket: "PROJECT_ID.appspot.com",
  messagingSenderId: "SENDER_ID",
  appId: "APP_ID",
  measurementId: "G-MEASUREMENT_ID",
};
```

Рисунок 4.5 – Шаблон конфігураційних даних для залучення Firebase проекту в Angular проект

Такий шаблон має метадані що є конфіденційної інформацією та повинні бути захищені від злочинних рук. Ці дані надаються Firebase та отримати їх можна в пару кліків.

Для внесення цих даних в Angular проект використовуємо @angular/fire SDK пакет.

Рисунок 4.6 – Ініціалізація зв’язку Angular проекту з проектом Firebase

Імпортуємо модулі AngularFireStorageModule та AngularFireModule. За допомогою другого, використовуємо його метод initializeApp та надаємо об’єкт конфігурації першим аргументом. Також вписуємо в imports масив AngularFireStorageModule щоб отримати доступ до сховища об’єктів. Таким чином ми отримуємо доступ за допомогою програмного інтерфейсу до великої кількості наданих сервісів Firebase, це такі як: AngularFirestore; AngularStorage; Authentication system та багато інших.

4.2.2 Налаштування Автентифікаційної системи

Налаштування системи автентифікації Firebase не є складним процесом дякуючи комфортному веб інтерфейсу системи в декілька кліків можна обрати способи автентифікації та поміняти шаблони повідомлень що надсилаються на пошту користувачу під час автентифікації.

Згідно з завданням, в системі повинен використовуватись спрощений процес входу в систему, тобто за допомогою повідомлення та посилання в ньому користувачеві. Такий підхід реалізовується за допомогою першого пункту наданих провайдерів автентифікації Firebase, обираємо лише його.

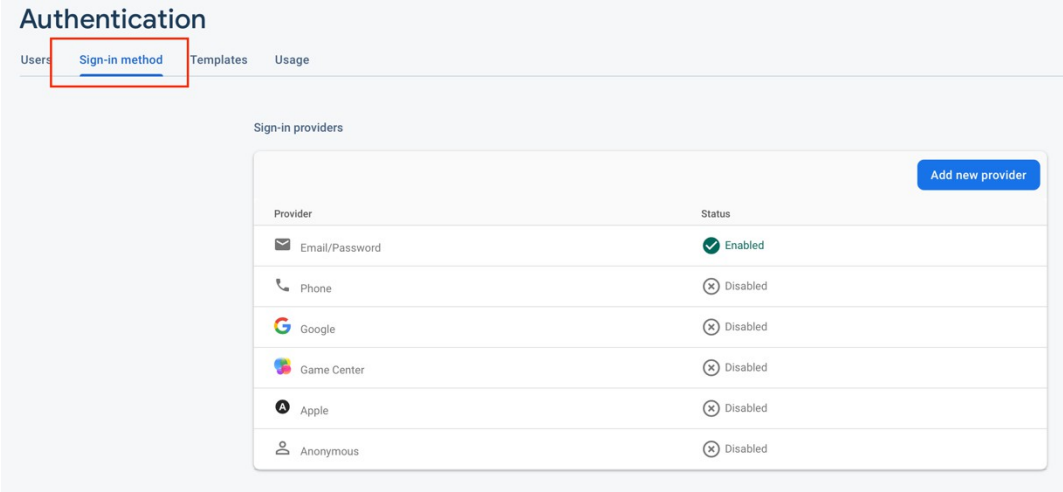


Рисунок 4.7 – Обрання способу автентифікації

4.2.3 Організація сховища великих об’єктів

Для інтеграції сховища об’єктів потрібно внести дані до властивості конфігурації firabase в Angular проекті, ця властивість називається storageBucket що зображена на рисунку 4.8.

Після проходження покрокової інструкції того як ініціалізувати сховище, перейдемо до огранізації структури даних в ньому.

Всі файли що будуть додані в кімнатах разом з повідомленнями, будемо зберігати по шляху /group_attachments/*room_id*.

Іконки заставки груп будуть зберігатись по шляху /group_icons.

Іконки профілю користувачів будуть зберігатись по шляху /profiles_images.

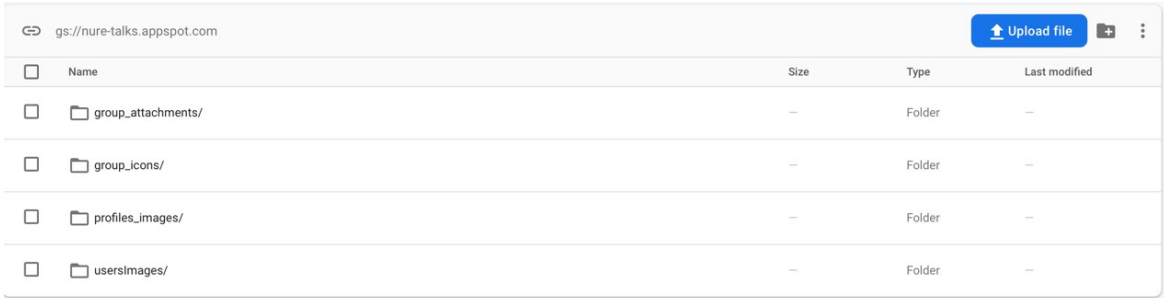


Рисунок 4.8 – Організація секцій збереження контенту в сховищі великих даних Fire Storage

4.2.4 Організація даних в Firestore базі даних

Згідно структурного рисунку 3.12, на якому вказано залежність та приналежність сутностей системи, реалізую ідентичну залежність сутностей в базі даних Firestore.

Структура університету буде зберігатись у колекції названій faculties. Та має наступний вигляд що зображено на рисунку 4.9.

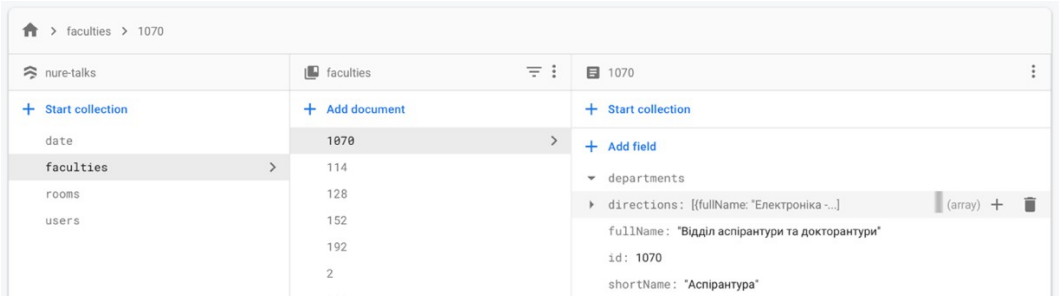


Рисунок 4.9 – Зображення сутностей університету в колекції faculties

Кожен вкладений документ в колекцію, має спеціальний ідентифікатор, для того щоб можна було побудувати лінейний маршрут до об'єкта.

Наприклад сутність факультету дублює свій id в якості головного ключа, тож цей ключ є індексом об’єкту, що ми можемо спостерігати. В другій колонці faculties на рисунку 4.9.

Користувачі зберігаютсья в колекції users. Первинними ключами виступає продубльований унікальний айді із системи автентифікації, таким чином отримуємо зв'язок між розширеною сутністю та автентифікаційною сутністю що дає змогу відстежувати автентифікаційні дії користувача.

Можна побачити що попри те що унікальний айду дублюється в властивість uid, та сам документ має такий самий ідентифікатор.

Подібна маніпуляція пророблена для всіх розроблених структурних приналежностей сутностей в Firestore.

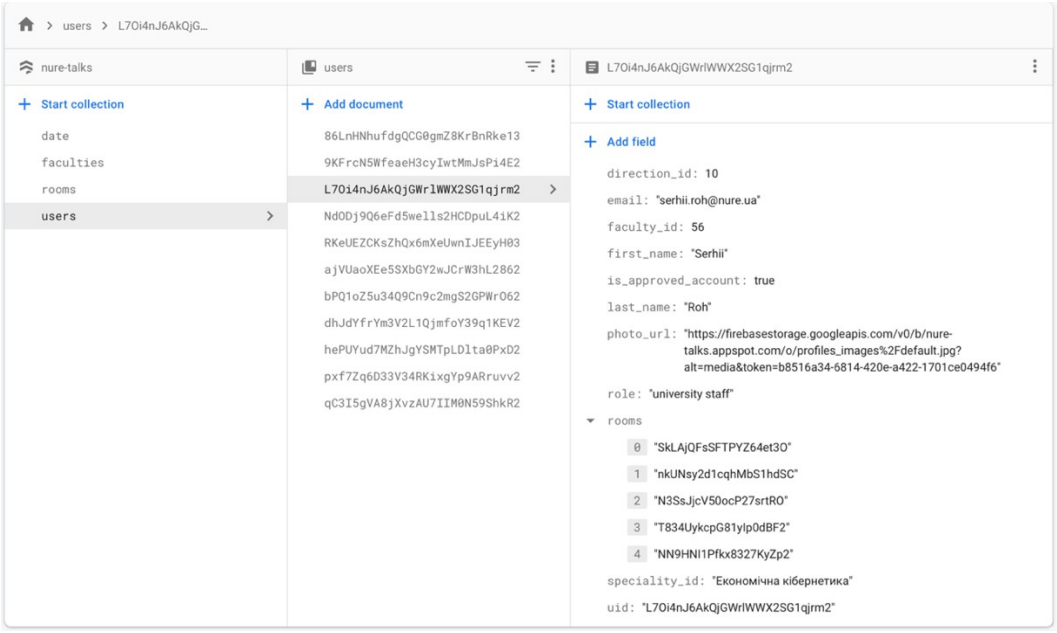


Рисунок 4.10 - Зображення сутностей користувачів в колекції users

При реалізації структури для кімнат бесід, було вирішено створити більшу зложеність сутностей, тож тепер на рівні об’єкту кімнати створюється нова колекція що називається messages. Принцип побудови запиту до об'єкту повідомленні залишається той самий але трохи складнішим з огляду на глубину запиту. Дивлячись на рисунок 4.11, запит до повідомлення буде виглядати наступним чином: rooms/*room_id*/message/*message_id.

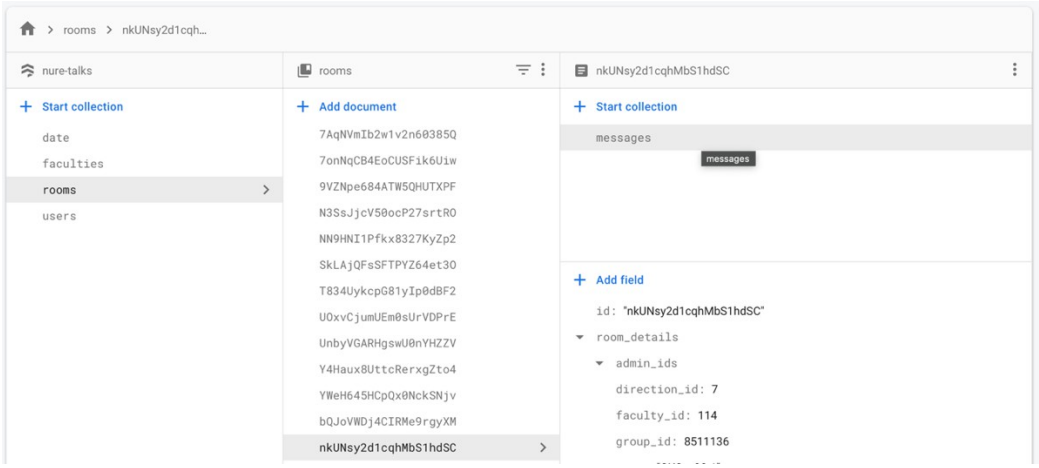


Рисунок 4.11 - Зображення сутностей кімнат в колекції rooms та коллекції messages

4.2.5 Налаштування доступу до скачування файлів за сховища

Найпростіший спосіб налаштувати CORS правила в Firebase Storage– це інструмент командного рядка gsutil. Інструкції з встановлення gsutil. Після того, як встановили gsutil та пройшли автентифікацію за допомогою нього, можемо використовувати його для налаштування CORS.

Наприклад, для рішення нашого завдання достатньо задати інструкцію що дозволить скачування об'єктів з сховища, записати інструкцію треба в файлі cors.json. Правки будуть мати наступний вид, що зображено на рисунку прикладу 4.12.

```
[
  {
    "origin": ["https://example.com"],
    "method": ["GET"],
    "maxAgeSeconds": 3600
  }
]
```

Рисунок 4.12 – Зображення зміненого файлу cors.json

Далі виконуємо команду в консолі gsuite: gsutil cors set cors.json gs://nure-talks.appspot.com.

Після цього, нові правила задіяні, користувачу тепер надано доступ до скачування файлів по прямому посиланню.

4.3 Розробка інтерфейсу користувача системи

Перед програмною розробкою та стилізацією системи, варто провести розробку візуальної частини системи, тієї частини, що є інтерактивною для користувача, що дає змогу користувачеві взаємодіяти з елементами системи, іншими словами – веб інтерфейс користувача. При розробці даної частини системи, буду спиратися но власний опит як користувача подібних комунікаційних систем, для реалізації інтуїтивно зрозумілих елементів інтерфейсу користувача, використовуючи спокійні кольори та відомі візуальні рішення щодо виділення елементів системи як інтерактивних.

Враховуючи те, що система є комунікаційною, а основними елементами є сутності університету, користувач, кімната бесіда, то варто виділити місце в дизайні під відображення списку кімнат користувачеві, елементи котрого – це репрезентація кімнат в системі. Секція контенту, котра повинна відображати повідомлення обраної кімнати, що буде включати форму вводу повідомлення та відображення повідомлень та їх метаданих. Додаткова секція на якій розташується секція користувача, налаштування, автентифікація, тощо.

Таким чином можна розробити наступний шаблон який буде задовольняти потреби описані вище, також шаблон зображає приблизну палітру кольорів що будуть використовуватись в побудові дизайну інтерфейсу користувача.

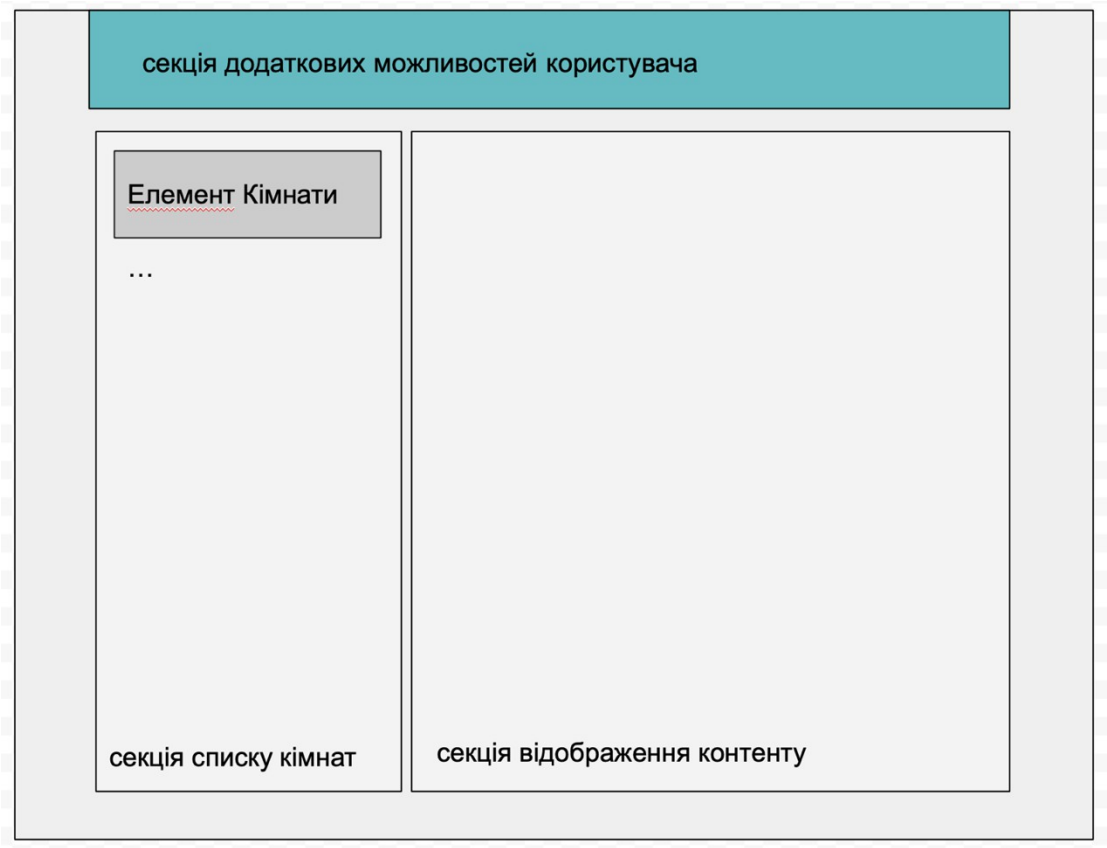


Рисунок 4.13 – Зображення шаблону розробленого інтерфейсу користувача

Почнемо з початкового елементу інтерфейсу що користувач побачить в перше, та згідно дизайн системі, що включає кольорову палітру інтерфейсу, побудуємо початкові компоненти дизайну.

Форма входу повинна буду простою, відділеною від зображення загального контенту до тих пір поки користувач не увійду успішно в систему.

Тому зобразимо користувачу просту форму з описом поля вводу та описанням функції форми, та кнопкою логіну, даний дизайн зображено на рисунку 4.13.

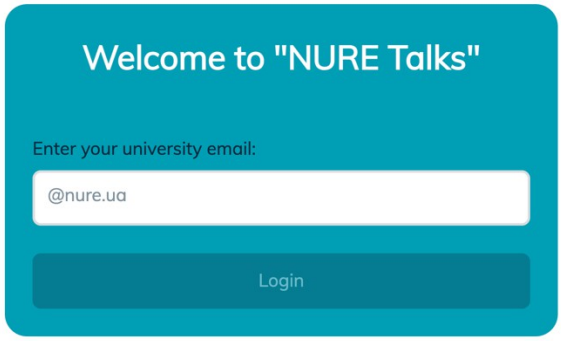


Рисунок 4.14 – Зображення початкової форми авторизації

Після того як користувач введе дані, згідно з завданням, йому буде надіслано автентифікаційне повідомлення на пошту, тобто він ще не є авторизованим в системи, тому поки користувач не перейде по посиланню авторизації, не відображаємо йому основний контент, відобразимо дизайн заглушку з підказкою його очікуваних дій, таку заглушку зображено на рисунку 4.14.

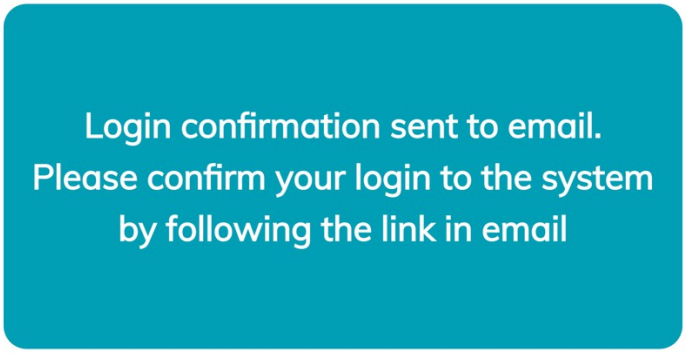


Рисунок 4.15 – Інформативна заглушка при очікуванні завершення авторизації користувача

Після успішної авторизації користувачем, потрибно надати можливість користувачеві заповнити дані згідно принадлежності себе до університету. Для цього розроблено дизайн форми другої фази атворизації яка віглядає наступним чином. така форма має випадаючі списки, кнопки «радіо кнопка».

Після заповнення такої форми користувача перенаправляємо до головного інтерфейсу системи який має наступний вигляд (рисунок 4.15).

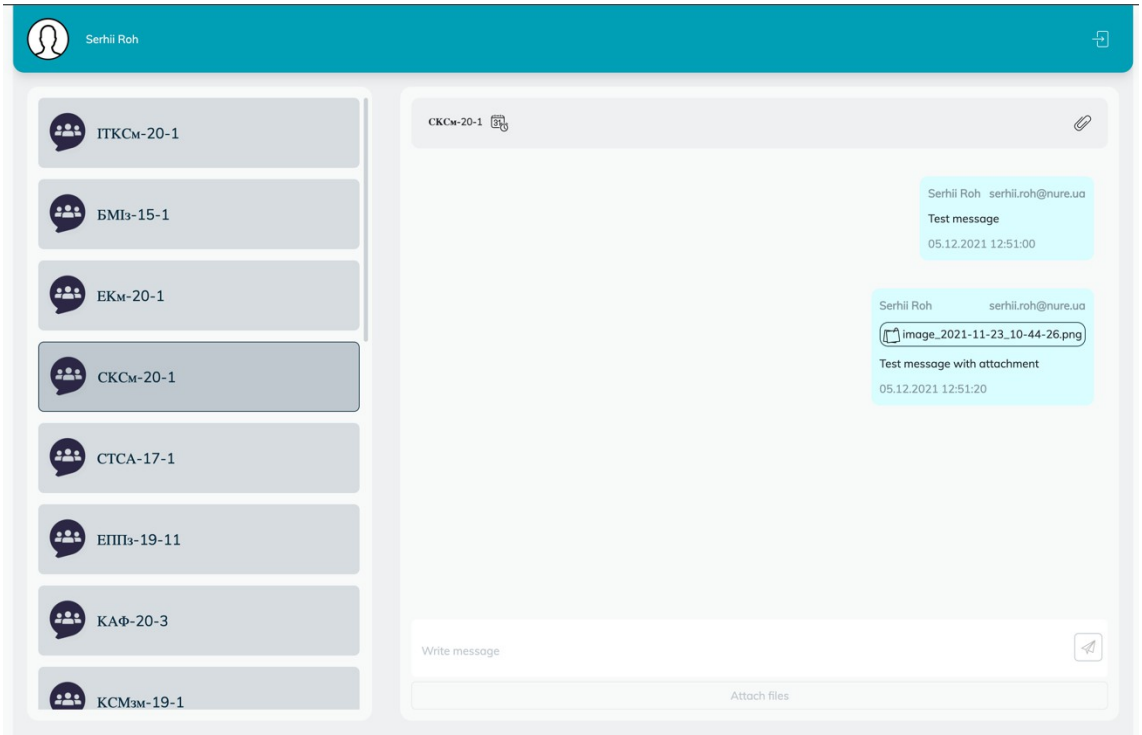


Рисунок 4.16 – Зображення головного інтерфейсу та елементів його управління, контенту

З огляду на результат розробки дизайну, можна відмітити те, що, вдалося реалізувати інтерфейс згідно до базового концепту та кольорової палітри.

В шапці проекту вдалося розмістити метадані по користувачу, а саме: іконку профілю; ім’я та фамілію. Також з правої сторони розмістилась інтерактивна іконка при взаємодії с якою, користувач може вийти з системи.

Елементи кімнат також отримали візуалізацію згідно з концептом. На елементі розміщені наступні метадані: іконка кімнати, назва кімнати.

Секція контенту, на рисунку демонструє чат кімнати, де знизу відображається поле вводу тексту, справа кнопка відправлення повідомлення, та ще нижче кнопка що дозволяє додати файл до повідомлення.

Також в основному середовищі контенту кімнати зображена «шапка» на якій розташувались назва кімнати, та іконка розкладу, взаємодія з якою повинна відкрити користувачеві розклад групи, якщо кімната відноситься до групи. Справа в тій самій «шапці» є іконка що зображує прикріплення файлу глобально до кімнати.

В секції відображення повідомлень зображено 2 повідомлення, різниця в тому що в другому повідомлені, що зображено нижче, зображено дизайн файлу що додано до того повідомлення. Взагалі, в повідомленні відображаються метадані користувача, дані самого повідомлення, та час відправлення повідомлення.

4.4 Програмування клієнтської частини системи

Побудування програмної частини системи буде вестись за умов схематичних розроблень що зображені на рисунках 4.1, 4.2, 4.3, та згідно дизайн системи на малюнку 4.16.

В цьому підрозділі будуть висвітлені розробки ключових елементів схем та дизайну, деякі функціональні можливості системи не будуть проілюстровані по причині другорядності функціоналу.

Абстрактний шаблон взаємодії елементів програмної частини системи виглядає наступним чином на рисунку 4.17.

Рисунок репрезентує розроблену концепцію взаємодії компонентів, сервісів, та елементів стейт менеджмент системи між собою. Головною ідеєю

є абстрагування компонентів і звичайних допоміжних сервісів від взаємодії з елементами стейт менеджмент системи. Цей підхід допоможе позбавитись небажаного розширення коду в компоненти Ангуляру зі сторони NGRX, підтримає компоненти в чистоті, та позбавить залежності від NGRX, також забезпечить кращий контекст для тестування компонентів, і варто звернути увагу на забезпечення більшої ясності коду для програмістів системи в майбутньому за допомогою такого підходу, що досягається реалізацією сервісів що реалізують паттерн «Фасад» до NGRX інфраструктури.

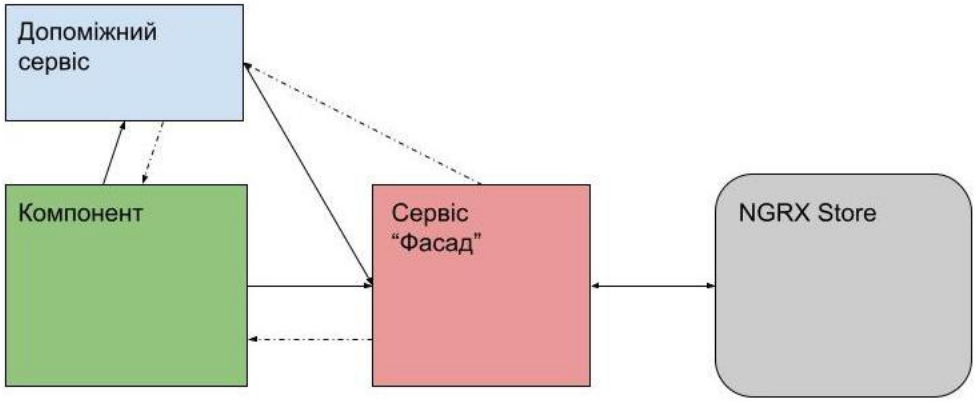


Рисунок 4.17 – Організація взаємодії елементів програмної частини системи

4.4.1 Автентифікація та автентифікаційний перехоплювач.

Усі компоненти та елементи системи що відносяться до візуальної частини обробки автентифікаційних запитів об’єднані в модуль модуль відображення AuthenticationModule. Цей модуль включає в себе компоненти EmailLoginComponent, та EmailCallbackComponent.

EmailLoginComponent – компонент що відповідає за відображення форми заповнення @nure.ua пошти, валідацію цієї форми, та ініціювання запиту до автентифікаційного сервісу передавши введені параметри в форму. Функція автентифікаційного сервісу в свою чергу викликає метод

sendSignInLinkToEmail сервісу FirebaseWrapper що є обгорткою на наданий сервіс SDK пакетом AngularFirebase, та записує в локальне сховище браузера пошту користувача для подальшої маніпуляції з нею.

Лістинг 4.1 – Відображення проміжного методу відправки повідомлення авторизації в системі на пошту користувача.

```
async sendSignInToEmail(email: string): Promise<void> {
  this.firebaseWrapper.sendSignInLinkToEmail(email);
  window.localStorage.setItem('emailForSignIn', email);
}
```

Далі, коли користувач перейде по посиланню авторизації, його буде перенаправлено до компонента EmailCallbackComponent, який виконує роль обробки такого запросу та виконує метод сервісу щодо авторизації в системі користувача. Лістинг коду виконуючої функції компоненту зображено на лістингу 4.2.

Лістинг 4.2 – Функція обробки направлення користувача до системи після переходу за посиланням в компоненті EmailCallbackComponent

```
private async trySignInWithEmailAsync(): Promise<boolean> {
  try {
    if (this.authService.isSignInWithEmailLink()) {
      var email = window.localStorage.getItem('emailForSignIn');
      const user = await this.authService.signInWithEmailLinkAsync(email);
      await this.userFacade.createUser(user.user.email);
      window.localStorage.removeItem('emailForSignIn');
      this.windowHelper.redirectToOrigin();
      return true;
    }
  } catch (err) {}

  return false;
}
```

Метод trySignInwithEmailAsync по перше перевіряє чи є запит до системи – запитом авторизації, це виконується кодом методу isSignInWithEmailLink сервісу authService. Якщо запит є запитом авторизації, з локального сховища зчитуєтсья записана пошта користувача та виконується

метод signInWithEmailLinkAsync що авторизує користувача успішно в системі, далі за другорядною логікою чиститься локальне сховище та користувача перенаправляє на по корінному шляху побудованої системи.

Якщо для користувача автентифікація в систему була вперше, його перенаправляє по шляху до компоненту ContinueProfileComponent, роль якого наступна: відобразити форму для користувача з обрання принадлежності та його ролі в системі; валідувати форму; сформувати дані для запиту до сервісу котрий виконає операції згідно з оновленням сутності користувача з цими даними.

Лістинг 4.3 – Зображення тіла методу обробки підтвердження вводу даних користувачем до форми в компоненті ContinueProfileComponent.

```
async sendUserPhaseTwoData(): Promise<void> {
  this.requestProceeding$.next(true);
  const universityStructureFormValues = this.selectUniversityStructureRef.form
    .value as UniversityStructureDynamicFormValuesMap;
  const rolesFormValues = this.formGroup.value as RolesFormValuesMap;

  const universityStructure: UniversityStructureByIds = {
    [UniversityEntitiesName.faculty]: universityStructureFormValues[UniversityEntitiesName.faculty].id,
    [UniversityEntitiesName.direction]: universityStructureFormValues[UniversityEntitiesName.direction].id,
    [UniversityEntitiesName.speciality]: universityStructureFormValues[UniversityEntitiesName.speciality].direction_id
      ? null
      : universityStructureFormValues[UniversityEntitiesName.speciality].fullName,
    [UniversityEntitiesName.group]: universityStructureFormValues[UniversityEntitiesName.group]?.id,
  };

  const role = RoleEnum[rolesFormValues.roles];
  await this.authService.sendCurrentUserSecondPhaseAuthData(universityStructure, role);
  this.router.navigate(['/']);
}
```

В методі sendUserPhaseTwoData, що зображено на лістингу 3, відбувається просте формування введених даних до форми, в об’єкт відповідного вигляду що є дійсним для передання в якості аргументу далі до методу sendCurrentUserSecndPhaseAuthData сервісу authService.

В свою чергу метод sendCurrentUserSecndPhaseAuthData абстрагує логіку виклику двох методів UserFacadeService’у, а саме changeUserRole та changeUserUniversityStructureAccessory, що зображено на лістингу 4.4.

Лістинг 4.4 – Зображення коду проміжного методу в сервісі AuthService з відправкою запиту до проходження другої фази авторизації

```
async sendCurrentUserSecondPhaseAuthData(
  universityStructure: UniversityStructureByIds,
  role: RoleEnum
): Promise<void> {
  const currentUserId = (await this.getCurrentUserAsync()).uid;
  try {
    await this.userFacade.changeUserRole(currentUserId, role);
    await this.userFacade.changeUserUniversityStructureAccessory(currentUserId, universityStructure);
  } catch (e) {
    // TODO
  }
}
```

Ці методи в UserFacadeService'і надсилають сутності Action від NGRX. В свою чергу на ці Actions є ефекти (Effects), що обробляють дії та формують запит до серверної частини. Код таких ефектів не є примітивним, а складним з огляду на ланцюг логіки та специфіку модифікацій що вони виконують.

Наприклад, в наведеному фрагменті коду ефекту, лістингу 5, комбіновано багато запитів та логічних операцій що реалізуються транзакційним способом.

В даному функціоналі додатково проходить перевірка надісланих даних форми згідно пренадлежності, з даними що лежать в базі даних. Таким чином запобігається будь яка спроба підмінити дані за допомогою такої валідації. Це реалізується в методі checkUniversityStructure.

Далі формується остаточне тіло запиту до бази в Firebase з додатковими метаданими та виконується запит щодо зміни даних користувача, за допомогою методу updateUser сервісу UserHttpService.

Після того як запит згідно зміни сутності користувача було виконано, виконується остання операція в ланцюгу операцій, а це операція добавлення користувача в кімнату бесіду обраної заздалегідь групи. Така операція виконується за допомогою методу joinUserToRoom сервісу UserFacadeService, але перед тим проходить перевірка чи існує вже відповідна до групи користувача кімната, якщо так то користувач додається, якщо не існує то виконується операція створення кімнати за допомогою методу ensureCreateGroupRoom сервісу RoomFacadeService.

Після цього, автентифікаційний та авторизаційний процеси користувачем вважаються повністю завершеними, тепер користувачу доступна головна сторінка, де він може обрати кімнату та перейти до чатування з коллегами.

Лістинг 4.5 – Зображення фрагменту коду ефекту на дію запиту відправки проходження другої фази авторизації

```
switchMap((user) => {
  return this.universityFacade.checkUniversityStructure(action.universityStructure).pipe( map((_) => {
      const parial: Partial<User> = {
        faculty_id: action.universityStructure[UniversityEntitiesName.faculty],
        direction_id: action.universityStructure[UniversityEntitiesName.direction],
        // Set user account approval immediately to true, in the future the Admin role will manage user approval
        is_approved_account: true,
      };
      if (action.universityStructure[UniversityEntitiesName.group]) {
        parial.group_id = action.universityStructure[UniversityEntitiesName.group]; }
      if (action.universityStructure[UniversityEntitiesName.speciality]) {
        parial.speciality_id = action.universityStructure[UniversityEntitiesName.speciality]; }
      return parial;
    }), switchMap((parialUser) => {
      const updateUserStructure$ = this.usersHttpService.updateUser(action.user_id, parialUser).pipe(
        tap((_) => this.operationsTrackerService.trackSuccess(TrackOperations.CHANGE_USER_UNIVERSITY_STRUCTURE, action.user_id)),
        map((user) => this.store.dispatch(UsersAdapterActions.userUpdated({ user_id: action.user_id, user })))  );
      if (user.role === RoleEnum.Headman || user.role === RoleEnum.Student) {
        return from(this.roomFacade.ensureCreateGroupRoom(action.universityStructure)).pipe(
          switchMap((res) => { return from(this.userFacadeService.joinUserToRoom(res.id)).pipe( tap((_) => {
                this.operationsTrackerService.trackSuccess(TrackOperations.CHANGE_USER_UNIVERSITY_STRUCTURE,action.user_id);
              })); }) ); }
      return updateUserStructure$; }),
```

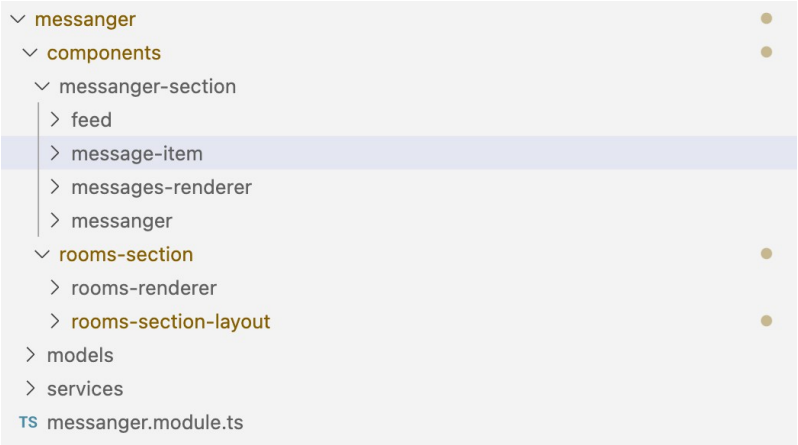
4.4.2 Реалізація функціоналу комунікації в кімнаті.

Секція веб системи що відноситься до головної сторінки з відображенням кімнат, чату обраної кімнати та інший додатковий візуальний контент цієї секції знаходиться в окремому модулі messanger який в свою чергу розділяться на messanger-section та room-section.

В messanger - секції знаходяться компоненти: message-item – відповідаючий за відображення елементу повідомлення; messages-renderer – компонент відповідаючий за логіку відображення списку повідомлень; feed – що включає в себе відображення messages-renderer компоненту та форми введення повідомлення; messanger – відповідає за відображення room-section-layout та feed компонентів.

В room-section – знаходяться наступні компоненти: rooms-renderer – відповідає за відображення списку елементів кімнат; rooms-layout – відповідає за відображення компоненту rooms-renderer.

Лістинг 4.6 – Зображення структури компонентів модулю Messanger



Алгоритми роботи з відображенням компонентів модулю Messanger залежать від дій користувача таким чином, що , першою його дією є обрання кімнати до чату та контенту якої він хоче отримати доступ. При натисканні на візуальний елемент кімнати, в параметри до строки вводу добавляється ідентифікатор кімнати обраної користувачем. Це реалізовано методом specificRoomClicked (Лістинг 4.7). В компоненті room-renderer. Також безпосередньо сам елемент кімнати спілкуєтсья з проміжним сервісом котрий визначає яка кімната є обрана. Цей сервіс називається RoomItemManagerService, його задачею є організація зв’язку між елементом кімнати та контент частиною. Використовується для відображення контенту відповідної кімнати (Лістинг 4.8).

Таким чином, інші елементи системи котрим треба дізнатися про поточну обрану кімнату, можуть зсилатись до параметрів строки вводу або підписатись на зміни щодо обрання поточної кімнати в сервісі RoomItemManagerService.

Лістинг 4.7 – Встановлення параметру в строку введення щодо поточної кімнати по натисканню користувачем на елемент кімнати.

```
async specificRoomClicked(room: Room): Promise<void> {
  if (await this.checkIfUserGroupMember(room)) {
    this.router.navigate([], {
      queryParams: {
        ...this.router.routerState.snapshot.root.queryParams,
        [MessagerRouterParams.roomId]: room.id,
      },
    });
  }
}
```

Лістинг 4.8 – Функціонал обрання кімнати що використовує RoomItemManagerService для виконання операції в room-item компоненті

```
selectItem(): void {
  if (this.relatedScopeKey) {
    this.roomItemManager.selectRequest(this.room.id, this.relatedScopeKey);
  }
}

unselectItem(): void {
  if (this.relatedScopeKey) {
    this.roomItemManager.unselectRequest(this.room.id, this.relatedScopeKey);
  }
}

ngOnInit(): void {
  if (this.relatedScopeKey) {
    this.roomItemManager
      .listenResultChangesByRoomIdAndScope(this.room.id, this.relatedScopeKey)
      .pipe(this.detacher.takeUntilDetach())
      .subscribe((data) => {
        switch (data.payload.event) {
          case RoomItemManagerMessage.CHANGE_SELECTION: {

            this.selected = (
              data.payload as RoomManagerDefaultPayload<
                RoomItemManagerMessage.CHANGE_SELECTION,
                RoomItemEventPayloadMapper
              >
            ).payload;

            this.selection.next(this.selected);
          }
        }
        this.cd.markForCheck();
      });
  }
}
```

В свою чергу, так як компонент feed підписано на зміни обрання поточної кімнати, то цей компонент реагує на обрання виконанням запитів до відображення контенту що в рамках відношення до цього компоненту, підписку на зміни зображено в лістингу 4.9.

Лістинг 4.9 – Код підписки в компоненті feed на зміну поточної кімнати користувачем

```
ngOnInit(): void {
  this.roomItemManagerService
    .listenRequestChangesByScope(this.roomsBarSectionKey)
    .pipe(
      this.detacher.takeUntilDetach(),
      switchMap((data) => {
        return this.roomFacade.getAllRooms().pipe(map((rooms) => rooms.find((r) => r.id === data.payload.room_id)));
      }),
      distinctUntilChanged((prev, curr) => prev === curr)
    )
    .subscribe((room) => {
      this.room = undefined;
      this.cd.detectChanges();
      this.room = room;
      this.cd.detectChanges();
    });
}
```

В методі ngOnInit формується підписка на прослуховування зміни обрання поточної кімнати, метод listenRequestChangesByScope сервісу roomItemManager надає можливість це зробити, в методі subscribe, в момент отримання змін, поточна кімната заміняється на нову обрану.

Після отримання поточної кімнати, всі дочірні компоненти також отримують нові дані згідно обраної кімнати та перезображають контент.

На цьому етапі, компонент MessagesRenderer, що є дочірнім компонентом feed компонента, ініціює запити через messageFacade для отримання даних про повідомлення кімнати, та для оформлення підписки на прослуховування нових повідомлень в кімнаті для їх динамічного відображення.

Лістинг 4.10 – Оформлення підписок та запитів на повідомлення в компоненті MessagesRenderer

```
async ngOnInit(): Promise<void> {
  this.loading$.next(true);

  this.operationService
    .getOperationDataFeed<Message>(TrackOperations.CREATE_MESSAGE)
    .pipe(
      this.detacher.takeUntilDetach(),
      switchMap((message) => this.messageFacade.getMessageById(message.id))
    )
    .subscribe((_) => {
      this.tempCheckPointMessagePosition = null;
      this.parentScroller.directiveRef.scrollToBottom();
    });

  this.messageFacade.setListenerForRoomMessages(this.room.id);
  this.messages$ = this.messageFacade.getMessages(this.room.id);

  const messages = await this.messageFacade.getLatestRoomMessages(this.room.id, this.bufferMessagesCount);
  if (!messages?.length) {
    this.noMessagesState = true;
  }
}
```

Методи сервісу MessageFacade згідно концепції, направлені на ініціювання «дій» NGRX що в свою чергу ініціюють запит до серверної частини та обробить результат і поверне методу.

Наприклад метод sendMessage MessageFacade сервісу, який використовується при створенні повідомлення в компоненті feed (Лістинг 4.11)

Лістинг 4.11 – Метод обробки відправлення введеного повідомлення в формі компоненту feed.

```
async sendMessage(): Promise<void> {
  this.sendingMessageLoader$.next(true);
  this.messageField.disable();
  this.fileField.disable();
  try {
    this.messagesFacade.sendMessage(this.messageField.value, this.room.id, this.fileField.value);
    this.messageField.reset();
  } finally {
    this.sendingMessageLoader$.next(false);

    this.fileField.reset();
    this.messageField.reset();

    this.messageField.enable();
    this.fileField.enable();
  }
}
```

Метод sendMessage ініціює відправлення NGRX дії sendMessage, в свою чергу ця дія перехоплюється ефектом що слухає виникнення дії цього типу, а метод ефекту метадані з дії приведе до комфортного виду для формування запиту до http сервісу що є остаточною точкою перед відправкою запиту на створення повідомлення.

Метод createRoomMessage сервісу MessageHttpService (Лістинг 4.12) перед відправленням запиту формує тіло повідомлення доповнюючи даними що є даними домен сутності, наприклад, час відправлення повідомлення, та поточний користувача отримуються безпосередньо перед створенням остаточного об’єкту повідомлення що буде записано до бази даних за допомогою метода .set що надається AngularFire SDK.

Лістинг 4.12 – Метод createRoomMessage Http сервісу що відноситься до формування запиту до створення повідомлення користувачем та виконання цього запиту.

```
createRoomMessage(room_id: string, message_text: string): Observable<Message> {
  return this.authService.getCurrentUserObservable().pipe(
    take(1),
    switchMap((user) => {
      const now = new Date() as any;
      const id = this.afs.createId();
      const message: Message = {
        id,
        room_id,
        text: message_text,
        sender_id: user.uid,
        type: MessageType.REGULAR,
        time: now,
        lastOperationTime: now,
      };
      return from(this.getMessagesCollectionReference(room_id).doc(id).set(message)).pipe(
        switchMap(() => {
          return this.getMessagesByQuery(room_id, (col) => col.where('id', '==', id), false).pipe(mergeMap((x) => x));
        })
      );
    })
  );
}
```

Розглянемо також функціонал прикріплення файлів до повідомлення та збереження їх в сховищі FireBase. Пропустимо частину рішення завдання на рівні компоненту, фасаду, ефекту, та перейдемо до розгляду важливішої частини коду завдяки котрій і вдається вирішити завдання з приєднанням відправити файл до сховища, отримати посилання, приєднати посилання до повідомлення та відправити його. А вже на рівні компоненту реалізоване відображення та можливість по натиску на файл його скачати по посиланню.

Тож в методі createRoomMessageWithAttachments сервісу MessageHttpService виконується ключова логіка що об’єднує в собі послідовні операції спочатку з обробки файлів та їх завантаження до

сховища, а далі записування сформованого повідомлення до бази даних (Лістинг 4.13 та 4.14).

Лістинг 4.13 – Перша частина методу createRoomMessage WithAttachments.

```
createRoomMessageWithAttachments(
  room_id: string,
  message_text: string,
  attachments: FileList
): Observable<MessageWithAttachments> {
  const attachmentsRequestArray: Observable<MessageAttachment>[] = [];
  for (let i = 0; i < attachments.length; i++) {
    const attachment = attachments.item(i);
    const fileIdAgainstName = this.afs.createId();
    const fileFullPath = this.fileStorageService.createFileFullPath(
      ROOM_ATTACHMENTS_PATH_FACTORY(room_id),
      fileIdAgainstName
    );
    attachmentsRequestArray.push(
      this.fileStorageService
        .uploadFileToStorageByFullPath(fileFullPath, attachment, {
          contentType: attachment.type,
          customMetadata: { original_name: attachment.name },
        })
        .snapshotChanges()
        .pipe(
          take(1),
          map((_) => {
            return { file_path: fileFullPath, id: fileIdAgainstName, name: attachment.name };
          })
        )
    );
  }
```

Лістинг 4.14 – Друга частина методу createRoomMessage WithAttachments

```
return combineLatest(attachmentsRequestArray).pipe(
  switchMap((attachmentsFiles) => {
    return this.authService.getCurrentUserObservable().pipe(
      take(1),
      switchMap((user) => {
        const id = this.afs.createId();
        const now = new Date() as any;
        const message: MessageWithAttachments = {
          id,
          room_id,
          text: message_text,
          sender_id: user.uid,
          attachments: attachmentsFiles,
          type: MessageType.ATTACHMENTS,
          time: now,
          lastOperationTime: now,
        };
        return from(this.getMessagesCollectionReference(room_id).doc(id).set(message)).pipe(
          switchMap(() => {
            return this.getMessagesByQuery(room_id, (col) => col.where('id', '==', id), false).pipe(
              mergeMap((x) => x)
            );
          })
        );
      })
    ) as Observable<MessageWithAttachments>;
  })
);
}
```

4.4.3 Функціональні можливості роботи з кімнатами

Як вже було зазначено, після того як користувач ролі «студент», отримує доступ до головної сторінки з контентом, його автоматично під’єднує до кімнати. В свою чергу для користувачів іншої ролі, такої як «university staff» автоматичного під’єднання до кімнати не відбувається, через те що при реєстрації користувачу не потрібно вказувати до якої групи він має приналежність. В цьому випадку користувач сам в праві визначити до якої кімнати -бесіди увійти. Тож при натисканні на візуальний елемент кімнати, у випадку якщо користувач не під’єднаний до неї, буде показано модельне вікно з підтвердженням входу до кімнати, що виглядає наступним чином (Рисунок 31), а логічна операція з підтвердження вступу користувача в кімнату зображена на лістингу 4.15

Лістинг 4.15 – Метод запиту вступу користувача до кімнати

```
joinCurrentUserToRoom(room_id: string): Observable<Room> {
  return this.authService
    .getCurrentUserObservable()
    .pipe(take(1))
    .pipe(
      switchMap((user) => {
        if (user.rooms?.length) {
          const roomIdAlreadyIncluded = user.rooms.includes(room_id);

          if (!roomIdAlreadyIncluded) {
            user.rooms.push(room_id);
          }
        } else {
          user.rooms = [room_id];
        }

        return this.usersHttpService.updateUser(user.uid, user).pipe(
          switchMap((_) => {
            return this.getSpecificRoomById(room_id).pipe(
              switchMap((room) => {
                if (room.users?.length) {
                  const userIdAlreadyIncluded = room.users.includes(user.uid);

                  if (!userIdAlreadyIncluded) {
                    room.users.push(user.uid);
                  }
                } else {
                  room.users = [user.uid];
                }
                return this.updateRoom(room_id, { users: room.users });
              })
            );
          })
        );
      })
    );
}
```

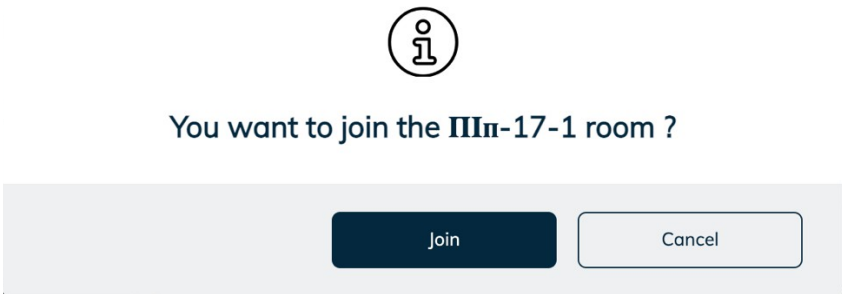


Рисунок 4.18 – Зображення модального вікна підтвердження входу до кімнати

Після успішного підтвердження входу та виконання запиту до серверної частини, одразу зображається чат кімнати.

Цікавою можливістю є надання користувачам функціоналу огляду розкладу групи до якої кімната належить, при натисканні спицифічної кнопки у «шапці» кімнати (Рисунок 4.16) виконується код формування

запиту до cist.nure.ua в реультаті який генерує розклад для перегляду, після генерації розкладу, користувачу відкривається додаткова вкладка в браузері для перегляду цього розкладу.

Рисунок 4.19 – Кнопка формування та перегляду розкладу кімнати.

Лістинг 4.16 – Код формування запиту до cist.nure.ua для отримання розкладу групи.

```
viewSchedule(): void {
  const currTime = new Date(Date.now());

  const timeFrom = this.datepipe.transform(currTime, 'dd.MM.yyyy');
  const timeTo = this.datepipe.transform(currTime.setMonth(currTime.getMonth() + 8), 'dd.MM.yyyy');

  window.open(
    `https://cist.nure.ua/ias/app/tt/f?p=778:201:3750290334261274:::201:P201_FIRST_DATE,P201_LAST_DATE,P201_GROUP,P201_POTOK:${timeFrom},${timeTo},${this.room.room_details.group_id},0:`,
    '_blank'
  );
}
```

5. ТЕСТУВАННЯ ПРОЕКТУ

5.1 Тестування послідовних функціональних можливостей системи

У рамках тестування розробленої системи комунікації для ХНУРЕ між викладачами та студентами, буде проведена демонстрація виконання алгоритмів системи починаючи від авторизації користувача – до написання повідомлення в кімнаті.

Проведемо тестування на прикладі реєстрації користувача студента за допомогою serhii.roh@nure.ua, моєї власної пошти ХНУРЕ.

При введенні некоректної адреси пошти або якщо пошта не є @nure.ua поштою, то форма буде заблокована для надсилання та повідомлення про валідаційні проблеми будуть зображені користувачу (Рисунок 5.1)

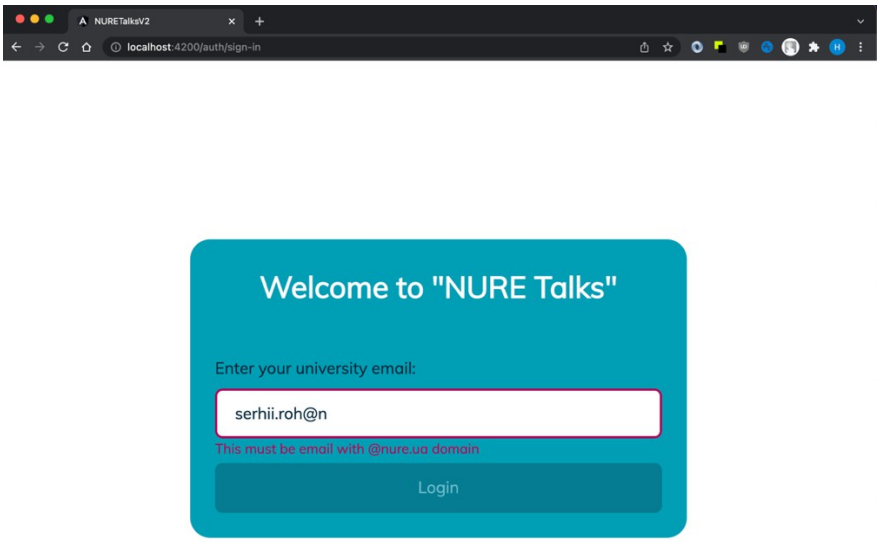


Рисунок 5.1 – Зображення вікна з авторизаційною формою та валідаційним повідомленням

Продовжимо допис пошти для проходження валідації домену пошти та натискаємо Login кнопку форм.

Після натискання кнопки бачимо інформаційне повідомлення (Рисунок 5.2) про те, що користувачу було надіслано авторизаційне посилання на пошту по котрому треба перейти (Рисунок 5.3)

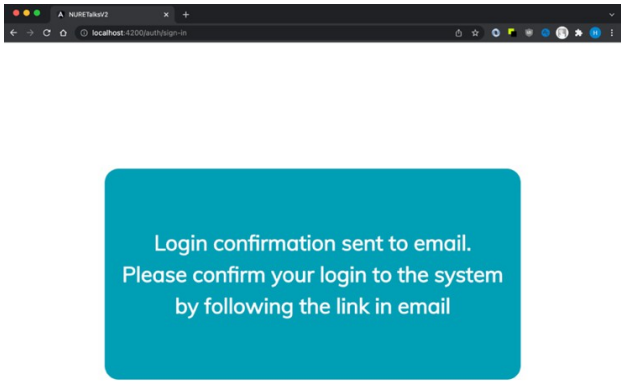


Рисунок 5.2 – Інформативне повідомлення для користувача після відправлення форми із вказаною поштою.

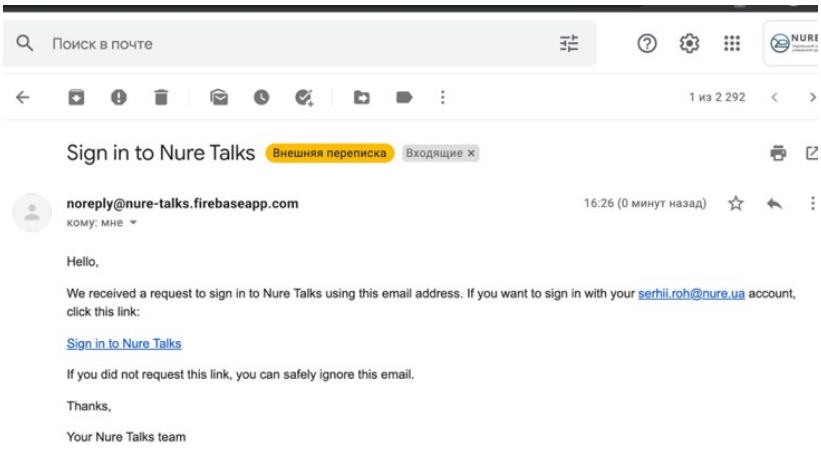


Рисунок 5.3 – Отримане повідомлення підтвердження автентифікації після віправлення форми із вказаною поштою.

Переходимо по вказаному посиланню в повідомленні про підтвердження автентифікації, посиланням є гіпертекст «Sign in to Nure Talks» зображене синім кольором (Рисунок 5.3).

Після переходу по посиланню, система проводить автентифікації користувача та направляє його на сторінку 2-гого етапу підтвердження даних користувача, це форма з вибором приналежностей користувача та обрання ролі. Проведемо заповнення форми згідно реальних даних студента, а тобто, оберу роль студент та приналежність до групи СКСм-20-1. Результат заповнення форми виглядає наступним чином на рисунку 5.4. Також допоки обов’язкові поля не будуть обрані, кнопка відправки форми буде недоступна користувачу.

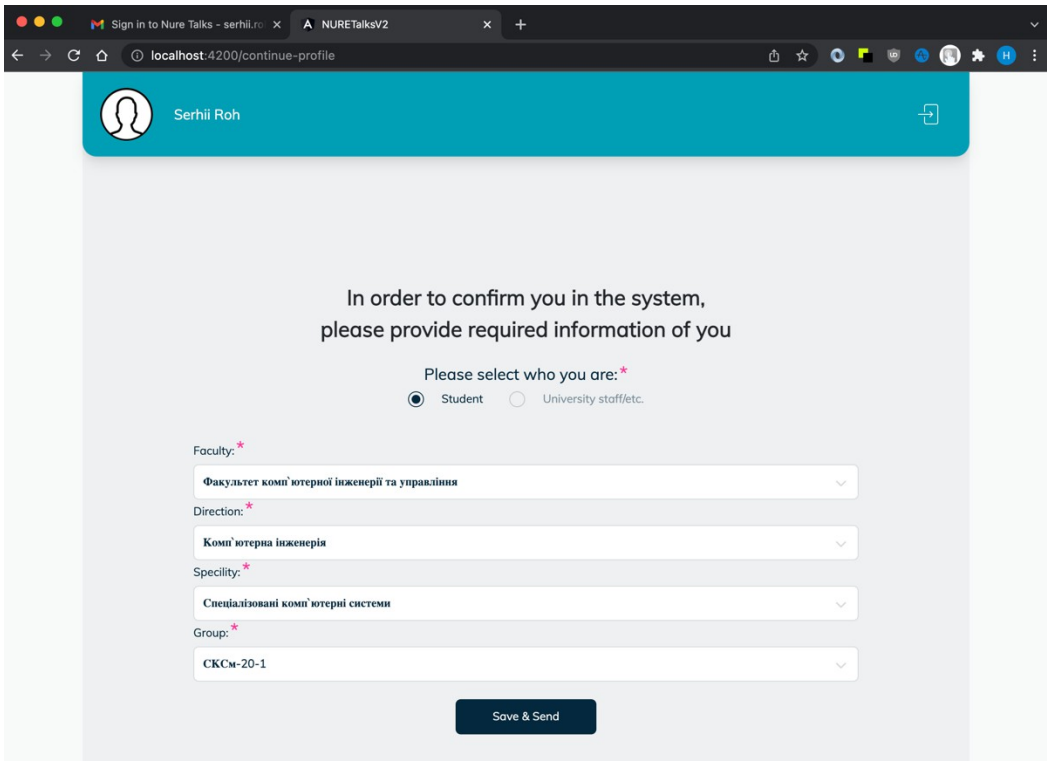


Рисунок 5.4 – Зображення заповненої форми другого етапу авторизації до системи.

По натисканню кнопки Save & Send користувачем, ініціюється процес підтвердження користувача в системі та одразу перенаправляє його на основну сторінку системи (Рисунок 5.5) де він здатен обрати кімнату групи до якої відноситься.

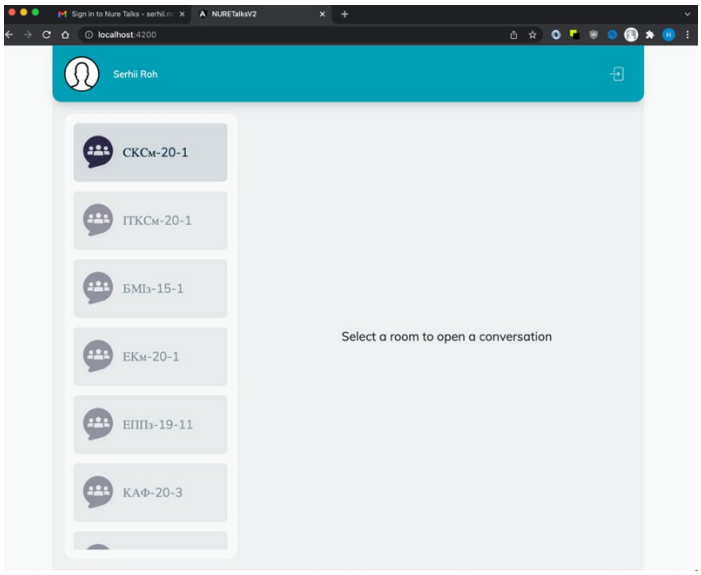


Рисунок 5.5 – Зображення головної сторінки після успішної авторизації користувачем

Натискаємо на елемент кімнати «СКСм-20-1», що провокує систему відобразити контент кімнати (Рисунок 5.6)

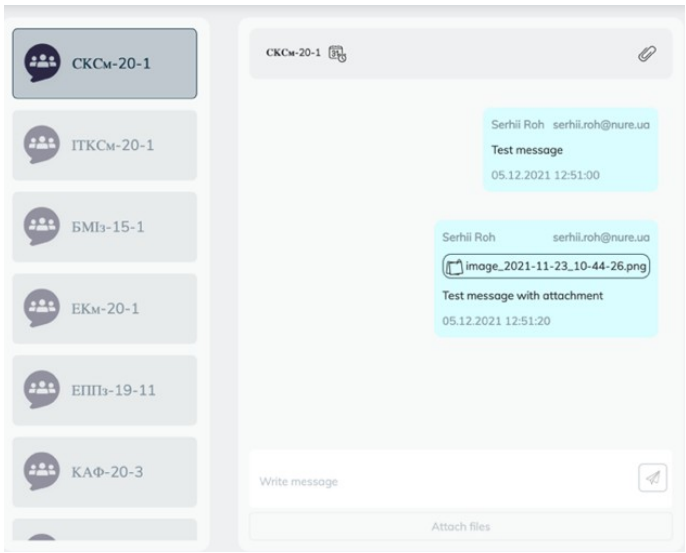


Рисунок 5.6 – Зображення контенту обраної кімнати «СКСм-20-1»

Далі проведемо тестування роботу функціонування відправлення повідомлення, його відображення, та елементів керування форми.

Введемо повідомлення з текстом, наприклад - «Тестове повідомлення до пункту 5 ТЕСТУВАННЯ ПРОЕКТУ». Натиснемо кнопку під строкою вводу що називається «Attach files», по натисканню відкривається файловий

оглядач, що дає можливість обрати 1 або більше файлів, наприклад оберемо 2 будь яких файлу та додамо, в результаті форма з повідомленням перед відправкою має наступний вигляд, що зображено на рисунку 5.7.

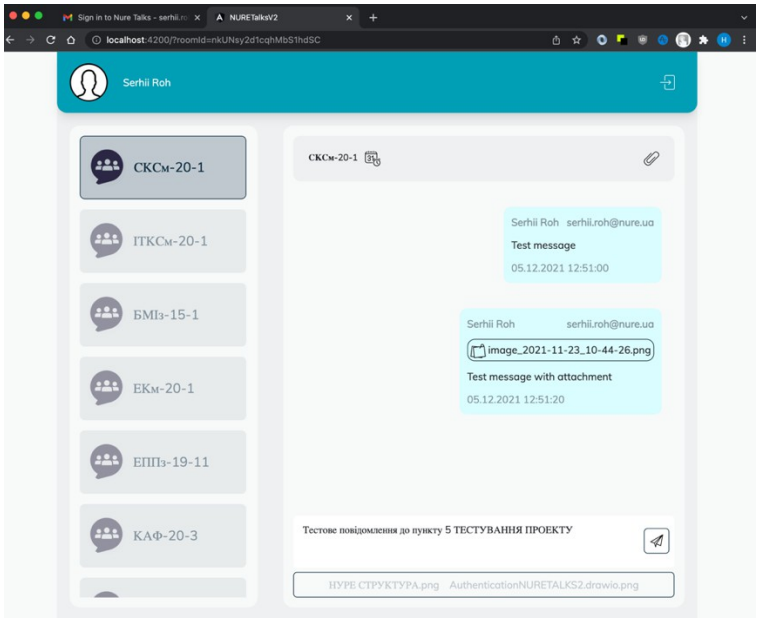


Рисунок 5.7 – Зображення заповненої форми відправлення повідомлення, з доданими двома файлами.

На рисунку 5.7 бачимо що форма відображає введений текст, а кнопка Attach Files тепер замість власного напису відображає ім'я двох файлів щ обули додані.

Натискаємо кнопку що має вигляд бумажного літачка, це ініціює процес надсилання повідомлення, по успішному результату обробки запиту, бачимо нове повідомлення в вікні повідомлень, та те що форма введення повідомлення та кнопка додавання файлу повертається в початковий «чистий» стан (Рисунок 5.8).

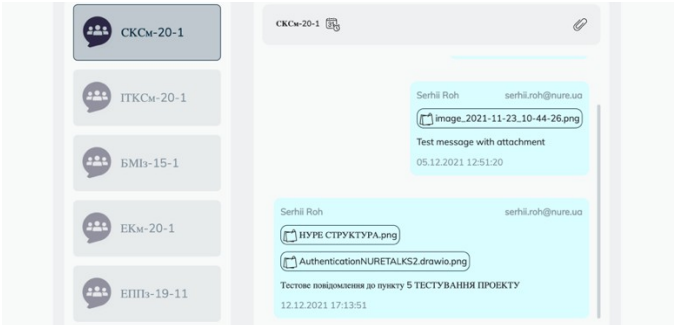
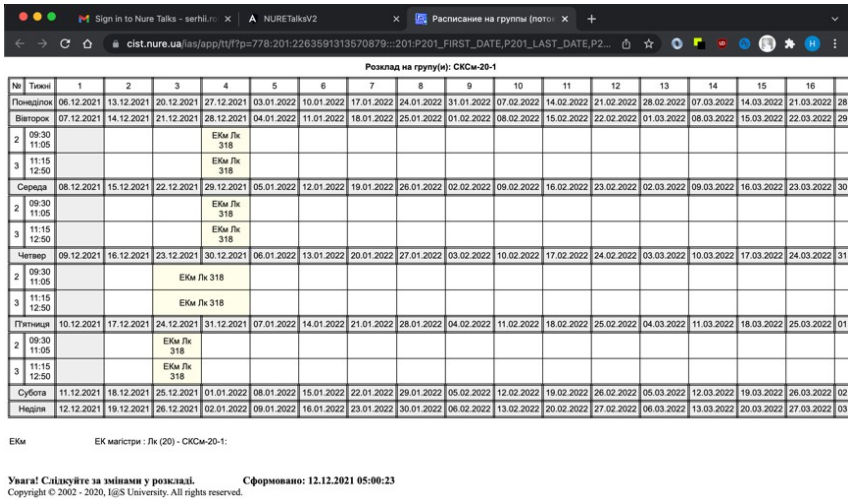


Рисунок 5.8 – Зображення відображення надісланого повідомлення з прикріпленими двома файлами.

Таким чином тестування основного функціоналу пройдено успішно. Ще додатково протестуємо перегляд розкладу кімнати групи. Натискання на ім’я групи в шапці секції контенту (секція з повідомленнями) згенерує посилання на розклад проміжком в рік для окремої групи. Протестуємо на прикладі кімнати групи СКСм-20-1, після натиску на кнопку з ім’ям у шапці, користувача перенаправляє в нову вкладку де відобразиться розклад (Рисунок 5.9).


Sign in to Nure Talks - serhii.r | NURETalksV2 | Расписание на группы (пото

cist.nure.ua/ias/app/tt/f?p=778:201:2263591313570879:::201:P201_FIRST_DATE,P201_LAST_DATE,P2...

Розклад на групу(и): СКСм-20-1

№	Тижні	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
Понеділок		06.12.2021	13.12.2021	20.12.2021	27.12.2021	03.01.2022	10.01.2022	17.01.2022	24.01.2022	31.01.2022	07.02.2022	14.02.2022	21.02.2022	28.02.2022	07.03.2022	14.03.2022	21.03.2022	28
Вівторок		07.12.2021	14.12.2021	21.12.2021	28.12.2021	04.01.2022	11.01.2022	18.01.2022	25.01.2022	01.02.2022	08.02.2022	15.02.2022	22.02.2022	01.03.2022	08.03.2022	15.03.2022	22.03.2022	29
2	09:30 11:05				ЕКм Лк 318													
3	11:15 12:50				ЕКм Лк 318													
Середа		08.12.2021	15.12.2021	22.12.2021	29.12.2021	05.01.2022	12.01.2022	19.01.2022	26.01.2022	02.02.2022	09.02.2022	16.02.2022	23.02.2022	02.03.2022	09.03.2022	16.03.2022	23.03.2022	30
2	09:30 11:05				ЕКм Лк 318													
3	11:15 12:50				ЕКм Лк 318													
Четвер		09.12.2021	16.12.2021	23.12.2021	30.12.2021	06.01.2022	13.01.2022	20.01.2022	27.01.2022	03.02.2022	10.02.2022	17.02.2022	24.02.2022	03.03.2022	10.03.2022	17.03.2022	24.03.2022	31
2	09:30 11:05			ЕКм Лк 318														
3	11:15 12:50			ЕКм Лк 318														
П'ятниця		10.12.2021	17.12.2021	24.12.2021	31.12.2021	07.01.2022	14.01.2022	21.01.2022	28.01.2022	04.02.2022	11.02.2022	18.02.2022	25.02.2022	04.03.2022	11.03.2022	18.03.2022	25.03.2022	01
2	09:30 11:05			ЕКм Лк 318														
3	11:15 12:50			ЕКм Лк 318														
Субота		11.12.2021	18.12.2021	25.12.2021	01.01.2022	08.01.2022	15.01.2022	22.01.2022	29.01.2022	05.02.2022	12.02.2022	19.02.2022	26.02.2022	05.03.2022	12.03.2022	19.03.2022	26.03.2022	02
Неділя		12.12.2021	19.12.2021	26.12.2021	02.01.2022	09.01.2022	16.01.2022	23.01.2022	30.01.2022	06.02.2022	13.02.2022	20.02.2022	27.02.2022	06.03.2022	13.03.2022	20.03.2022	27.03.2022	03

ЕКм ЕК магістри : Лк (20) - СКСм-20-1:

Увага! Слідкуйте за змінами у розкладі. **Сформовано: 12.12.2021 05:00:23**

Рисунок 5.9 – Згенерований розклад групи СКСм-20-1

5.2 Комунікації на симульованому прикладі реальної ситуації

Як тестовий приклад розглянемо типово можливу ситуацію інформування студентів про деякі зміни в поточному освітньому процесу, на приклад, здвиг проведення заняття на інший час та зміна місця проведення відеоконференції. Також для повноти, приклад націлений на зображення усіх поточних особливостей системи: передачі файлу в рамках групи, перегляд сформованого розкладу для групи від cist.nure.

Вхідні дані для проведення тесту наступні:

Ім’я	Фамілія	Пошта ХНУРЕ	Роль	Принадлежність
Олександр	Шкіль	oleksandr.shkil@nure.ua	Викладач	Кафедра АПОТ
Сергій	Рог	serhii.roh@nure.ua	Студент	Група СКСм-20-1
Дмитро	Пашков	dmytro.pashkov@nure.ua	Студент	Група СКСм-20-1

Таблиця 5.1– Фрагмент бази студентів та викладачів що беруть участь в комунікації

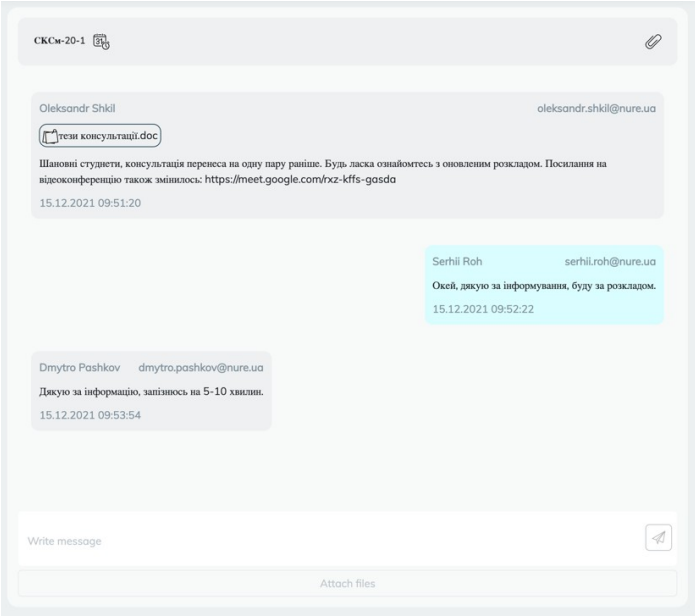


Рисунок 5.10 – Результат проведеного тесту.

В описаній тестовій ситуації, викладачу потрібно інформувати всіх студентів групи про зміну в розкладі заняття що наближається, пересилає файл всім студентам для ознайомлення до початку заняття та очікує від студентів повідомлень про підтвердження отримання ними інформації. В свою чергу студенти інформують викладача про отримання ними інформації та в індивідуальному порядку ознайомлюються з надісланим файлом та звіряють зміни в розкладі групи.

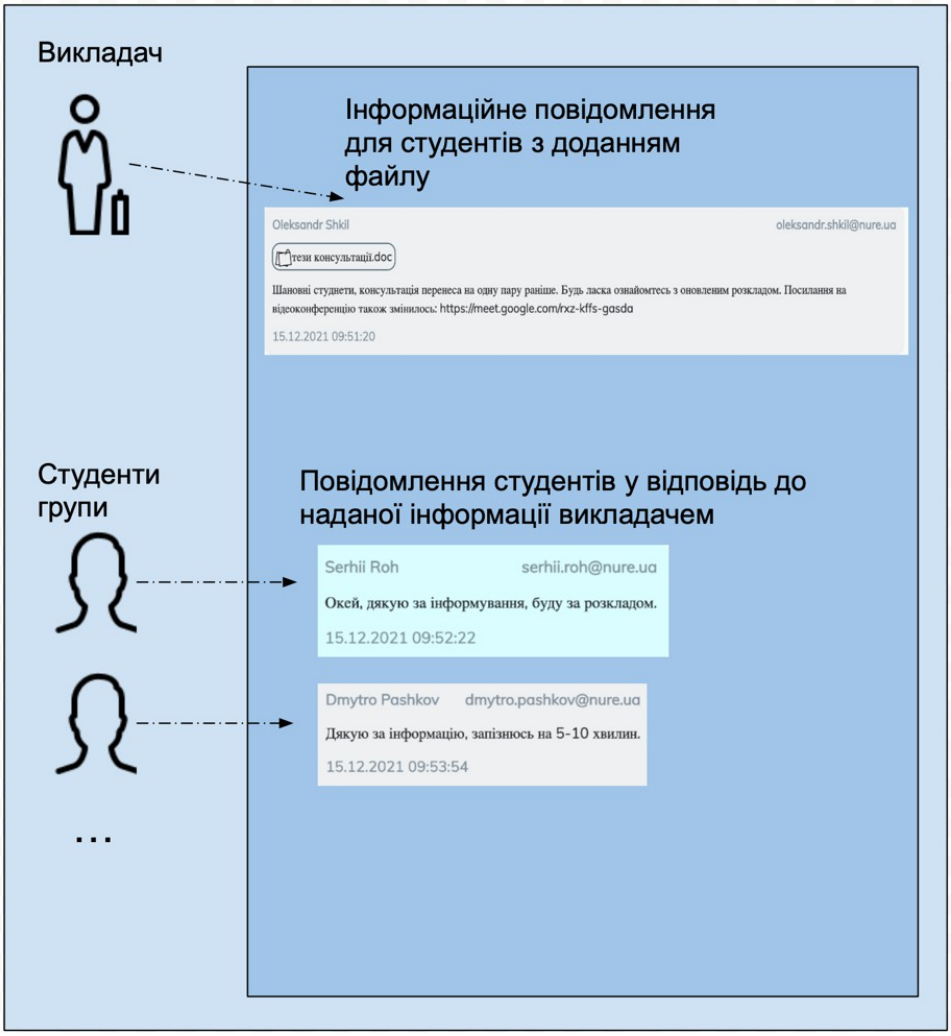


Рисунок 5.11- Схема комунікації тестового прикладу.

ВИСНОВКИ

В результаті виконання науково-дослідницької роботи було проведено аналіз сучасних рішень побудови комунікаційних систем, із переліку подібних собі технологій були обрані най підходящі, для побудови комунікаційної веб-комунікаційної системи ХНУРЕ, згідно об’єктивного та суб’єктивного аналізу.

Зібрано матеріал-завдання з огляду на потреби викладачів в комунікаційних системах що описало завдання що-до побудови системи.

В роботі досягнуто цілей з побудови комунікаційної системи для ХНУРЕ згідно до специфіки організації структури університету та потреб потенційних користувачів системи, викладачів та студентів. А саме, вдалось реалізувати веб-комунікаційну систему в якій: доступ мають лише користувачі з електронною поштою nure домену; комунікація включає в себе текстовий вид комунікації та передачу файлів; користувачі поділяються на ролі які дають доступ до певних функціональних можливостей в системі.

Розроблено архітектурне рішення за результатами аналізу технологій що можуть бути залучені в побудову такого роду системи, де таке рішення включає в себе: організацію структури збереження даних в базі даних реального часу, організація комунікації між рівнями системи, та між компонентами клієнтської частини системи. Розроблено користувацький інтерфейс для веб системи. Система безпосередньо не використовує та не очікує паролі користувачів для отримання доступу до системи.

Система є доведеною до працездатного виду в якій реалізовані основні потреби користувачів. Продукт може бути залучено до життя університету та бути використаною в тому вигляді в поточному вигляді.

Рекомендацією з подальшого залучення системи є оформлення хостингу для цієї системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Шкіль О. С. Хмарні технології у підтримці самостійної роботи студентів / В.І. Каук, В.О Гребенюк, О.С. Шкіль // Вісник Національного університету «Львівська політехніка». Інформатизація вищого навчального закладу. – 2016.– №853. – С. 11-17.
2. Офіційна документація Angular Framework [Електронний ресурс] / Introduction to the Angular Docs - Режим доступу: www/ URL: https://angular.io/docs – 10.06.2021 г. – Загл. з екрану.
3. Офіційна документація Reactive Extensions Library for Java Script (RxJS) [Електронний ресурс] / Introduction- Режим доступу : www/ URL: https://rxjs-dev.firebaseapp.com/guide/overview – 06.06.2021 г. – Загл. з екрану.
4. Офіційна документація по роботі з хмаровим сервісом Google Firebase та його елементами [Електронний ресурс] / Learn Firebase fundamentals - Режим доступу: www/ URL: https://firebase.google.com/docs/guides – 04.07.2021 г. – Загл. з екрану.
5. Опис загальних відомостей мови TypeScript [Електронний ресурс] / TypeScript - Overview - Режим доступу: www/ URL: https://www.tutorialspoint.com/tutorial_view.htm?cid=typescript&pid=typescript_overview.htm– 21.03.2021 г. – Загл. з екрану.
6. NGRX state management system [Електронний ресурс] / What is NgRx?- Режим доступу: www/ URL: https://ngrx.io/docs – 05.05.2021 г. – Загл. з екрану.
7. Tailwind дизайн фреймворк [Електронний ресурс] / Get started with Tailwind CSS- Режим доступу: www/ URL: https://tailwindcss.com/docs – 06.05.2021 г. – Загл. з екрану.